Alma Mater Studiorum · University of Bologna

Department of Computer Science and Engineering - DISI
Two year Master degree in Computer Engineering

Interaction and Behaviour Evaluation for Smart Homes: Data Collection and Analytics in the ScaledHome Project

Master thesis in Mobile Systems M

Supervisor:
Prof. Paolo Bellavista

Correlator:
Prof. Damla Turgut

Author:
Matteo Mendula

Session VI
Academic year 2018/2019

Contents

Abstract

Nowadays more and more devices are becoming "smart", in fact they can take autonomous decision and interact proactively with the surrounding environment.
Smart home is just one of the most popular terms related with this relevant change we are witnessing and its relevance in this project is mainly due to the fact that the residential sector account an important percentage in terms of energy consumption.
New ways to share and save energy have to be taken into account in order to optimize the usage of the devices needed by houses to make the environment cozy and comfortable for their inhabitants.
The work done with Professor Turgut's team has improved the knowledge in the smart home system area providing a scalable and reliable architecture, a new dataset and an example of application of these data useful to save energy while satisfying the demands of its inhabitants.

Introduction

Since the days of prehistory the humankind has always looked for a shelter in order to protect itself from elements and bigger and stronger animals. Nowadays our habitations are not caves or huts anymore, this is because we are now able to force the environment to our will and to create houses, apartments, skyscrapers and then cities. Today houses are not only mere buildings, they are smart systems able to react to external events in order to improve the lifestyle quality of its inhabitants. They can acquire data from outside, predict actions performed by users and then react.
It could be reasonable to think that the introduction of so many additional devices can lead to higher power consumption. This is in part correct, but exploiting the smart capabilities of the introduced devices a new cooperative system can be designed which takes into account the desired state by the inhabitants, their behaviours, the outside and inside environment and then acts autonomously in an appropriate way.
By doing this the smart home systems can be used not only to ease the life of its living beings, but also to make power saving reasoning based on the different circumstances and then operate accordingly to save and share energy.
Since the smart home environment is composed by interconnected devices it is quite easy to collect large amount of data because the M2M(Machine to Machine) communication is already set to make the system working properly. This information flow can be forwarded to an artificial intelligence algorithm able to identify the best action to perform in order to react to the current circumstances.
Data collection plays an essential role because it is needed to achieve the needed accuracy and reliability of a predictive model, but how is it possible to simulate the variegated and heterogeneous set of phenomena that have to do with housing? Scaled models allow the simulation of events and circumstances that would be prohibitively expensive to do with the corresponding entities in the real world.
While scaled models had been extensively used in fields such as architecture, structural engineering or fluid dynamics, ScaledHome is novel in its goals of modeling a range of phenomena with different scaling similitude properties such as: the physical properties of the home, the heating/cooling balance of the interior of the home, as well as energy saving policies which consider the set state by the inhabitants as a main requirement to accomplish.

In order for a green home to take the optimal actions that lower cost and minimize environmental impact, it needs a high quality predictive model based on the various actions that can be taken. Due to the complex interactions between internal and external factors, the geometry of the home and the actuators, such a predictive model is difficult to build. For instance, opening a window at night might lower the temperature, while during the day it might raise it, especially if the window is on the sunny side of the home. Naturally, this depends on the location (Arizona versus Minnesota), season and other factors (is the home shaded by a tree?).
ScaledHome provides a simulation environment where multiple real world scenarios can be mapped in such a way that it is not necessary to install real size actuators and sensors inside real size houses. This will be not only expensive, but it will lead to the need of setting multiple copies of the same system in each location where the real product will be applied since each geographical location has its own specific peculiarities.
The resulting system consists of several agents that actively take actions modifying the environment in order to allow us to model a miniaturized home with an accuracy and realism significantly exceeding a simulation model. Despite real-world homes remain the gold standard in terms of accuracy, ScaledHome represents a flexible and adaptive simulation environment to collect and analyze data and perform predictions about the state of the house in order to react to the outside circumstances by taking appropriate actions in an autonomous way.
Since a good part of the energy consumption is due to the residential usage, the optimization of the employment of the devices inside the house can lead to a better solution in terms of less environmental impact and lower electricity bills. Despite the fact that the progress of science and technology is causing higher energy demands this kind of optimization is a proof of how the technological development do not consist of just the expansion of energy-hungry data centers, but it can be redirected to greener purposes.
The work done with Professor Turgut's team led to a an heterogeneous system composed of different controllers and agents coordinated by a central entity in charge of communicating with all of them and providing high level interfaces to interact with. This has been done in order to design a flexible and scalable system which can be extended by adding new entities without changing the essential structure of the other components.
First we designed an architecture suitable for our simulation and testing purposes able to scale up and serve a desirable increasing traffic generated by third parties since one of the main goals of the project has been to encourage different teams to participate to the project by running their own simulation and so increase the chances of finding the best solution as possible. This has been done by following the software design principles theory as a basement to apply the most suitable and disruptive technologies.
The business logic of the system has been delegated to a single entity which has been scaled up by using micro-services techniques to guarantee reliability and fault tolerance of the whole system.
The containerization of the central entity has been followed by the deployment of several replicas of the service in such a way to load balance the traffic among them.
Then we collected sufficient data to analyze and identify the constrains of the ScaledHome system in order to map real world scenarios inside our scaled model. This part had a great importance in the development of the project because without a reliable mapping to the real world any result even if relevant would not be applicable in a real environment. We had to take into account both the scaling factors and the bounds of the system in order to find a solid mapping between the actual and the simulation scenario.

Thanks to that analysis we have been able to simulate a real world day inside our simulation environment with a reasonable matching.
Then we used the data collected to train several supervised machine learning models in order to find the best predictor to use in order to forecast the consequences of possible sets of actions on the system.
A specific entity called hyperparameter tuner has been implemented to tune the hyperparameters of each model in a flexible way. In fact it can be integrated by adding new machine learning model implementations with the minimal effort. Each model can be set by providing the most promising hyperparameters to evaluate so the hyperparameter tuner can train on all of them using a grid search algorithm in order to find the ones which will lead to the best accuracy.
After implementing our custom hyperparameter tuner we checked the performances and the usability of one new technologies that can be used to ease the tuning by providing higher interfaces for the tuning.
Each machine learning model has been trained after the tuning with more data and then we have been able to identify as the most suitable to our purposes the one with the higher accuracy.
Finally the best model has been used to explore the space of the actions that can be performed at each interval inside the house to reach the target state while minimizing the energy consumption of the whole system.
The final result represents a great improvement in the smart home area since it provides a well scalable architecture and a policy implementation of an energy saving autonomous planner.
The reminder of this paper is organized as follows. A first section introduces the topic and the reasons behind the project. Then the main software engineering principles adopted are underlined in order to support the understanding of the final implementation.
After that the essential machine learning concepts have been explained to provide to the reader a complete overview of the subject which will be needed to interpret the outcomes appropriately.
The final section documents the resulting product reporting some relevant modules of the implementation and commenting the most important parts in addition to describing the significant goals accomplished.

1 ScaledHome

1.1 IoT

It is an old term, probably its first appearance in 1999 by Kevin Ashton and at that time it was mainly applied to RFID, supply chain, and the Internet. It stands for internet of things and it is a general term often used to address all the kind of heterogeneous entities interconnected by the same network. A "thing" can be any physical or digital objects that can be monitored or controlled and it is generally classified into two different categories: actuators and sensors; depending on if they perform actions or if they collect data respectively.
While the human being has limited time, attention and accuracy, computers fed with sufficient data are able to greatly reduce wast, loss and costs in many different areas.
In 2009 it has been realized how the conventional diagrams of the Internet was keeping out the most important gateway of information: people.
In this sense the lifestyle of each single individual is a great source of information which can be easily collected. In fact it is going to spread the idea that the size of the network where interconnected device operate will shrink drastically. Actually we now refer as PAN (Personal Area Network) a computer network for interconnecting devices centered on an individual person's work-space.
Examples of objects that can fall into the scope of Internet of Things include connected security systems, thermostats, cars, electronic appliances, lights in household and commercial environments, alarm clocks, speaker systems, vending machines and more. A thing in the IoT can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low or any other natural or man-made object that can be assigned an Internet Protocol (IP) address and is able to transfer data over a network.
According with SAP/Intel and Ericsson statistics the number of connected devices will reach 50-100 billions by the end of 2020. It is reasonable to think that this trend is going to change not only the big industry workflow but also the lifestyle of simple consumers.
IoT architectures can consists of one horizontal layer/platform managing heterogeneous information in an efficient manner or they can be composed by merging several different applications to provide specific information in a market-tailored manner.
One of the common consideration about IoT is its relationship with Cloud models. They provide resources on demand "as a service", but they are not designed for the volume, variety, and velocity of data that the IoT generates.
In order to overcome this limit Fog Computing can be considered. Its primary goal is to process as much data as possible on the edges of the network. This process can became challenging when the devices at the edge of the network are constrained in their computation capabilities. What cannot be processed at the edges is then passed to the Cloud infrastructure which will perform the most demanding operations in terms of resources usage.

1.2 The Smart home scenario

A classical definition for smart home system could be the following: a smart home is a home equipped with lighting, heating, and electronic devices that can be controlled remotely by phone or computer; and that was completely correct four or five years ago. Nowadays smart houses are much more that just an interface for remote control, they use what in the literature is called automation scheduling to plan different actions that have to be performed. These actions can be manually declared by a human or they can be the final result of an artificial reasoning process which has as main goal to reach a target state. Despite the apparent simplicity, a smart home system became more and more complicated every time we introduce a new smart device in the same network. But having everything connected to the same hub can be incredibly convenient, nonetheless. Think of it this way: each time you are coming back home and you are about a mile away from it, your smartphone tracks your destination and interacts with your house to inform it about your arrival; thanks to that the space heating system increases the temperature inside the house, the lights turn on, the security system disarms and the door opens. So each agent inside the house has a specif task to perform and given a target state to achieve (make the house cozy and welcoming) in such a way that given a context (your arrival) the system is able to select which is the best action to perform.

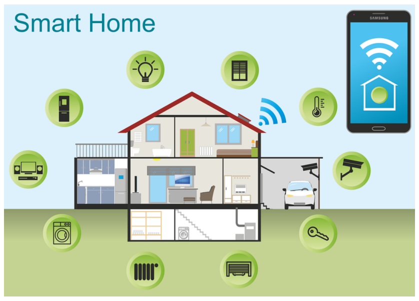


Figure 1: Smart home schema

In order to do that the computation needed by the reasoning process can be remarkable both if we are distributing it among the set of smart devices or we are focusing all the business logic into a central broker entity.
It might seems that because of new entities have to be introduced inside the system we are increasing the amount of energy consumed. In general terms this is correct, but another goal of the smart home systems is to optimize energy demands by for example keeping the temperature lower when no one is in the house.
So, instead of consuming more energy to have more comforts and more support from the smart home system it is possible to obtain all these advantages while saving energy just considering it as an optimization problem.

1.3 HAN: Home Area Network

Home Area Network connects smart appliances, thermostats and other electric devices to an energy management system.
The home area network (HAN) communicates with various smart devices to provide energy efficiency management and demand response. Compared with Neighbourhood area network (NAN) and Wide area network (WAN), the former connects multiple HANs to local point while the latter provides communication links between the NANs and the utility systems to transfer information. As one of the core technologies, an efficient, reliable, and secure communication network plays an important role in realization of all the goals of smart grid NANs.

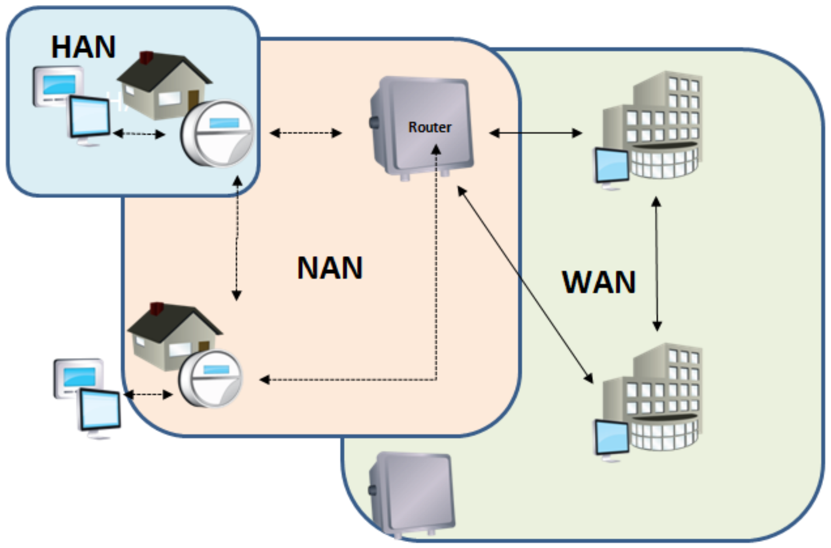


Figure 2: HAN, NAN and WAN relationship

1.4 Energy consumption due to air condition and space heating

Most of the time when we think about energy consumption we are focusing our attention on huge data centers and big companies or maybe vehicles and public infrastructures, but we forget about the impact that each of us has on the environment and the energy demands around the world.
According with the U.S. Energy Information Administration statistics "in 2018, the residential and commercial sectors accounted for about 40% (or about 40 quadrillion British thermal units) of total U.S. energy consumption." [1] where for BTU is a non-SI which is defined as the amount of heat required to raise the temperature of one pound of water by one degree Fahrenheit.
The graph below come from the same source [1] and it shows as air conditioning and space heating have the highest percentage of energy consumption when we are considering the residential usage of electricity.

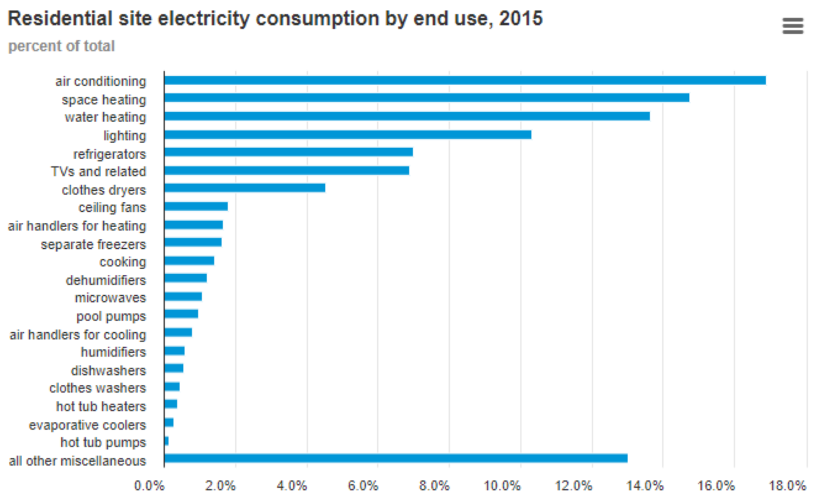


Figure 3: Residential energy usage

Being aware of the main contribution given by the air conditioning and space heating it seems reasonable to focus the optimization process of energy consumption inside residential houses on these two factors, but many additions can be made in order to take into account the others.
According with the review made by Amir Mosavi [2] machine learning has recently contribute very well in the advancement of the prediction models used for energy consumption. This kind of forecasting has been becoming crucial after the energy crisis in 1973 and it is even more important in these days because of the environmental impact plays an important role, too.
Given a sufficient amount of data about temperatures, contexts and energy consumed it could be possible to use this information to predict the incoming state and the actions required to move to the desired one.

1.5 The Smart Grid

Another factor related with the smart home area concerns the power distribution which is the final stage of in the delivery of electric power from the transmission system to individual customers.
Our current electric grid was conceived more than one hundred years ago when the electricity needs were simple. The power generation was localized and built around communities and at that time there were small energy demands such as few light bulbs and a radio. The grid was designed for utilities to deliver electricity to consumers homes in a unidirectional way; this limited one-way interaction makes it difficult for the grid to respond to the ever-changing and rising demands of the 21st century.
The smart grid introduces a two-way dialog where electricity and information can be exchanged between the utility and its customers. It is a set of smart systems, communication controllers and computers automation working together to make the grid more efficient, more reliable, more secure and greener.
This smart grid enables newer technologies to be integrated such as wind and solar energy production and plug-in electric vehicle charging. Most likely with the participation of informed consumers the smart grid will replace the aging infrastructure of today's grid and utilities can better communicate with the consumers to help manage our electricity needs.
A smart home can communicate with the grid enabling consumers to manage their electricity usage by measuring home's electricity consumption more frequently through a smart meter. Utilities can provide their customer much better information to manage their electricity bills.
Renewable resources such as wind and solar are sustainable and growing source for electric power; however these are variable by nature and add complexity to normal grid operations. The smart grid provides the data and automation needed to enable solar panels and wind farms to put energy onto the grid and optimize its use. To keep up with constantly changing energy demands utilities must turn power on and off depending on the amount of power needed at a certain times of the day.
The cost to delivery power depends on the time of day it is used, electricity is more costly to deliver at peak times because additional and often less efficient power plants must be run to meet the higher demand. The smart grid will enable utilities to manage and moderate electricity usage with the cooperation of their customer especially during peak demand times, as a result utilities will be able to reduce their operating costs.
By deferring electricity usage away from peak hours and having appliances and devices run at other times, electricity production is even more distributed throughout the day.
Smart grid technologies provide detailed information about the NAN that enables grid operators to check and manage electricity consumption in real time. This greater insight and control reduces outages and lowers the need for peak power avoiding this way to fire up costly secondary power plans.

The distribution system routes power from the utilities to residential and commercial customers through power lines switches and transformers. While on the actual grid utilities rely on complex power distribution schemes and manual switching to keep power flowing to their customers, the smart grid distribution intelligence counters the unpredictable energy fluctuations due to break in the system caused by storm and bad weather by automatically identifying problems in rerouting and restoring power delivery. Utilities can further use distribution intelligence to predict and manage electricity usage with the collaboration of their customers leading to lower production cost.
In addition increasing the number of plug-in electric vehicles to the grid we have the potential to reduce fuel costs, lower our dependency from foreign oil and help reduce greenhouse gas emissions.
We can then say that smart grid is then another very promising aspect of today's smart changes related with IoT.

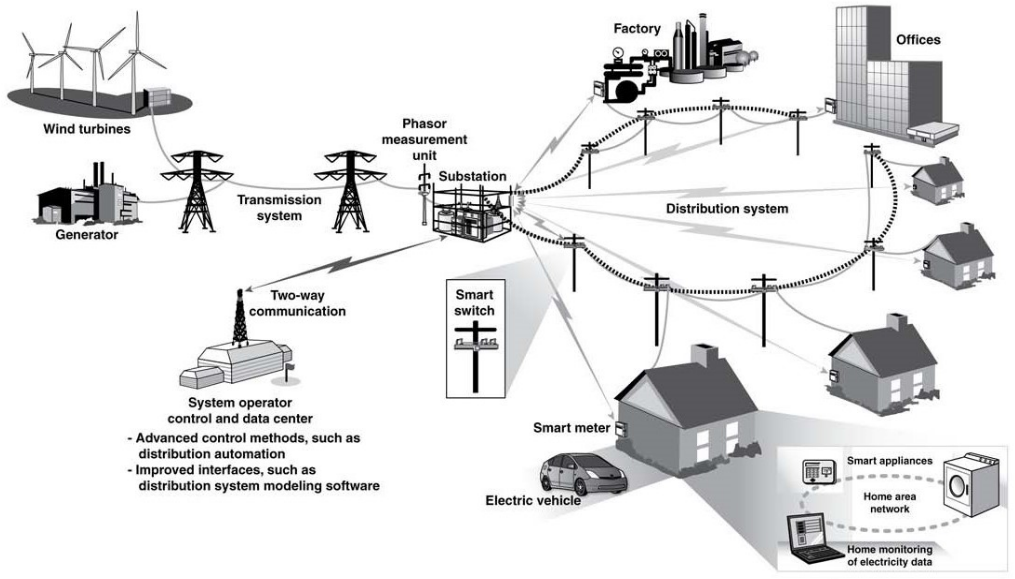

Wind turbines
Generator
Transmission system
Phasor measurement unit
Substation
Factory
Offices
Distribution system
Two-way communication
Smart switch
System operator control and data center
- Advanced control methods, such as distribution automation
- Improved interfaces, such as distribution system modeling software
Smart meter
Electric vehicle
Smart appliances
Home area network
Home monitoring of electricity data

Figure 4: Smart grid schema

1.6 A formal representation of the goal

We define:

- T the whole simulation period time.
- ti is a time in T
$$ti \in T \tag{1}$$
- Ns as the number of sensors
- Na as the number of actuators
- C as the context of the scenario we are simulating, actually it includes the outside temperate and humidity.
- A as the vector of action chosen by the planner in a time $t1$
$$A(t1) = [a0, a1, ..., an]; i \in [0, Na] \tag{2}$$
- S as the state of the house, including the temperature, humidity and the actuators state in a time $t1$
$$S(t1) = [s0, s1, ..., sn]; i \in [0, Ns * 2 + Na] \tag{3}$$
- $f(X)$ is the function able to predict $S'(t2)$ by $S(t1)$ and $C(t1)$ where $t1 > t2$. The following (4) is a formal mathematical representation of f
$$f(S(t1), C(t1)) = S'(t2) \tag{4}$$
- Lp is the loss between the predicted state and the real one in ti.
$$Lp(ti) = |S'(ti) - S(ti)| \tag{5}$$

Because of $\sum_{i=1}^{tn} Lp(ti)$ is prone to be greater than 0 we will have a loss equal to
$$L(ti) = |St(ti) - S'(ti)| > 0 \tag{6}$$
everytime we try to select the best $A(t1)$ action vector to reach $St(ti)$.

So our goal is to find the best f(X) that minimizes the $Lp = \sum_{i=1}^{tn} Lp(ti)$ loss and consequently the $L = \sum_{i=1}^{tn} L(ti)$ loss.

In addition we want to minimize the *Lcost* function which represents the energy consumption due to the actuators working time.

1.7 Different types of data sources

To predict new states of the house as consequences of planned actions a huge amount of records is needed.
Three ways to collect information are possible:

- From a real size home.
- From a virtual environment.
- From a miniaturized version of a real home.

While the last has to face many problems due to the simulation aspect, which will be discussed in the next chapters, the other two are going to be unfeasible for our purposes. At a first glance, a real size home looks a better solution because it seems really easy to map the simulation results into the real world.
The work done by A. Anvari Moghaddam [3] and his team based their results on this kind of approach and it has been proved without any doubt the accuracy of the outcomes achieved. Despite this, if we want to provide to the planner a sufficient amount of data we would have to install sensors and actuators all around the countries where we want to use it. This happens because it is not possible to use the data collected, for instance, from a small cottage in North Dakota to predict the actions the planner has to perform in a tent placed in the middle of the Sahara desert. These two environments would be too different from each other and the result will be neither satisfactory or useful.
With respect to the virtual environment simulation the researches made by Joonseok Park[4] and Kyungmee Lee[5] have shown how even artificial data can be useful to the energy saving programming model, but the fact that data is not collected by real sensors in a real world scenarios still remains. Furthermore, the veracity of the results is hard to demonstrate because many different simulations have to be run and compared with real world scenarios in order to prove the accuracy of the model. In addition, unpredictable factors can effect the consequences of an action while in a virtual environment the entropy of the system is limited by the set of circumstances taken into account during the simulation design phase.
The same virtual approach has been adopted during the simulation performed by Lee which consists of a virtual smart home and the sensors within the house created in Unity. This kind of virtual simulation approach is indeed the cheapest alternative for researchers to simulate a configurable smart home environment which enables autonomous agent generation.
On the contrary a scaled approach is widely used in many different areas where the consequences of performable actions and surrounding environments are challenging to predict a priori and expensive to test empirically. For example, a scaled version of the Mississippi river has been built about an hour west of Boston to rebuild Louisiana's vanishing coast, which is rapidly being lost to rising seas and sinking land. The team leaded by Dan Gessler [6], of Alden Research Laboratory, is now using this $4 million dollar model to guarantee flood protection while preserving the health of the wetlands. In fact, the today hydraulic architecture is not taking care of the river since it has as its main and only goal to prevent the urbanized area from being surrounded by floods. The physical scaled model is going to be used to study the behavior of hydraulic structures and to find alternative and greener solutions.

With reference to our specific scenario a physical scaled version of a suburban home provides real data and it will be easier to set up compared with several real houses located all around the world. Also, it would be possible to use the same house for different scenario because the environment can be changed with only a little effort. Despite the fact that the simulation will be easier and a larger amount of data can be collected, it will be more challenging to use the results in a real home environment and many considerations should be taken into account in order to reach reliable results.

1.8 Related work

The work done by Prashant Shenoy and the members of the Laboratory for Advanced Software Systems [7] has led to a set of heavily instrumented real-size smart homes used to collect data and to write a large amount of high quality papers. They are able to provide a wide amount of test-beds useful to experiment new techniques and algorithms to improve the smart home efficiency. A part of the data collected is available on the web page of the project and it has been very useful to introduce the optimization matter and its related problems since it shares with ScaledHome the same goals and the same procedure even if the data set collected comes from real houses instead of scaled ones.
It has also necessary to mention the work done by Lin and his team [8]. They have examined the relationship between the behavior of in-home inhabitants and indoor air quality based on the data collected from several smart home sensors and chemical indoor air quality measurements. This has been done by gathering data from two smart homes and analyzing how indoor air quality affected the human behaviour inside the house, as well as the relationship between different groups of smart home features and indoor air quality variables. To analyze this data they used different machine learning models such as random forest, linear regression, and support vector regression. The conclusion can be summarized as follows: there is a strong relationship between in-home human behavior and air quality and this observation can be generalized across multiple smart homes. The temperature was found to be the most relevant selected feature among human activities. The author believed that the temperature changes within the homes were caused by multiple human activities, making it the most impactful feature within the data-set.
Instead of proving the importance of air quality the work done by Jin et al. [9] has produced as final result a prediction model able to optimize the power consumption for heaters within a smart home environment. The model has been achieved by implementing a recurrent neural network (RNN) and a long short memory model (LSTM) which utilized real data about temperature and humidity collected by a real-size physical home. The final outcome is an optimization scheme that saves energy while providing a comfortable environment in terms of user-desired temperature and humidity.
The study conducted by Mateo [10] and his team has used different kind of regression models in order to predict the temperature with a average error of about 0.1°C. It has to be pointed out that the model has been applied only to large buildings and because of that target the results cannot be applied in a smart home system.
Weatherman is the model presented by Chen and Irwin [11] that analyzes energy consumption data as well as wind and solar generation data to predict where on earth a set of coarse energy consumption data has occurred. The main idea behind Weatherman is that a set of weather signatures appear in a similar way in different environments around the world. The result of the research can be applied in the construction of energy consumption aware environments around the world.
The prototype presented by Teich and his team [12] is a neural network aimed to be

implemented inside smart homes in order to maintain a cozy temperature defined by the user in an energy-efficient way. The dynamicity of the model allows it to re-train itself based on new activities inside the home.
The work done by Kim [13] has been about a different probability-based algorithm to identify human activities inside the house. The target application scenarios are human-centered such as healthcare and education. The research focused on the recognition of multiple patterns and the ambiguity of different actions.
As well as the algorithm developed by Kim, the smart home created by Cook et al. [14] has as main goal to accurately predict inhabitant action. Their neural network algorithms predictions facilitated an adaptive and automated environment meant to improve the experience of its inhabitants.
Fritz and Cook [15] focused their work on elderly healthcare. They discussed the application of health-assistive smart homes by using data collected from sensors in elderly patients' homes, monitoring it with intelligent algorithms, and predicting potential changes in the health status of the patients.
The smart home health-based model proposed by Alberdi et al. [16] is about healthcare too. Using data collected in more than two years they developed two models. The first is able to detect changes in mobility skill while the second is related with the detection of changes about memory skills. Both of them act as an early warning system meant to prevent potential dangerous symptoms to be harmful.
A completely different purpose leaded Dahmen and his team [17] to focus on the security aspect in smart homes. The model implemented has the goal to detect anomalies within the home behaviours in order to automatically detect threads and identify proper actions to be taken in order to guarantee the security of the environment.

1.9 The reason behind ScaledHome

ScaledHome has been built to accomplish the goal of collecting a large amount of data in order to set up a predictive model able to support the air conditioning system.
We used the term support with respect to AC system because our purpose has not been to take over the AC system. In fact, there are many different scenario where the importance of the role played by AC cannot be neglected. For example, when we want to reach a state which is very different than the current state no energy-saving actions inside the house can be performed to achieve it and the only solution still remain to turn on the AC system. Nonetheless, there are many other scenarios where our target state is close to the one we want to achieve. In those cases the heterogeneous state of the rooms inside the house can be balanced in order to reach the target state saving as much energy as possible.
For instance, consider a simplified situation where we have just two rooms with different temperatures (Ta and Tb) and we are interested in reaching a target temperature T. If T is in the [Ta, Tb] range it is possible to decrease the temperature of the warmest and increasing the one of the coldest just allowing them to exchange thermal energy.
With this consideration in mind a planner able to take autonomous choices in order to accomplish the target state requirements with the minimum energy consumption as possible would be a great improvement in the smart house area. For this reason a large amount of data is needed in order to build the predictive model used by the planner.
It is also necessary to point out the nature of the data collected, they are in fact gathered by empirical experiments run in the real world in order to guarantee the maximum correspondence between actual scenarios and the simulation ones. The similitude is reliable because in contrast with a virtual simulation the actions performed and their consequences are true actions and effects on the state of the house.

1.10 Interpreting ScaledHome experiments: the scaling problem

Some considerations about the scaling approach adopted have to be taken before moving on. Because of every simulation runs in a scaled environment the results of this work has to be mapped into the real world.
This mapping has to consider not only that the time speed inside the house is different, but also that every length change leads to a spatial dimension raised to the third ratio change since this is due to the relation between length and a physical space.
It is not hard to understand how this kind of similitude could be very difficult to achieve for a human being and for a standard artificial planner, too.
That is an additional reason why several machine learning techniques have been employed in order to reach our goal.
The mapping between scaled models and real world models is not novel, it has already been faced in many engineering areas and all of them found their base on the Buckingham p-theorem which formalizes how a n dimensional problem can be described as the combination of $p = n - k$ dimensionless parameters where k is the number of physical dimension.
This kind of considerations have to be taken into account in order to apply the results to a real world scenario.

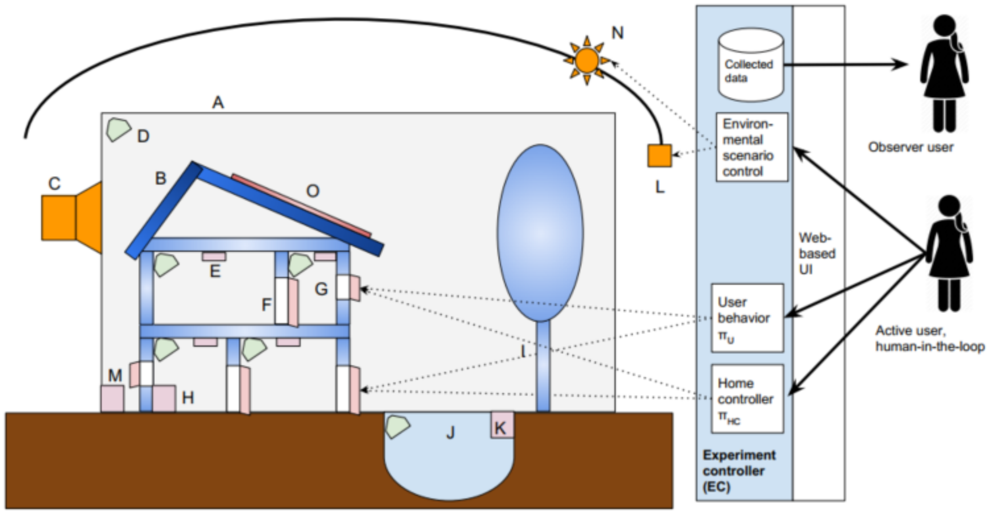

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
Collected data
Environ-mental scenario control
Web-based UI
User behavior π_U
Home controller π_{HC}
Experiment controller (EC)
Observer user
Active user, human-in-the-loop

Figure 5: ScaledHome simulation schema

1.11 ScaledHome: a physical scale model of a suburban home

ScaledHome is composed of 6 rooms and each of them has at least a temperature and humidity sensor, a door and a window. The model has eight windows and seven doors, two of which are entrances to the house. Each door and window is connected to a motor to open and close it, this is done in order to both simulate a real human being moving inside the house and to provide to the planner an alternative way to change temperature and moisture inside each room.

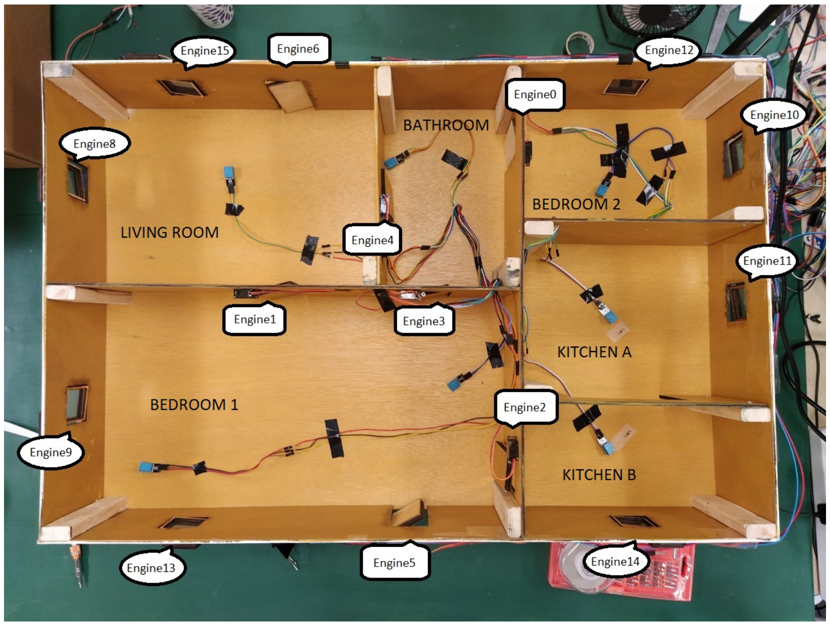


Figure 6: Rooms and motors

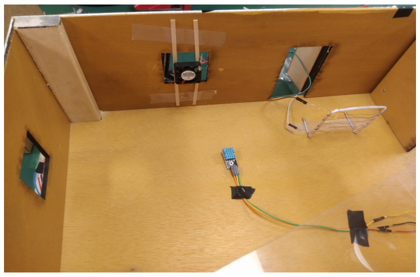


Figure 7: AC and heater

A small fan and an heater have been installed inside the house to reproduce the AC cooling and heating systems, these two devices were needed to change the inside temperature and moisture independently from the outside environment. In fact, the goal of this project has been to use these two devices as less as possible and instead of consuming energy accomplish the desired requirements by opening and closing doors and windows. In order to simulate the outside environment there are also a fan and a lamp representing the wind trend and the sun respectively.

2 Software architecture principles

Before stepping into the real implementation which has led the project to the results we were interested in; it is better to introduce the principles that have been taken into account during the development of the product in terms of software architecture.
When we talk about software architecture we mean a set of fundamental a priori choices made before implementation than will be hard to change once performed.
A good software architecture should be resilient to extreme cases even when they are not expected. In order to do that several considerations have to be taken into account, such as:

- Performance: a good set of choices will lead to a fast and performing application able to handle multiple request or multiple action all together.
- Scalability: an entity is well scalable if it can grow and manage increased demand over the time. In order to do that the system has to be able to require and obtain new resources if the number of requests increases beyond a well defined threshold, this can be set in terms of resources consumption or in terms of latency.
- Reliability: this property is related with the probability of failure-free software operation for a specified period of time in a specified environment, it should be as great as possible because we want make services and resources available most of the times.
- Flexibility: a flexible architecture can handle many changes with less effort as possible. New features and new entities have to be introduced modifying the original structure as less as possible.
- Security: all the application which want to provide services for entities able to consume them have to be aware that not each of them will have an honest purpose. For that reason safety precautions have to be taken into account, even knowing that this will always effect badly the performance.
- Availability: it is the characteristic which represents how easy and how often the services provided by the architecture are accessible by interested entities.

A good architecture can still be changed if the requirements will change, but an high frequency change of the architecture is often a symptom of bad choices made in the beginning of the project. For that reason every time a new system has to be developed it is a good habit to spend some time analyzing the requirements and designing the architecture before starting to implement the code. Also it is better to test each component of the application by itself before integrate it with the whole system and then test all the components together.
The following diagram shows which is a good practise to take when we are facing a complex design architecture.

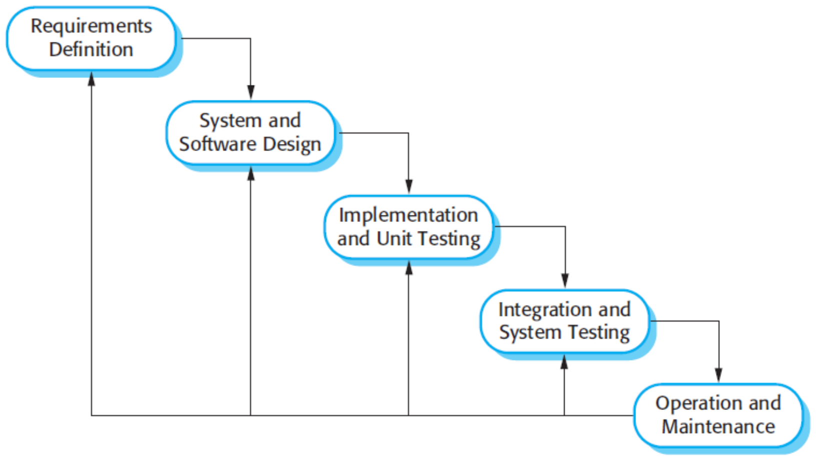

Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

Figure 8: Rooms and motors

As we can see after the requirements definition and the system and software design we find the implementation step alongside testing. That consideration leads us to the next main concept.

2.1 SOA paradigm

The term SOA stands for Service Oriented Architecture and according to the SOA Manifesto it ”is a paradigm that frames what you do. Service-oriented architecture (SOA) is a type of architecture that results from applying service orientation.” [18].
A Service Oriented Architecture leads the system to share as many components and services as possible. There are two major roles within Service-oriented Architecture:

- Service provider: the service provider is the maintainer of the service and the organization that makes available one or more services for others to use. To advertise services, the provider can publish them in a registry, together with a service contract that specifies the nature of the service, how to use it, the requirements for the service, and the fees charged.
- Service consumer: The service consumer can locate the service metadata in the registry and develop the required client components to bind and use the service.

Basically, a system developed according with the SOA paradigm has the main aim of splitting the code into services. Each module has its own service to provide, but it can interact with the others when multiple services have to work together in order to satisfy some common requirements or when a service is needed to make available an other service.
SOA leads to a more flexible architecture and it is commonly considered a good practise in every complex scenario where the system has to change frequently every time a new module has to be integrated inside the whole application.
The following picture illustrates the main difference between a monolithic and SOA architectures.

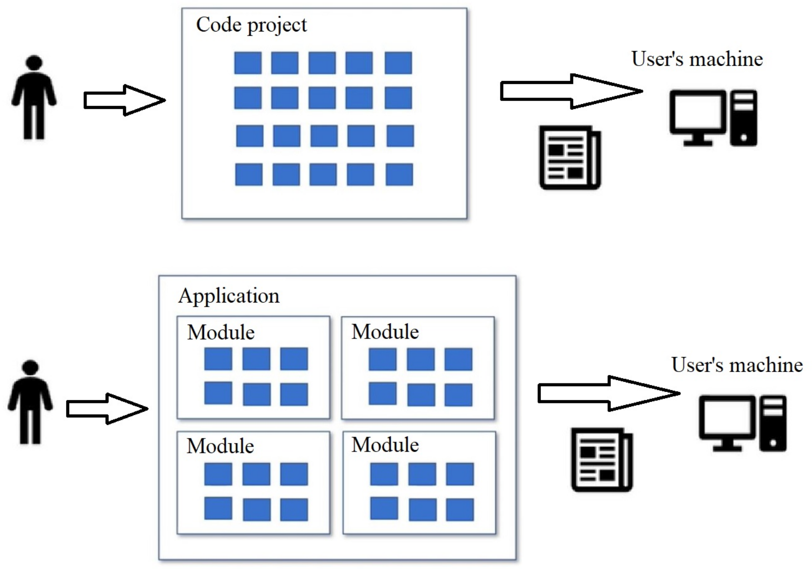


Figure 9: Monolithic and SOA architectures comparison

Once a SOA architecture has been designed it can be deployed as a single application bundling all the modules together or it can be distributed among different nodes which communicate each other in a cluster or cloud architecture.

2.2 Continuous Integration (CI) and Continuous Deployment (CD)

First we have to define these two terms:

- Basically, CI means merging different developing branches into a common main flow. This became even more important when teams with different scopes and skills have to work on the same project such as ScaledHome where I had to work with a mainly computer science team which has the primary background in the data mining and machine learning while my background was main devoted to the distributed system area. CI is often used in combination with test-driven deployment which is done by running and passing all unit tests in the developer's local environment before committing to the mainline. In order to do that many automated test tools are widely used and available to the developers communities depending on their different languages and technologies.

- CD is often confused with Continuous delivery and despite of they share some common denominators they are two different concepts. Both of them have the aim of being able to ship a product in any moment, but the academic literature agrees that while Continuous deployment involves a manual activity, Continuous development uses an automated deployment approach.

2.3 DevOps

DevOps combine these good practices into a broader scope whose purpose it to shorten the system deployment life cycle while keeping an high quality software delivery.

2.3.1 Docker

Docker is now a well known technology, by know it can be referred as a standard even if it is a set of PaaS product. It uses OS-virtualization to run custom applications in a self contained environment called container. That means that an application deployed into a docker container can be shipped and moved across different platform independently of the underling hardware infrastructure. A custom application lies on top of different layers, basically the more complex is the application the more layers will have to be stack. An image is a file containing the instructions needed to run one or more docker instance. Each layer decorate the previous one adding more libraries and frameworks used by the custom application plugged on

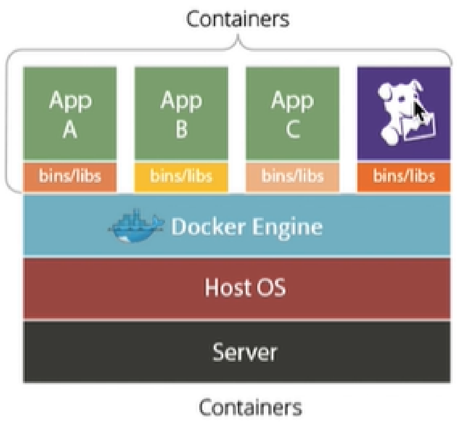


Figure 10: Docker stack diagram

top of the stack.
Compared with a classical Virtual Machine which wrap the entire operating system we are going to use, a Docker container shares the host OS kernel with the other containers on the same physical machine and, usually, the binaries and libraries, too. Because of their lightness due to the fact that they do not have a own OS containers can be migrated and replicated in a easier way than VMs. Load balancing policies and changes are made easier too and this is the reason why when we talk about DevOps we often refer to Docker as the main solution. Thanks to these advantages CI and CD are widely used in containerized environments. For example if we need to add a new feature to our dockerized application while keeping the service available to an imaginary customer we can create a replica of the application using the old images and in the meantime develop and test the new feature in a new isolated environment creating a different image. Only after a reliable validation the new version can be released replacing the previous one. Dynamic load balancing can be easier for the same reasons, too.
Docker became even more powerful when we are facing cloud or cluster type scenarios because many containers can be spun on the same machine with a minimal effort. In fact when an image has been built it can be stored in an online repository accessible from any kind of platform as shown in the following image.

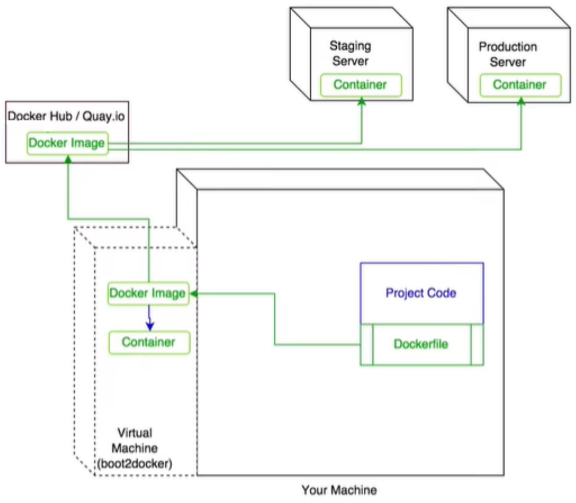


Figure 11: Docker production environment

In a cluster or cloud environment different containers can interact with each other and there is where the DevOps concept finds its best realization. Usually docker instances communicate throw REST API in order to use specific services. Thanks to that kind of interaction different teams can develop in parallel specific and heterogeneous features and services with just an agreement on which are the API endpoints used to access a service provided by and other containers.

2.3.2 Kubernetes

Kubernetes is an orchestrator for containers instances. It is used to automate containers instantiation and behaviour within the same cluster. Also, Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and does not advertise them to clients until they are ready to serve.
Kubernetes's core is the Cluster services, it receives a desired state via API and it takes care of the actions needed to move and to maintain it. The configuration file can be a yaml file or a json file and once it has been fed to the cluster services core it will run the desired images on container hosts called workers.
The atomic configuration entites in Kubernetes are called Pods. A pod is a group of one or more containers which share the same storage, network and specifications for how to run the containers. Most of the times a pod and a microservice are in a one to one relationship even if when two microservices are very coupled, for example if they share the same persistent storage, it can make sense to place them in the same pod.

2.3.3 Micro-services architecture

We have already discussed about SOA architecture and its main advantages and before the advent of distributed system architecture it was the best solution without any doubt. But with the diffusion of internet more and more applications start moving the computational overhead towards back-end services making them available over HTTP for the users. This has eased the update of software versions because new releases can be deployed on the same server the users where accessing before the release. This solution keeps an high QoS because flexible providers are able to scale up the number server instances when the demand increases and vice versa.
The problem arises when for example just one of the services is overwhelmed by a traffic peak, in this case the whole application has to scale up even if the other services are working without additional effort. The approach suggested by Docker and Kubernetes in the previous section is called Micro-services architecture and after what we have said about CI and CD it should be easy to understand why Micro-services are so widespread used. Disparate services can be developed, tested and delivered while guaranteeing scalability, reliability, flexibility and availability. Instead of keeping the entire application bundled as a single entity each service can be containerized keeping just the few resources and code needed to serve a specific task. When a container fails or when the traffic arises Kubernetes can automatically instantiate new replicas of the same image while keeping still the number of the other containers.
Despite the Micro-services community tends to stress the difference between SOA architecture and Micro-services it seems reasonable to say that they have the same purpose, but they differ in the context where they can be applied. The former has been used mainly on single machine architectures while the latter is now wide spreading because in many cases we face distributed and cloud architectures.
In addition, we have to say that moving a SOA architecture to a micro-services system is not so complicated. Thanks to the SOA paradigm the different functionalities inside the application are already well split and thanks to Docker, Kubernetes and CI/CD tools the greatest part of the deployment is heavily eased by external support.
In order to achieve such architecture functionalities have to be split and organized wisely delegating as less duties as possible to each of the modules comping the application. These well organized pieces of code are so called micro-services because they are small

and they provide few specif services. On the other hand the interaction between them can be tricky and more and more challenging while the size of the project increases. For this reason an entity able to manage their interaction grant an higher level interface is needed.

2.4 Middleware

In a loose sense, a middleware in the software engineering area can be described as the "glue" which keeps together different and modular part of the same system. It is the infrastructure with coordination duties which overcomes the problems due to the heterogeneity nature of many ad-hoc scenarios.
Basically a middleware acts as support base to higher level applications.
The following can be a more formal definition.

2.4.1 Definition

The definition of middleware was born in 1968 in the famous NATO school of Software Engineering, but it become more significant after the 90' because the the widespread diffusion of distributes system design practices. Nowadays, according with Etzkorn, a middleware can be defined as a "software that connects computers and devices to other applications." Another way to define middleware is to say that "it acts as an intermediary. It is often used to support complicated and distributed applications. It can be a web server, application server, content management system, or other tool that supports application development and delivery. It can also be a software application that connects two or more applications so that data can be shared between them." [19].

2.4.2 Advantages

A middleware approach leads to the following advantages:

- Transparency: a middleware provides to the application level an higher level of abstraction to interact with hiding the heterogeneity due to several subsystems that have to share data and access or provide services even if they are completely different. This transparency enable the developer to spend more time and energies to improve the business logic without taking into account how the underlying assets interact each other, and the same concept eases the life of a new person that has to interact with the whole system.

- Flexibility: because of the middleware has to respect the "loose-coupling" principle it makes changes of existing entities or introduction of new ones easier.

- Grant necessary availability and QoS: a middleware has also to deal with resources discovery and QoS.

2.4.3 Disadvantages

In terms of main disadvantages we have to say that the introduction of a new entity make our system more complex and it force the addition of a new interaction each time two entities have to communicate.
In addition without any replication techniques the middleware service represents a single point failure since the communication flow is force to pass though this entity.

2.5 Entities interaction patterns

Many ways are possible to make different entities interact each other. Because of the set of requirements for this project includes interaction in a high heterogeneous environment we are going to mention only those pattern suitable to this requirement.

2.5.1 REST

REST stands for ”Representational state transfer”. RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations. Compared with SOAP which is a standardized protocol such as HTTP and SMTP that describes how programs built on different platforms and programming languages could exchange data in an easy manner, REST is an architectural style in which a web service can only be treated as a RESTful service if it follows the constraints of being:

- Client-Server
- Stateless
- Cache-able
- Layered System
- Uniform Interface

In terms of bandwidth usage SOAP is a more complicated service oriented protocol. Since SOAP Messages contain a lot of information inside of it, the amount of data transferred using SOAP is generally a lot.

```xml
<?xml version="1.0"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
    SOAP-ENV:encodingStyle=" http://www.w3.org/2001/12/soap-encoding">
    <soap:Body>
        <Demo.guru99WebService xmlns="http://tempuri.org/">
            <EmployeeID>int</EmployeeID>
       </Demo.guru99WebService>
    </soap:Body>
</SOAP-ENV:Envelope>
```

Listing 1: Soap message as XML file

While REST does not need much bandwidth when requests are sent to the server. REST messages mostly just consist of JSON messages. Below is an example of a JSON message passed to a web server. You can see that the size of the message is comparatively smaller to SOAP.

```json
{
    "city":"Mumbai",
    "state":"Maharastra"
}
```

Listing 2: REST message as JSON file

2.5.2 Publish Subscribe (MQTT)

Publish/subscribe messaging, or pub/sub messaging, is a form of asynchronous service-to-service communication used in serverless and microservices architectures. In a pub/sub model, any message published to a topic is immediately received by all of the subscribers to the topic. Pub/sub messaging can be used to enable event-driven architectures, or to decouple applications in order to increase performance, reliability and scalability.
Pub/Sub interaction is compliant with the "Loose coupling principle" which states that entities in the same system have to be as much decoupled as possible, in fact in a pulishers and subscribers are not only decoupled in time, but in space too.
They do not need to know the location of each other because the broker takes care of all these aspects.

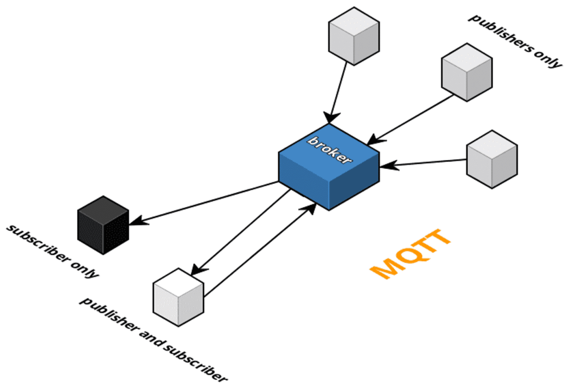


Figure 12: MQTT mono and bidirectional communication

MQTT is an open OASIS and ISO standard (ISO/IEC PRF 20922) and it was invented by Dr Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech), in 1999. It stands for MQ Telemetry Transport. According with the mqtt.org website MQTT "is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging "achine-to-machine (M2M) or Internet of Things world of connected devices, and for mobile applications where bandwidth and battery power are at a premium."[20].
Because of its lightness MQTT represents a gold standard in the M2M communication and it is even more suitable in cases where devices with limited hardware resources have to interact each other. Its usage in this project is due to the fact that it is a well known standard and because many Open SaaS are available online.
In this project I have used "cloudmqtt" as SaaS because it offers a free tier with up to 5 clients and 24/7 availability [21]. It is quite easy to configure and it offers interfaces to monitor the state of the connections and to set ids and passwords for each authorised client.
A single topic for the three clients composing ScaledHome (Home-Controller, Actuators-Controller and Sensors-Controller) has been deployed.

2.5.3 MQTT QoS

It is important to underline that it is also possible to set a desired QoS. The possible options are:

- At most once (QoS 0): this service level guarantees a best-effort delivery. There is no guarantee of delivery. The recipient does not acknowledge receipt of the message and the message is not stored and re-transmitted by the sender. QoS level 0 is often called “fire and forget” and provides the same guarantee as the underlying TCP protocol.

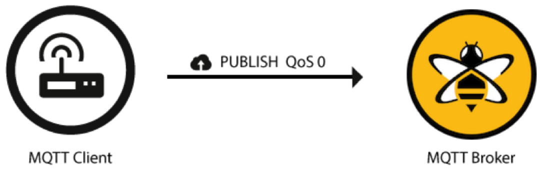


Figure 13: Pub/Sub interaction diagram

- At least once (QoS 1): QoS level 1 guarantees that a message is delivered at least one time to the receiver. The sender stores the message until it gets a PUBACK packet from the receiver that acknowledges receipt of the message. It is possible for a message to be sent or delivered multiple times.

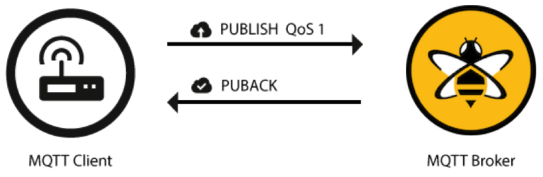


Figure 14: Pub/Sub interaction diagram

- Exactly once (QoS 2): QoS 2 is the highest level of service in MQTT. This level guarantees that each message is received only once by the desired recipients. With QoS 2 the service will be slow and safe. The guarantee is provided by at least two request/response flows (a four-part handshake) between the sender and the receiver. The sender and receiver use the packet identifier of the original PUBLISH message to coordinate delivery of the message.
  When a receiver gets a QoS 2 PUBLISH packet from a sender, it processes the publish message accordingly and replies to the sender with a PUBREC packet that acknowledges the PUBLISH packet. If the sender does not get a PUBREC packet from the receiver, it sends the PUBLISH packet again with a duplicate (DUP) flag until it receives an acknowledgement. Once the sender receives a PUBREC packet from the receiver, the sender can safely discard the initial PUBLISH packet. The sender stores the PUBREC packet from the receiver and responds with a PUBREL packet. After the receiver gets the PUBREL packet, it can discard all stored states and answer with a PUBCOMP packet (the same is true when the sender receives the PUBCOMP). Until the receiver completes processing and sends the PUBCOMP packet back to the sender, the receiver stores a reference to the packet identifier of the original PUBLISH packet. This step is important to avoid processing the message a second time. After the sender receives the PUBCOMP packet, the packet identifier of the published message becomes available for reuse.

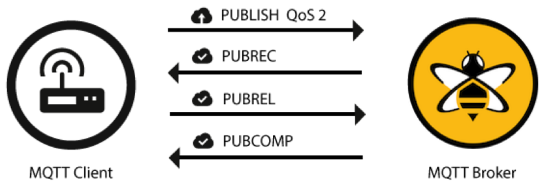


Figure 15: Pub/Sub interaction diagram

In our case we have used QoS 2 which means that messages can arrive duplicated, but they will arrive for sure. In fact, we are not interested in receiving each message exactly once because any kind of filtering can be done on the Home-Controller in case messages arrive duplicated, while a QoS 1 would not be reliable enough because we have to be sure that every message arrive to the intended receiver in order to collect data or perform actions while maintain the synchronization with the model stored inside the middleware.

2.6 MVC pattern

MVC stands for Model-View-Controller and it is a design pattern commonly used for developing user interfaces.
According with the Design principles book: "MVC decouples views and models by establishing a subscribe/notify protocol between them. A view must ensure that its appearance reflects the state of the model. Whenever the model's data changes, the model notifies views that depend on it. In response, each view gets an opportunity to update itself. This approach lets you attach multiple views to a model to provide different presentations." [22].
It splits the system into three different entities:

- MODEL: this entity directly manages the date, keep track of the current state and handles sessions.
- VIEW: this is what is going to be presented to the user, it has to be user-friendly even to those who are not very familiar with the specific user-case the application is facing.
- CONTROLLER: it accepts inputs from the user and performs actions in order to modify the state inside the model or to retrieve data from it.

It eases the development of applications that have to interact with the user by mapping different services into different entities. Each of the different parts composing the application can be developed by different person simultaneously because they are decoupled from each other with reference to the "loose coupling principle".

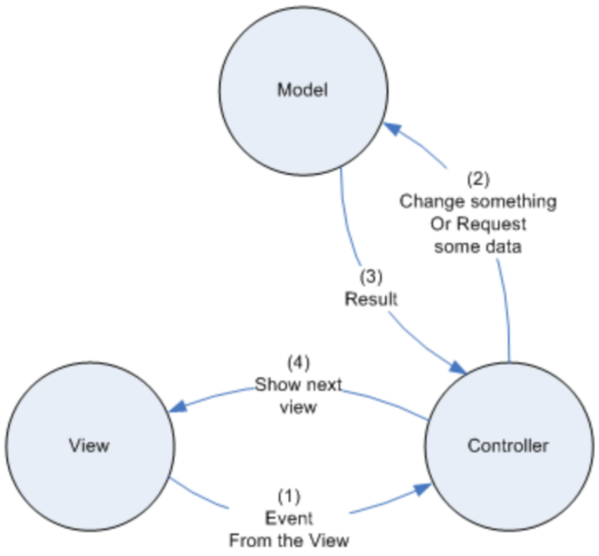


Figure 16: MVC pattern diagram

2.7 Relation and Non-Relational Databases

Databases are used to store persistent data which can be gathered by interested applications later.
Relational databases represent and store data in tables and rows. They're based on a branch of algebraic set theory known as relational algebra by using Structured Querying Language (SQL) which makes them a good choice for applications that involve the management of several transactions. The structure of a relational database allows to link information from different tables through the use of foreign keys (or indexes), which are used to uniquely identify any atomic piece of data within that table. Other tables may refer to that foreign key, so as to create a link between their data pieces and the piece pointed to by the foreign key. This comes in handy for applications that are heavy into data analysis. Relational algebra states that the relationship between tables must remain consistent as a consequence of the normalization which is the practise consisting in referencing existing elements by other tables. For many years relational databases have been the most common type of databases; MySQL, PostgreSQL and SQLite3 are just some of the widespread implementations.
Nowadays the requirements may have changed; in particular when we have to face specific scenarios such as IoT or real-time application the solid structure offered by relational databases is often not suitable. In fact many savings of unstructured data have to be handled concurrently and simply they do not fit the tables defined a priori.
In that situation it is may needed to consider going with a non-relational database. A non-relational database just stores data without explicit and structured mechanisms to link data from different tables (or buckets) to one another. This way they overcome the limits of relational databases storing data in more flexible formats. Instead of storing data into rows and columns the atomic data is stored into a document; which is compliant with JSON format standard. In the same way rows and columns are organized into tables, sets of documents can be grouped into collections. Documents in the same collection do not have to share a common structure and for this reason they are more suitable to store unstructured data which would be hard to process and normalize real-time. One of the most popular non-relational databases is MongoDB, it offers also standardized connectors for Javascript frameworks like NodeJs. Since Mongo doesn't automatically treat operations as transactions the way a relational database does, transaction must be manually created and then manually verified, the same happens in case of commit or roll back. This is of course the main disadvantage, but we have also to consider that since there are no joins like there would be in relational databases, multiple queries have to be handled manually in the code. This main disadvantage can be partially solved avoid any kind of reference distributing multiple replicas of the same record among the documents instead. This is only a partial solution that can save time when a new document is inserted while updates lead to the distributed caching invalidation problem which occurs when a single replica is modified and the update has to be propagated to the others.

3 Data mining and Machine Learning principles

3.1 Data preprocessing in Data Mining

Vast amounts of data are around us in the world; it has always been this way, but thanks to the continual improvement of digital technologies we can now gather them automatically saving effort. In fact, most of the times the hardest part of the Data Mining process is not to collect the data, but it is to properly process them instead.
Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues.
The importance of the preprocessing phase comes from the need of handling reliable data instead of sparse and noisy data.
When we have to deal with real world data we have to face one or both the following problems:

- Noise: it can be generated by faulty data collection, data entry errors or a wrong data collection procedure. It will affect badly the results in the learning phase.
- Inconsistency: we can define inconsistency as anything that affects the data integrity. In fact especially if you're using the database as a System of Record any duplicated record or unpredicted data type can lead to unexpected results in the learning phase.

3.1.1 Steps Involved in Data Preprocessing:

- Data cleaning: The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing and noisy data.
  - Missing Data: this situation arises when some data is missing in the data. It can be handled in various ways, such as ignoring the tuples or fill the missing values with attribute mean or the most probable value.
  - Noisy Data: noisy data is meaningless and it cannot be interpreted by a machine. To solve it we can use one of the following methods:
    - Binning method: this method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.
    - Regression: here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).
    - Clustering: this approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

- Data tranformation: this step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:
  - Attribute selection: in this strategy, new attributes are constructed from the given set of attributes to help the mining process.
  - Normalization: it is done in order to scale the data values in a specified range. Usually the target range is between zero and one.
  - Standardization: it transforms data to have a mean of zero and a standard deviation of one, it is also called z-score standardization.
  - Discretization: this is done to replace the raw values of numeric attribute by interval levels or conceptual levels.
  - Concept Hierarchy Generation: here attributes are converted from level to higher level in hierarchy. For Example-The attribute "city" can be converted to "country".
- Data Reduction: since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we uses data reduction technique. It aims to increase the storage efficiency and reduce data storage and analysis costs.
  The various steps to data reduction are:
  - Data Cube Aggregation: aggregation operation is applied to data for the construction of the data cube.
  - Attribute Subset Selection: the highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use level of significance and p- value of the attribute.the attribute having p-value greater than significance level can be discarded. This is done because an high p-value is symptom of a great difference of the attribute compared with the null hypothesis.
  - Numerosity Reduction: this enable to store the model of data instead of whole data, for example: Regression Models.
  - Dimensionality Reduction: this reduce the size of data by encoding mechanisms. It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction are called lossless reduction else it is called lossy reduction.

3.1.2 Regression

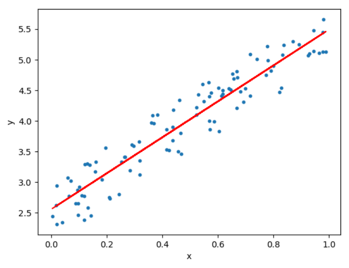


Figure 17: Example of linear regression

Regression is a technique for determining the statistical relationship between two or more variables where a change in a dependent variable is associated with or depends on a change in one or more independent variables. This relationship can be linear or not and it can vary depending of the degree of the achieved approximation. We have to take into account that every regression model is prone to some error which can come from many sources like variables not taken into account or just random variation like outliers. This is a main concern when we have to face a regression problem because outliers are in most cases very far away from the rest of the data and they have an undue influence on the regression function.
Generally speaking the lower is the error the better is the quality of the regression model or in other words the function has to be as close as possible to all the data points at once.
That means that the best regression will be the one that minimize the distances between the predicted data and the actual one. So the function will change depending on which method will be chosen to calculate the distance between pairs of data points.

3.1.3 Interpolation vs Extrapolation

Both interpolation and extrapolation are related with the concept of regression which basically is the process needed to find the best function which fits the data.
The main difference between these two terms is that interpolation consists of predicting new values using the function obtained by regression for values in the domain of the original data used to find the regression function, while on the other hand extrapolation uses the same regression function to predict corresponding values outside its domain. In most case we will end up with better result in case of interpolation just because the new data has a better matching with the given one while when we propagate the knowledge about the data outside its domain by extrapolation unpredicted oscillation trends will be likely found.

3.1.4 Different types of distances

Both Data mining and Machine Learning have to consider a way to evaluate distances between data-set record and between predicted values and actual values respectively. In Machine Leaning models the choice of the distance algorithm can heavily affect the performances of the model. In fact, the best distance techniques depends on each specific case. In the next chapters we will take them into account with reference to a specific ML algorithm called K-neighbours.
The following are just some of the algorithms used to calculate the distance between data and prediction points:

- Euclidian distance: it is the ordinary straight-line distance between two points in Euclidean space. It is the simplest, but it can still perform well in simple regression scenarios.
  Given two data vectors p and q it can be mathematically formalized as follows:

$$d_{euc}(\mathbf{p},\mathbf{q}) = d_{euc}(\mathbf{q},\mathbf{p}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{7}$$

- Manhattan distance: the manhattan distance d_{man} between two vectors p and q in an n-dimensional real vector space with fixed Cartesian coordinate system, is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes. More formally:

$$d_{man}(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=1}^{n}|p_i - q_i| \tag{8}$$

- Mahalanobis distance: it is also called "generalized squared interpoint distance" and it can be defined as a dissimilarity measure between two random vectors $\vec{p}$ and $\vec{q}$ of the same distribution with the covariance matrix S:

$$d_{mah}(\vec{p},\vec{q}) = \sqrt{(\vec{p}-\vec{q})^T S^{-1}(\vec{p}-\vec{q})} \tag{9}$$

3.1.5 From Data Mining to Machine Learning

All the steps we discussed in the previous section belong to the Data Mining area, but it may be reasonable to wonder how these steps are related with Machine Learning.
Both Data Mining and Machine Learning fall under the aegis of Data Science, which makes sense since they both use data. Both processes are used for solving complex problems, so consequently, many people (erroneously) use the two terms interchangeably. This is not so surprising, considering that machine learning is sometimes used as a means of conducting useful data mining as well as the information gathered from data mining can be used to train ML models. For this reason the line between these two concepts become blurred.
Furthermore, both processes employ the same critical algorithms for discovering data patterns. Although their desired results ultimately differ, something which will become clear as you read on.
Despite the common aspects, they have completely different purposes. Data mining is designed to extract the rules from large quantities of data, while machine learning teaches a computer how to learn and comprehend the given parameters. In other words data mining is simply a method of researching to determine a particular outcome based on the total of the gathered data. On the other side of the coin, we have Machine Learning, which trains a system to perform complex tasks and uses harvested data and experience to become smarter.
Machine learning can use Data Mining techniques in order to find which are the most significant and predictable features in data-sets, while on the other hand Data Mining can use Machine Learning to automatically manage data processing on the fly in big data real-time scenarios where not only we have to face huge amount of data, but the acquisition velocity is high, too.
For these reason we can say that Machine Learning and Data Mining are two very bounded topics and each of them can take mutual benefit from the other.

3.2 Artificial intelligence

John McCarthy coined the term Artificial Intelligence and defined it as “the science and engineering of making intelligent machines, especially intelligent computer programs”. In other words AI is an umbrella term which refers to any kind of autonomous decision making performed by an algorithm.
Despite the fact that it cannot still replace a real human being reasoning process it is nowadays a useful tool that can be used as support engine both to human common problem and companies pipelines. In fact, there are many tasks which are easy to perform for a human being but are hard and uncomfortable for an artificial intelligence algorithm. For example the speech comprehension of a human conversation is still hard to understand and interact with for a machine.
Apart from that artificial intelligence algorithms can be applied in many decision scenarios. Every time a choice has to be taken many consequences have to be considered, in fact it is not a coincidence that the first AI application were related to games where the forecasting capability plays a very important role such as chess or chekers. In these kind of scenarios the search space can be enormous for a human being, but it can be faced by a machine equipped with enough computational resources since the number of possibilities is limited and easy to formalize.

3.2.1 Tree search algorithms

A tree graph simulates a hierarchical tree structure with a root value and sub trees of children with a parent node, represented as a set of linked nodes. Each node constitutes the consequence of a choice and each link is the choice taken.
Different kinds of algorithm can be used to traverse the tree and find the best routes to the target node. Each choice is taken based on a function called heuristic which estimates at each branching step the quality of the considered choice with reference to the target. The objective of a heuristic is to produce a solution in a reasonable time frame that is good enough for solving the problem. The main approaches are:

- Breadth-first search (BFS): BFS starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a ”search key”), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.
- Depth-first search (DFS): it starts from a search key node which is usually the root node and it explores as far as possible along each branch before backtracking.
- Hybrid search: it combines the main advantages of the two previous approaches and it can be the best one in many cases where the time available to compute the best choice plays and important role.

3.3 Machine Learning

Nowadays, Machine Learning is one of the most common terms used in the software engineering and computer science area. Nonetheless this term is not always used properly.

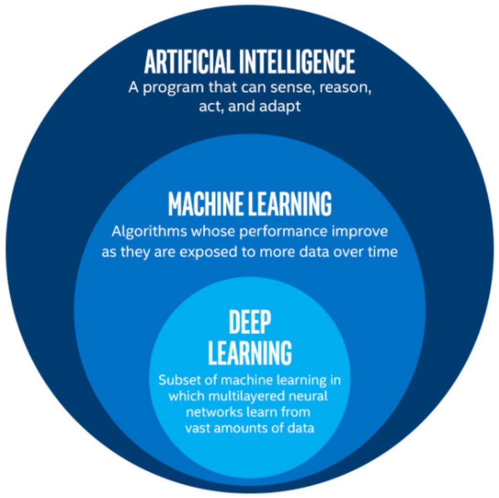

ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

Figure 18: Artificial intelligence taxonomy

When we talk about Machine Learning we have to consider it as a part of Artificial Intelligence. Machine learning is not separate from AI, but instead is a subset of AI. Tom Mitchell's definition for ML is the most widely known: "A computer program is said to learn from experience E, with respect to some class of tasks T and performance measure P if its performance at tasks in T as measured by P improves with experience E."
In other words, machine learning is simply a technique for realizing AI which involves large amounts of data and algorithms to learn how to perform a specific task. Another important concept to take into account is that ML cannot access any knowledge outside of the data provided.
The same consideration is still valid when we compare ML and Deep Learning. These are not two separated approaches, but the latter is a subset of the former instead.
Specifically, Deep Leaning is just a particular type of ML where multiple hidden layers are used in order to improve the accuracy of the model.
Machine learning can be summarized as learning a function f that maps input variables X to output variables Y.

$$Y = f(x) \tag{10}$$

An algorithm learns this target mapping function from training data.
The complexity of the task comes from the fact that the form of the function is unknown, so different machine learning algorithms have to be compared in order to find which of them fits the data better. Different algorithms make different assumptions or biases about the form of the function and how it can be learned.

3.4 Parametric and non parametric machine learning algorithms

The parametric assumption can greatly simplify the learning process, but can also limit what can be learned. Algorithms that simplify the function to a known form are called parametric machine learning algorithms.
Basically, the form of the function is determined a priori and the learning phase consists of finding the best parameters for the function based on the training data. The assumed functional form can be a linear combination of the input variables and in that case the learning algorithms are called "linear machine learning algorithms", but non-linear combinations can be used, too. In that case the complexity of the model increases according with the degree of the polynomial.
Parametric algorithms have the great advantages of being faster and simpler to implement and interpret results. Furthermore, they can perform well even with smaller training data and even if they do not perfectly fit the data.
On the other hand algorithms that do not make strong assumptions about the form of the mapping function are called non-parametric machine learning algorithms. By not making assumptions, they are free to learn any functional form from the training data. For these reasons they achieve better flexibility because they do not make any kind of assumption on the data. Nonetheless they require a greater amount of data and generally speaking they are slower to train. They are prone to a greater overfitting, too.

3.5 Supervised and Unsupervised Learning

Another characteristic of ML algorithms is related to their Supervised or Unsupervised nature. We talk about supervised learning when given input variables X and an output variable Y the goal of the algorithm is to learn the mapping function from the input to the output. The majority of practical machine learning uses supervised learning.
On the other hand we define unsupervised learning any algorithm which can use only input data X for its reasoning. In that case no corresponding output variables are available. This kinf of algorithms are left to their own devises to discover and present the interesting structure in the data.

3.6 Reinforcement learning

Reinforcement learning is nether a kind of supervised or unsupervised learning. In fact the labels of the target are not known and it has not as main goal to find similarities and differences between data-points. We can then say that it has its own independent taxonomy.
Since people from many different backgrounds have stated using deep neural networks to solve a wide range of new tasks it has literally exploded in recent yeas. Its usage includes how to learn intelligent behaviour in complex dynamic environments where the adaptiveness of the system plays an important role and the classical training and testing approach is not suitable.
Another motivation that justifies the huge success of reinforcement learning is that thanks to the progress made by the mechanic and electrical engineer areas the robot and in general any kind of actuator are now physically able to execute all the performable tasks from a human being and even more. So the only limitation of the artificial agent is related with their reasoning capabilities and in that sense reinforcement leaning provides the needed flexibility.
Basically a reinforcement learning model instead of process a huge amount of input and target data in order to minimize the residual function with the techniques discussed in the previous sections, the policy network produces a random initial network and then interacts with the environment in order to determine the quality of the result. If the result is positive the network will receive a reward while if it is negative it will receive a punishment.
The model will then change the architecture and the weights of the network accordingly to the feed-backs received by interacting with the environment.
The results of the network are usually choices to be taken and the model selects the next action alternating exploring phases with exploiting phases. This means that the network will not select always the best action even if the model already knows that it will lead to an high reward because the approach consist of admitting the possible existence of new and better choices. In any case the policy network will slowly avoid actions that lead to negative rewards while positive rewards will become more and more likely.

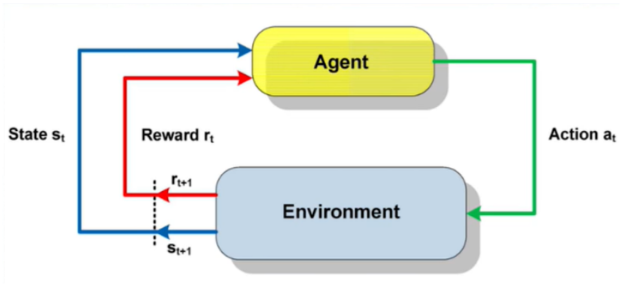


Figure 19: Reinforcement learning model diagram

3.7 Overfitting

Overfitting occurs when our model becomes really good at being able to classify or predict on data that was included in the training set, but does not perform so well with data that it was not trained on. In other words the model fits so well the training set that it cannot be applied to anything else.

An overfitted model is a statistical model that contains more parameters than can be justified by the data. Because of that reason such a model is not useful for forecasting purposes since it is not able to generalize what it has learnt to new data.
The figure alongside illustrates an overfitted polynomial model which best follows the training data compared with a regularized linear model which does not fit perfectly the data, but it is more suitable to predict on new data.
In order to avoid overfitting during the training phase a validation is performed on the training metrics. Basically the weights learnt on the training set are evaluated applying the model to the validation set to calculate the accuracy and loss. If the validation accuracy and loss are considerably worst that the training metrics then that is indication that our model is overfitted.

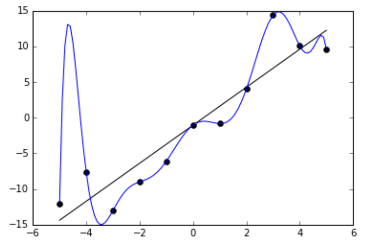


Figure 20: Example of overfitted polynomial model and regularized linear model

3.8 Splitting the data into Training, Validation and Test sets

In order to evaluate the quality of a model the data have to be split into three parts. The first of them is called training set and it consists of the majority of the data. This data are used to train the model and so find the "first version" of the NN. The validation set is used to evaluate the fitting quality of the model on the data and to tune the hyperparameters, while the test set is not used at all in the learning process and it consists of new data never seen before by the NN. It is used to measure the predictive capabilities of the model in a real world scenario.
About the data-set split ratio some models need substantial data to train upon, so in that case it is better to opt for the larger training set as possible. If we are facing models with very few hyperparameters it will be easy to validate and tune, so you can probably reduce the size of your validation set, while on the other hand if the model has many hyperparameters, a large validation set will be needed as well. A cross-validation approach can perform better when a small amount of data is available. For instance K-fold cross-validation consist of dividing the data into K parts and iteratively use one part at the time as validation set while training on the others.

3.9 Neural Networks

Neural Networks are a particular kind of parametric and supervised model. They are inspired by the human brain and for this reason the atomic element composing a neural network is called neuron. At each neuron corresponds a value representing if the neuron is activate or not. Neurons are organized into layers and multiple layers can be stack in order to compose the Neural Network. The neurons belonging to the same layer are not connected with each other but they are connected with the neurons of the next layer.
A Neural Network can be defined fully connected or dense if all the neurons of a layer are connected to all the next layer while it is called sparse Neural Network otherwise.
The first layer is called input layer and the last layer is called output layer, while the layers in the middle are called hidden layers and they perform most of the computation required by the NN.

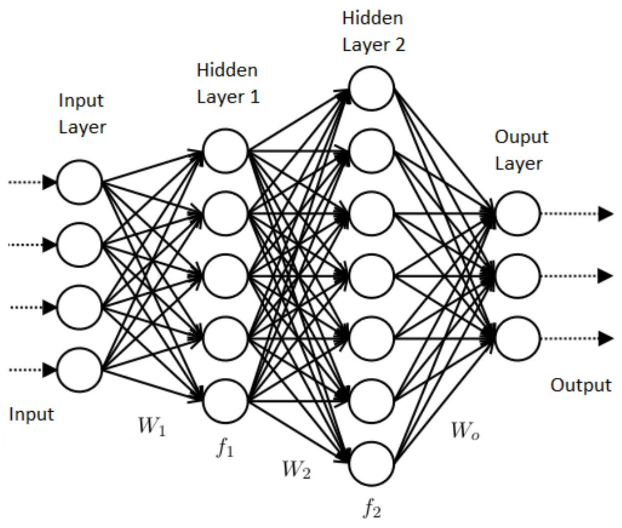


Figure 21: Example of NN with 4 layers

The connection between two neurons is called channel and every channel has a related value called weight. For each layer the inputs are multiplied to the corresponding weights and their summation is sent as input to the neurons in the next layer. This way the data is propagated through the network and this process is called forward propagation. The hyperparameters are setup setting of the network defined a priori and they are not going to be learnt by the NN. The number of layers and the number of neurons for each layer are some of the hyperparameters. At the end of the forward propagation the network is not trained yet. During the training process along with input the NN also has the output fed to it. So the predicted output is compared with the actual output to realize the error and so define what is called the residual function. The magnitude of the error indicates the quality of the current model and it can be used to update the weights though the network. This process is called back-propagation and it can be performed multiple times in order to improve the skills of the network.
The number of back-propagation steps is called epoch and it is an other hyperparameters. In general terms the higher is the number of epochs the better is the accuracy of the

network, but this is not always true since the model can achieve a better accuracy and then diverge if the number of epochs is too high. In addition it has to be noticed that an high number of epochs will affect the amount of time needed to train the network without achieving a significantly better performances.
At the end of each layer the weights have to be recalculated using what is called an activation function. Despite an activation function can be both linear or not, non-linear activation functions are used in most of the cases. In fact, without a non-linear activation function the output of the final layer would be equal to a single layer with a polynomial representation of all the previous layers.
Because of the nature of specific activation function adopted many layers will not have any impact on the model. In order to make them fire in any case, even when the value of the activation function calculated in the corresponding point is zero, a constant value is often added to the output of each layer. This is called bias and it can be interpreted as an artificial input of the layer with feature value equal to one.
About the loss estimation several ways are possible and similarly to how the distances are calculated in Data Mining, some of them performs better that the other. For this reason the loss formula is an other hyper-parameter since it cannot be learnt by the model.
Some of the most common loss evaluation formulas are the following:

- MAE (Mean Absolute Error):

$$MAE = \frac{\sum_{i=1}^{n} |y_i - f(x_i)|}{n} \tag{11}$$

- MSE (Mean Square Error,):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 \tag{12}$$

- RSS (Residual Sum of Squares):

$$RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2 \tag{13}$$

Where for all of them $f(x_i)$ stands for the predicted value of x_i and y_i is the actual value.
At this point it is reasonable to wonder how the weights are updated depending on the current error. Thinking about it it will soon be clear that is not possible to use a grid approach. Trying all the possible updates in the domain will take too much time and there will be any proof that at some point the result obtained will the best. So a better and optimized approach is needed.

3.10 Gradient descent

Gradient descent is an optimization algorithm that leads to the best weight update for each epoch and despite many different variations are possible the main concept remains valid for all of them.
It is used to minimize the loss function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. The requirement is that the loss function has to be totally differentiable in the domain we are considering.
Considering the chain rule for derivative function:

$$\frac{d}{dx}f(g(x)) = f'(g((x)) * g'(x) \tag{14}$$

it should be clear that in order to calculate the gradient of the weights for the last layer we first have to calculate the gradient of the the previous layers.
The step size indicates the magnitude of the update for each back-propagation. A large step size can cause the model to skip the global minimum of the residual function while a small step size will effect badly the performances.
An other consideration about gradient descent regards the application to convex or non-convex function.
If the loss function is convex we are facing a convex optimization problem where the loss function has only one optimal solution which is the a global minimum while on the other hand if the loss function has more then one minimum (many local minimums and one global minimum) the problem needs a non-convex optimization approach and there is a good chance to end up being stuck into a local minimum; of course in that case the solution will not be globally optimal.

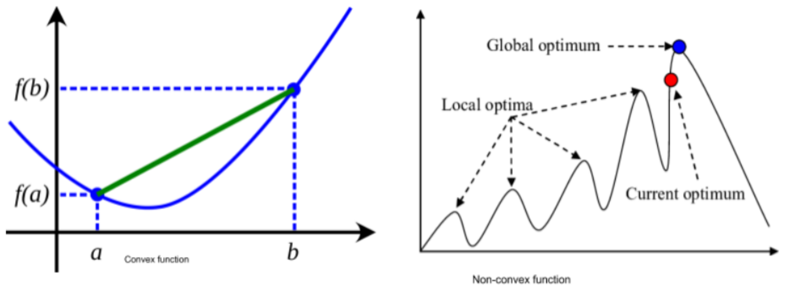


Figure 22: Convex and non-convex functions comparison

If we are facing a convex optimization problem Gradient descent (also called Batch Gradient descent) works quite well, it will use the whole data-set and it will eventually find the minimum located in its basin of attraction.
On the contrary Stochastic gradient descent does not use the whole data-set at the same time and several samples (called mini-batches) are selected in order to give to the model the possibility to overcome a local minimum avoiding to be stuck in its basin of attraction. Choosing the better size of a mini-batch is a trade off between being able to bounce out of a poor local minimum while not avoiding the global minimum or a better local minimum.

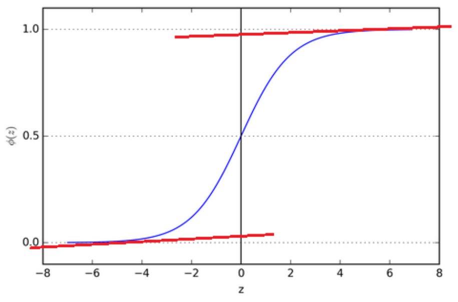


Figure 23: Vanishing gradient due to sigmoid activation function

Some specific activation function or weights can lead to what is called vanishing or exploding gradient. It occurs when the gradient of the activation function is too small or too big. For example taking into consideration the sigmoid activation function:

$$sig_\theta(x) = \frac{1}{1 + e^{-}\theta^T x} \tag{15}$$

We can see how the values close to zero and one have a derivative close to zero. This will cause a very small learning rate of the network. For this reason nowadays the most used activation function is relu (Rectifier function) which has a faster learning speed than sigmoid and it is not prone to vanishing gradient.

$$relu(x) = x^+ = \max(0, x) \tag{16}$$

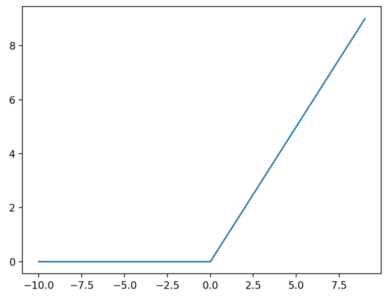


Figure 24: Relu function

3.10.1 Mini-batching

Mini-batching is the technique used by some modified versions of Gradient descent such as Stochastic gradient descent to overcome the limits imposed by vanishing or exploding gradient.
The batch size is a hyperparameter of gradient descent that controls the number of training samples to work through before the model's internal parameters are updated.
The batch size can be between one and the cardinality of the data points.
If the batch size is equal to one the process is called "Online leaning" and in that case the the network weights are updated after each training example. Despite this can make the learning faster, it will also adds instability to the learning process because the weights widely vary with each batch. On the other hand if the weights are updated considering all the data points the learning process is called "Batch Forecasting".

4 Implementation of the ScaledHome system

The implemented ScaledHome system is composed by the following entities:

- The **Actuators Controller** is the entity in charge of managing all the actuators installed inside the house, it does not contain any business logic since it has been built to merely perform actions once it receive an Mqtt command message by the Home Controller.
- The **Sensors Controller** manages all the sensors placed in the house collecting humidity and temperature from each of them every time it receives a Mqtt request message by the Home Controller.
- The **Home Controller** is the centralized entity which contains all the business logic related with the hardware installed inside the house. It sends appropriate Mqtt messages depending on the behaviour needed to perform the desired simulation and it stores the data collected while keeping track of the state of the house during the the simulation interval.
  It provides a discovery service used to find and interact with the two controllers and guarantees the synchronization of the model with the actual sensors and actuators state.
- The **Environment Simulation Agent** interacts with the Home Controller to run real world simulations taking care of all the scaling aspects due to the mapping of the real state into the simulation one.
- The **Actions Planner Agent** provides to the house the actions to perform in order to make the ScaledHome system react to the outside environment taking into account the target state that has to be reached and the power saving policy.

While both the Actuators and Sensors controller have to be placed on a corresponding Raspberry Pi in order to perform the actual actions and provide real data, the Home Controller can be deployed on a local machine or on a cloud host provider since it works independently by the underlying infrastructure thanks to the micro-services techniques adopted.
Regarding the Environment Simulation Agent and the Actions Planner Agent, they can be deployed on a different machine because they do not interact with the controllers directly, but they have to know the URI location of the Home Controller in order to get the state of the house and to provide the required actions to react to the outside environment.

4.1 Actuators and Sensors controllers

4.1.1 Actuators Controller

We define actuators all the devices installed inside the house able to change proactively both the outside and the inside state. According to that definition the lamp, the fan, the heater, the AC and all the motors make up the set of the actuators. They are controlled by a Raspberry Pi3 equipped with a Servo HAT which once properly connected to an additional power supply can provide the needed energy to move all the 15 motors.

The following code shows how the different actuators are handled: for the motors it has been used a specific library called **ServoKit** which is sending the right binary sequence representing the desired angle to the motor while for the other actuators (lamp, fan, ac and heater) GIO standard output ports have been used to turn these actuators on or off depending on the needed behaviour.

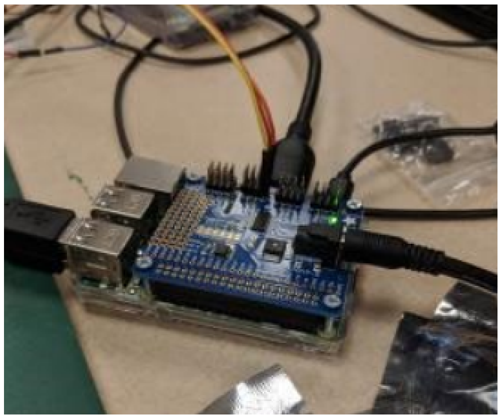


Figure 25: Servo HAT connected to Raspberry Pi3

```python
from adafruit_servokit import ServoKit
import RPi.GPIO as GPIO
kit = ServoKit(channels=16)
def closeMotor(pin):
    kit.servo[pin].angle = getClosingAngle(pin)

def openMotor(pin):
    kit.servo[pin].angle = getOpeningAngle(pin)

def openAll():
    for x in range(0, 16):
        openMotor(x)

def closeAll():
    for x in range(0, 16):
        closeMotor(x)

def setupGPIO():
    GPIO.setup(5,GPIO.OUT) # lamp
    GPIO.setup(6,GPIO.OUT) # heater
    GPIO.setup(12,GPIO.OUT) # fan
    GPIO.setup(13,GPIO.OUT) # ac

def turnOnLamp():
    GPIO.output(5,GPIO.HIGH)

def turnOffLamp():
    GPIO.output(5,GPIO.LOW)
```
Listing 3: Actuator controller

4.1.2 Sensors Controller

The sensors scattered throughout the house are controlled by a Raspberry Pi3 connected to a T-Cobbler Breakout used to connect signal, power supply and ground wires to the physical sensors. The model of the sensors used is DHT11 and they do not need any additional power supply because they can work with a 3.3V output which is already supplied by standard Raspberry Pi3 outlets. According to the data sheet the sensor DHT11 can measure both temperature and humidity. Temperature can be measure in the range [0°C,50°C] while humidity is a percentage and it can be measured in the range [20%,90%]. In terms of accuracy the one related with temperature is ±2 °C while the one for humidity is ±5%.
The following table shows the data sheet specifications.

	Measurement range	Accuracy	Response Time
Temperature	[0°C, 50°C]	[±1 °C, ±2 °C]	[6s, 30s]
Humidity	[20%, 90%]	[±1%, ±5%]	[6s, 15s]

The code below describes the behaviour of the sensors controller, in fact it discards the temperature and humidity records tagging them as corrupted if they are out of the range allowed or if the new record collected is too different if compared with the last one.

```python
import Adafruit_DHT
import time

sensor=Adafruit_DHT.DHT11
gpioArray=[4,6,12,18,19,24,25,26]

def checkTemp (humidity, temperature, h_lower_bound, h_upper_bound,
    t_lower_bound, t_upper_bound):
    control = True
    if humidity is None or temperature is None:
        control = False
    elif temperature<t_lower_bound or temperature>t_upper_bound :
        control = False
    elif humidity<h_lower_bound or humidity>h_upper_bound :
        control = False
    return control

for pin in gpioArray:
            attempts = 0
            while True:
                humidity, temperature_celsius = Adafruit_DHT.read_retry(
    sensor, pin)
                check_temp = checkTemp (humidity, temperature_celsius,
    humidity_lower_bound, humidity_upper_bound, temperature_lower_bound,
    temperature_upper_bound)
                if (check_temp ):
                    break
                elif (attempts >= attempts_limit):
                    humidity = "error"
                    temperature_fahrenheit = "error"
                    break
                else:
                    attempts += 1
                    time.sleep(1)

```

Listing 4: Sensors controller

4.2 Controllers interaction

In the beginning the two controller did not interact each other at all, they were simply managing the corresponding devices without any kind of entity in charge of leading their business logic.
In fact, there was not business logic at all, their behaviour was strictly bounded to the simple testing scenario and it was not possible to aggregate the state of the motors with the corresponding humidity and temperature for each sensor.

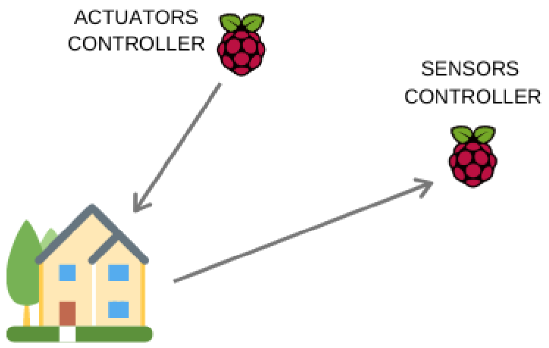


Figure 26: Controllers interaction

In order to collect data and perform actions inside the house the first task has been to delegate the incoming business logic to a specific entity and to define the best way to make the two controllers interact each other while keeping a scalable and flexible architecture.
In fact, according to the proposal published by the Professor Turgut's team[23], the first task has been to provide an easy way to access the data and to run new simulations inside the ScaledHome system.

4.3 ScaledHome architecture

4.3.1 Requirements analysis

The two controllers inside ScaledHome had to collect data and to perform actions in order to map them to each other. In fact, the goal of this project could be achieved only merging the two data flows into a single dataset so it could be processed and analyzed to predict new temperatures depending on different actions or vice-versa.

4.3.2 Two different options

The satisfaction of the requirements could be achieved in two different ways:

- HTTP/REST interaction: this options consists in storing the merged data in one of the two controllers setting a master/slave interaction where the one storing the state contains not only its own data, but also the data received by the other controller.

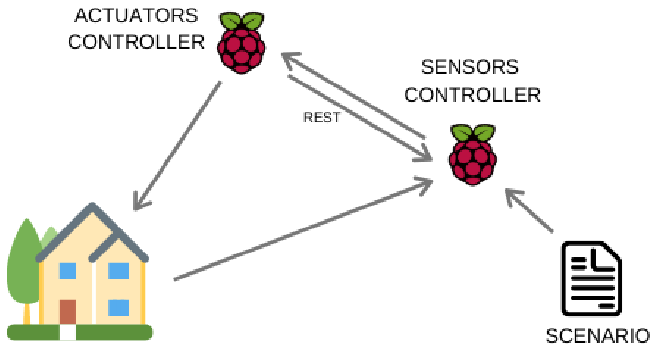


Figure 27: HTTP/REST interaction diagram

The picture above shows one of the two controllers, for example the Sensors controller, acting as a master. In fact it is collecting data from its sensors, but it is also interacting with the actuator controller in order to send new actions to be performed depending on the scenario that has been loaded. In that case a WebServer on the actuator controller would be needed in order to receive instructions and execute them and then reply with the state of the actuators system so the state stored inside the master can be updated.

- Pub/Sub interaction A Pub/Sub system is usually composed of one broker and a set of publishers and subscribers. Basically a Pub/Sub broker handles one or multiple topics, where publishers can publish messages which will be forwarded to interested subscribers. Each publisher has to define the type of the message he is going to publish while each subscriber has to explicitly declare its interest for specific message types. Taking into account this kind of interaction between the ScaledHome entities a different solution consists of avoiding a direct interaction between the two controllers by placeing a rendezvous point between the two, such as a Pub/Sub Broker entity. This broker also has to communicate with another entity, that has been called "Home Controller", which plays a tougher role in terms of responsibilities and computation than the other two controller.

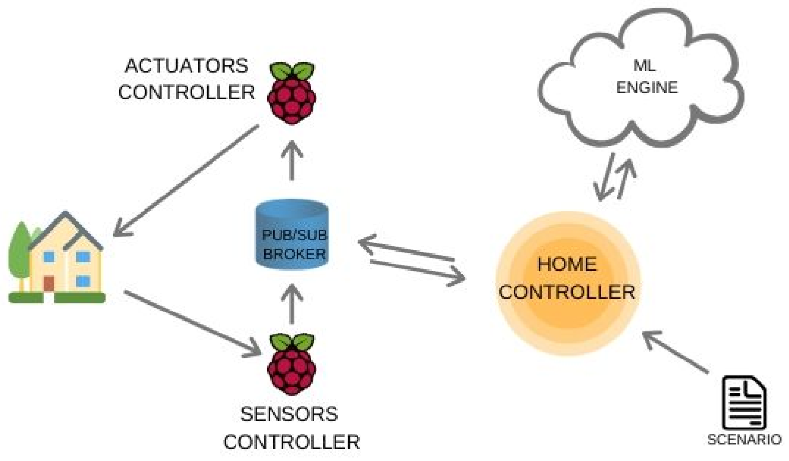


Figure 28: Pub/Sub interaction diagram

4.3.3 Options comparison

The first solution has the great advantage of being simpler that the other one, in fact the number of entities involved is lower and their interaction is based on a http connection which is very well compatible with an heterogeneous set of other technologies. But on the other hand the introduction of a rendezvous point guarantees a greater decoupling in terms of time and space and by adopting this approach the two controllers would just have to perform instructions and collect the data while all the business logic and the coordination of the system is dedicated to the entity with the best computational capabilities.
Furthermore the Home Controller can be exposed outside the in order to collect data and run simulations via a Web Server interface. In addition the Home Controller can serve as the bridge between the Machine Learning engine and the set of simpler and less smart devices close to the house.
Overall the second architecture appears more scalable because the simpler devices close to the house have just to perform instructions and collect data while the Home Controller manages the interaction with the outside components which are more capable in terms of computation. Also the second architecture provides a better defined partitioning of roles inside the system because each component has a specific duty to perform.
The second solution suggest a splitting of components into two different sets:

- The local components set: it is composed by all the actuators and the sensors and their controllers.
- The remote componets set: it is composed by one or more Home Controller and by the Machine Learning engine.

The local components set can be easily improved adding new sensors or new actuators in order to increase the amount of data collected and processed while the remote components set can scale or not depending on the needs without causing a any impact on the local components set system.
So to summarize the second solution is better in terms of:

- Scalability: the Home-Controller can scale without changing the two controllers, while new sensors and actuators can be added without changing the Home-Controller.
- Flexibility: the HomeController provides an higher level of abstraction a different application which want to retrieve some data o perform actions inside ScaledHome.
- "Loose-coupling" rule compliance: the two controllers do not have to know each other because the interaction between them is handled by the HomeController. In fact, every component subscribed to the ScaledHome topic is decoupled in space and time.
- Remote access: thanks to the HomeController is possible to connect to ScaledHome remotely and run different scenarios and simulations.

4.4 ScaledHome middleware implementation

The HomeController acts as a middleware inside the system. It coordinates the entities composing ScaledHome ensuring availability of services and providing a discovery service to find and communicate with the two controllers. It has been implemented in Javascript to guarantee high performance in a Web environment. A well known disadvantage of this programming language is that it is not suitable to handle systems with complex architecutures. Furthermore, it has always been considered unsuitable for server-side services and because of that it has often used only to implement front-end applications. Nowadays these limits have been overcome by using different frameworks.

4.4.1 NodeJS

NodeJS is an asynchronous event-driven JavaScript runtime and it is very well suitable for this case because it can handle server-side services and ”it is designed to build scalable network applications.” In fact, ”many connections can be handled concurrently. Upon each connection, callbacks are fired, but if there is no work to be done, NodeJS will sleep.” [24]
For these reasons we have chosen NodeJS as framework to develop the Home-Controller middleware, because is well scalable, lightweight and it performs very well in cases where we have to face concurrent and multiple requests scenarios.
Despite NodeJS is not ES6 compliant by itself it can be integrated easily using a toolchain such as Babel.js which is already contained into many module bundlers. For this project I have used WebPack as module bundler of the application, it does not only guarantees the ES6 compliance, but it minifies the code footprint compressing variables and lines in order to achieve better performances.

4.4.2 Interaction with the MQTT broker

The middleware handles the interaction with the MQTT broker sending and receving MQTT messages. The whole business logic of the application is so self-contained into the middleware and it interacts with the two controllers through the broker.
The MQTT messages exchanged to shape the behaviour of the system are listed below:

- *discovery*: this is the message published by the HomeController to identify if the Actuators Controller and the Sensors Controller are active and subscribed to the MQTT broker.
- *discovery_reply*: it is the message sent by the two controllers when they receive the discovery message. It contains the specif controller identifier.
- *reconnection_retry*: this is the message sent by the middleware when one or both the controllers are not replying to one or more of the messages they are supposed to be interested in.
- *cmd*: a cmd message is sent to the the Actuators Controller by the middleware and it specifies which are the actions to be performed by the controller.
- *request_record*: a request_record message is published by the HomeController to obtain from the Sensors Controller new data about the state of the house. It can ask for a full record containing the whole state or it can ask just a smaller subset of the state of the system such as the temperature of a room.

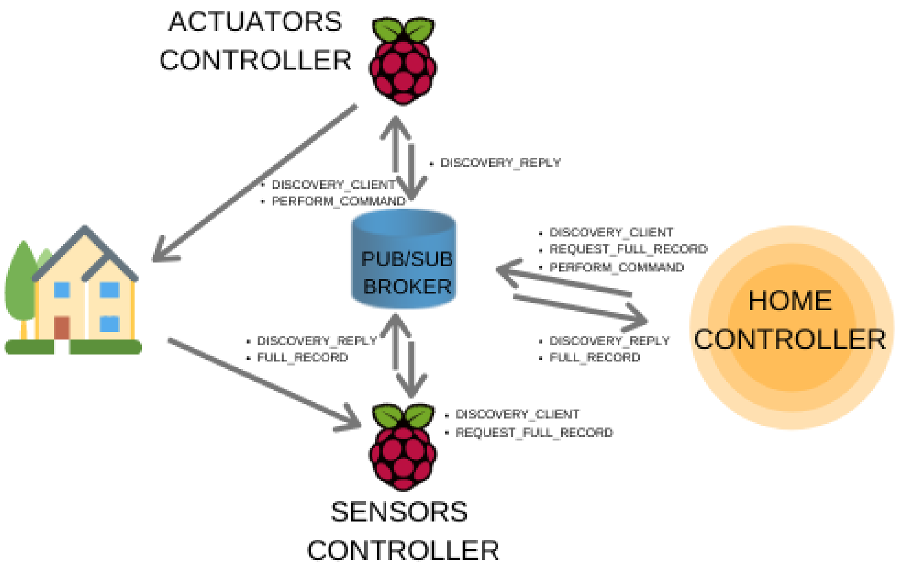


Figure 29: Mqtt message exchange diagram

4.4.3 Web Interface

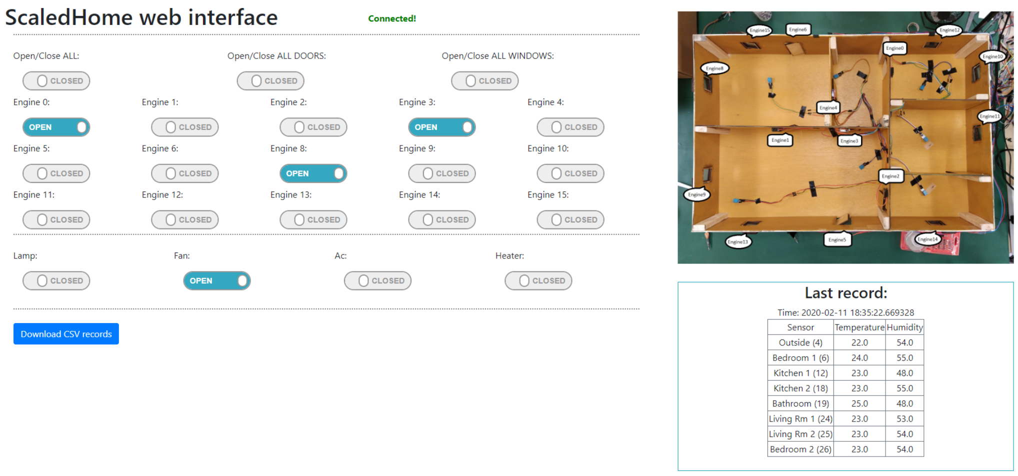
As in the aforementioned section, one of the main goal of the project has been to to encourage its use by third party, potentially remote groups ofresearchers, who can participate in experiments in two different ways:

- Passive: observe in real-time the ongoing experiments via the web interface, streaming data and video. Download and process historical experiment data.
- Active: remotely control the user policy, home policy and smart grid policy through scripts or using a man-in-the-loop model. User can also specify weather patterns or perform real time control of the weather.

This requirement has been satisfied by implementing a Web-Server in NodeJs able to interact with the user via a graphical interface and to perform action as consequences of REST requests. The final results is illustrated by the screenshot below:

It provides:

- an easy interaction with the user to change the state of each actuator in the house
- the last humidity and temperature record collected by the sensors with timestamp
- a way to download all the records collected (as a CSV file) since the activation by pressing the Download button on the GUI.



Last record:

Time: 2020-02-11 18:35:22.669328

Sensor	Temperature	Humidity
Outside (4)	22.0	54.0
Bedroom 1 (6)	24.0	55.0
Kitchen 1 (12)	23.0	48.0
Kitchen 2 (18)	23.0	55.0
Bathroom (19)	25.0	48.0
Living Rm 1 (24)	23.0	53.0
Living Rm 2 (25)	23.0	54.0
Bedroom 2 (26)	23.0	54.0

Figure 30: HomeController middleware GUI

4.4.4 WebSocket

A WebSocket is a communications protocol for a persistent, bi-directional, full duplex TCP connection from a user's web browser to a server. It is initiated by sending a WebSocket upgrade over HTTP via an handshake protocol. Along with the upgrade request header, the handshake request includes a 64-bit Sec-WebSocket-Key header. The server responds with a hash of the key in a Sec-Websocket-Auth header. This header exchange prevents a caching proxy from resending previous WebSocket exchanges. As WebSocket remains open, either the server or the user can send messages at any time until one of them closes the session. The communication can be initiated at either end, which makes event-driven web programming possible. In contrast, standard HTTP allows only clients to request new data.
A WebSocket has been implemented for the exchange of commands and data between the server and the view component. **SocketIO** is an open source packet available for NodeJS which take care of the initial handshake negotiation and provides the WebSocket tunnel in a transparent way. Thanks to that technology the data shown on the GUI are always synchronized with the model maintained on server.

4.4.5 RestFul API

An API is an important concept for Web development. It stands for Application Programming Interface and basically it is a piece of code which allows a software component to talk to another. It can connect multiple servers, containers, application or a server and its graphic interface. The Home Controller makes available end-points for different client connected to the internet in order to allow them to retrieve data and to run custom simulations.
In order to guarantee the security of the system, it is a good practise to check for each rest request if it contains an API key or not. Requests containing the key will be served while wrong or missing keys will lead to the refusal from the server.

The following code shows how the different request by python clients are handled differentiating them by the type field which corresponds to Mqtt message types to send to the Mqtt broker.

```
router.post('/pythonAPI',function(req,res){
    var key = req.body.key;
    var type = req.body.type;
    var value = req.body.value;

    var response = "error no valid key";

    if (key == web_app_settings.api_key) {
      if (type == "cmd"){
        response = "ok";
        if (value.includes("lamp ")){
          middlwareActions.handleLamp(mqttClient.mqttPublish, value.split("lamp ")[1], state);
        }else if (value.includes("fan ")){
          middlwareActions.handleFan(mqttClient.mqttPublish, value.split("fan ")[1], state);
        }else if (value.includes("ac ")){
          middlwareActions.handleAc(mqttClient.mqttPublish, value.split("ac ")[1], state);
        }else if (value.includes("heater ")){
          middlwareActions.handleHeater(mqttClient.mqttPublish, value.split("heater ")[1], state);
        }else if (value.includes("all")){
          middlwareActions.handleMotors(mqttClient.mqttPublish, value.split(" all")[0], state);
        }else if (value.includes("doors")){
          middlwareActions.handleMotors(mqttClient.mqttPublish, value.split(" doors")[0], state, "all doors");
        }else if (value.includes("windows")){
          middlwareActions.handleMotors(mqttClient.mqttPublish, value.split(" windows")[0], state, "all windows");
        }else if (value.includes("motor")){
          middlwareActions.handleMotors(mqttClient.mqttPublish, value.split(" motor")[0], state, value.split(" motor ")[1]);
        }else{
          console.log("unknown cmd from Python: ", value);
        }
      }else if (type == "request"){
        if (value == "last record"){
          response = state.getLastStateAsJsonString();
        }else if(value == "all records collected as string"){
          response = state.getAllRecordsCollected();
        }//else if ... different kind of requests
      }
    }
    res.end(response);
  });
```

Listing 5: REST API for python clients

4.4.6 Home Controller model

According with the aforementioned MVC pattern the state of the architecture has to be delegated to a specific entity inside the system.
Inside the Home Controller this role is played by a JavaScript class which guarantees the synchronization with the Actuators and Sensors controllers. In order to avoid any update conflict this class follows the singleton pattern.
The following code shows the properties stored and how they can be updated by new data records and new actions.

```
class houseModel{
    constructor(){
        this.last_out_temp = "initial_value";
        this.same_temp_counter = 0;
        this.max_temperature_SH = -100000;
        this.min_temperature_SH = 100000;

        this.lamp_state = "initial_value";
        this.fan_state = "initial_value";
        this.ac_state = "initial_value";
        this.heater_state = "initial_value";


        this.header_has_been_written = false;

        this.sensors_controller = {
            id: "sensors_controller",
            state: 0,
            conn_attempts: 0
        };
        this.actuators_controller = {
            id: "actuators_controller",
            state: 0,
            conn_attempts: 0
        };

        this.motors_state = {};

        this.sensors = {};

        this.last_time_record = "initial_value";

        this.all_records_collected = utility.getHeader();

    }

    updateMaxTemp(new_temp){
        var previous_max_out_temp = this.max_temperature_SH;
        this.max_temperature_SH = new_temp;
        var log = "Max temp has been updated from: "+previous_max_out_temp+
    " to: "+this.max_temperature_SH;
        utility.myConsoleLog("updateMaxTemp",log,0);
        fileManager.saveOnFile("./log","txt",utility.myStringLog("
    updateMaxTemp",log));
    }

    updateMinTemp(new_temp){
        var previous_min_out_temp = this.min_temperature_SH;
        this.min_temperature_SH = new_temp;
        var log = "Min temp has been updated from: "+previous_min_out_temp+
    " to: "+this.min_temperature_SH;
        utility.myConsoleLog("updateMinTemp",log);
        fileManager.saveOnFile("./log","txt",utility.myStringLog("
    updateMinTemp",log));
    }

    getLastStateAsJsonString(){
        return JSON.stringify({
            time_record: this.last_time_record,
            sensors: this.sensors,
            lamp: this.lamp_state,
            fan: this.fan_state,
            ac: this.ac_state,
            heater: this.heater_state,
            motors: this.motors_state
        });
    }

    updateStateByRecord(record){
        var record = ''+record;
        var record_list = record.split(settings.csv_separator);
        this.last_time_record = record_list[0];
        this.sensors = {
            outside: {
                temperature: record_list[1],
                humidity: record_list[2]
            },
            sensor_6: {
                temperature: record_list[3],
                humidity: record_list[4]
            }
            // ...
        };
        this.all_records_collected += '\n' + this.
    getStateAsString_no_header();
    }
}

module.exports = houseState;
```

Listing 6: Home Controller Model

The data collected are also stored into a MongoDB database in order to keep track of simulations run in a previous Home Controller sessions.
MongoDB has been chosen not only because the faster saving operations on real-time, but also because of the unstrucuted format of the documents it would be possible to extend the ScaledHome system adding new sensors and actuators without changing the data structure inside the database.
In addition, no joins are required in this specific case, so the normalization which is the main main disadvantage of non-relational databases does not represent a great problem.

The model is initiated by the middleware and it can be referenced by the other components which need to update it.
In order to keep the code as clean as possible the updates are always performed by a delegated functions as shown below:

```
function handleLamp(mqttPublish,action, state){
    var log = "Turning lamp "+action;
    utility.myConsoleLog("handleLamp",log);
    fileManager.saveOnFile("./log","txt",utility.myStringLog("handleLamp",
    log));
    mqttPublish("cmd: lamp "+action);
    state.lamp_state = (action == "on") ? 1 : 0;
}

//...

function handleMotors(mqttPublish,action, state,motor = "all"){
    binary_action = (action == "open") ? 1 : 0;
    var motors_to_change = [];
    if (motor == "all"){
        motors_to_change = settings.allowed_motors;
    }else if (motor == "all doors"){
        motors_to_change = settings.allowed_motors.filter(function(x){
    return x<7});
    }else if (motor == "all windows"){
        motors_to_change = settings.allowed_motors.filter(function(x){
    return x>7})
    }else{
        motors_to_change = [motor];
    }
    mqttPublish("cmd: "+action+' '+motor);
    for (var m in motors_to_change){
        state.motors_state['motor'+motors_to_change[m]] = binary_action;
    }
    var log = (action == "open") ? "Opening "+motor : "Closing "+motor;
    utility.myConsoleLog("handleMotors",log);
    fileManager.saveOnFile("./log","txt",utility.myStringLog("handleMotors"
    ,log));
}
```

Listing 7: Home Controller Model

The same happens when the a new Mqtt message containing the new data collected has been received.

```
async function onMessage(topic, message, mqttClientInstance, state,
    socket_io, mode){
    var mex = ''+message;
    utility.myConsoleLog("main","new mex \""+ mex + "\" from topic \""+
    topic + "\"");
    if (!mex.includes("error")){
        if (mex.includes("record:")){
            var header = utility.getHeader();
            fileManager.saveOnFile("./data","csv",header);
            state.header_has_been_written = true;
            record_file = mex.split("record: ")[1].split(',');
            state.updateStateByRecord(record_file);
            var out_temperature = ''+record_file[1];
            // other types of mqtt messages ...
    }else{
        var log = "Bad message, discarding: "+mex;
        utility.myConsoleLog("main",log,1);
        fileManager.saveOnFile("./log","txt",utility.myStringLog("main",log
    ,1));
    }

}
```

Listing 8: Mqtt onMessage function changes the model

It has to be noticed that as consequence of the filtering done by the Sensors Controller, the data outside the expected range defined on the Home Controller have already been tagged as ”bad data” and this information is used to drop these messages without performing any update on the Home Controller model.

4.5 Home Controller CD and CI

The Continuous Delivery and Integration have been guaranteed by implementing specif tests inside the application.
The following example is a test meant to check if the state provided by the Home Controller model get function is a parsable JSON object and if the same state parsed into string is suitable to be saved on file.

```
const assert = require('chai').assert;
const state = require('../model/state');

var ShState = new state();

describe('ScaledHome state', function(){
    // a check on the json file is needed in order to acces its properties
    it('getLastStateAsJsonString should return a parsable JSON string into
    obj', function(){
        let result = ShState.getLastStateAsJsonString();
        assert.typeOf(JSON.parse(result), 'object');
    });
    // this test is important too, because the result of this function will
    // be saved to file so it should be a string
    it('getStateAsString_with_header should return a String', function(){
        let result = ShState.getStateAsString_with_header();
        assert.typeOf(result, 'string');
    });
});
```

Listing 9: Test on Home Controller model get function

Chai is an NPM support package which provides several and high level assertion operators to check the integrity of the system. These tests are made automatic by an other NPM package called *Mocha* which checks in a transparent way if the tests are passing each time the application is going to be deployed. Since the code is stored in a GitHub repository it is possible to integrate to the deployment process a filter meant to check if the tests are passing or not. The automatic tool which acts as a filter used for the Home Middleware is Called *TravisCI* and it is well compatible with GitHub. It checks automatically if there is a new commit on the interested repository and then is able to approve or not the incoming update. This way every time a commit is made on a local machine it will be checked before been pushed to the shared repository pointed by the automatic application deployment tool in such a way that the code will be free of the bugs if they have been checked by the implemented tests. Inside the NodeJS framework this process is almost completely transparent, the only update that has to be made is inside the *package.json* file and it consists of defining *Mocha* as the default tool for testing. NodeJS will read the script line at every time the application will be stated and it will automatically check if the tests are passing or not. Because the deployment tools are not needed at run-time inside the application it is a good practise to define them as deployment dependencies. This way they will not be installed on the cloud host machine and the application will be faster.

Below there is an example of *package.json* file that shows what just described.

```
{
  "name": "homecontroller",
  "engines": {
    "node": "12.13.0",
    "npm": "6.x"
  },
  "version": "1.0.0",
  "description": "",
  "main": "homeController.js",
  "scripts": {
    "start": "node app.js",
    "test": "mocha"
  },
  "author": "Matteo Mendula",
  "license": "ISC",
  "dependencies": {
    "ejs": "^3.0.1",
    "esm": "^3.2.25",
    "express": "^4.17.1",
    "mqtt": "^3.0.0",
    "prompt": "^1.0.0",
    "readline": "^1.3.0",
    "socket.io": "^2.3.0"
  },
  "devDependencies": {
    "chai": "^4.2.0",
    "mocha": "^7.0.1"
  }
}
```

Listing 10: Example of package.json file

4.6 Docker containerization of the Home Controller

The Home Controller middleware has been containerized in order to ease the deployment over different platform and to guarantee CD and CI.
At the end of this process the middleware application can be run on every machine which has Docker installed. In fact, it is not a coincidence that many hosting providers sell cloud machines with containers application managers already installed.
In order to run the middleware application inside a container, few operations have to be made:

- Declaration of dependencies: if the development environment does not support any kind of packet manager the dependencies have to be declared manually and the installation have to be performed through a bash script file since Docker run over a Linux distribution. Otherwise if the application framework support the development it will be possible to install all the required dependencies in a transparent way by declaring a description file.
  In our case NodeJS has its own packet manager called Node Packet Manager (NPM) which takes care of the dependencies and of their updates. All the dependencies are listed inside the *package.json* file and it is possible to differentiate them depending on their development or production nature.

- Declaration of the application entry point: independently by the framework or language used the declaration of which file has to be run and the command to use will be needed in order to start the application. Regarding the NodeJS specific case, this information is stored in the same JSON file used by NPM to install the dependencies.

- Definition of the Docker settings: this is done by writing a Dockerfile. Docker will read the file and it will used this file to build the image of the application. An example of Dockerfile can be seen below:

```
FROM node:10

# Creation of app directory
WORKDIR /usr/src/homeController

# Installation of app dependencies
COPY package*.json ./

RUN npm install
# If you are building your code for production
# RUN npm ci --only=production

# Bundle app source
COPY . ./homeController

EXPOSE 8080
CMD [ "npm", "start" ]
```

Listing 11: Home Controller Model

The first line (1) declarea which is the image to be inherited in order to build the application. The image *node* : 10 has been chosen because it has already NodeJS installed. Then the working directory is declared and the *package.json* file is copied inside it.
Line (9) runs the NPM command used to install the required dependencies listed inside the JSON file. After that the whole application is bundled and it is now ready to be run. The last thing to do before is to link a port on the physical machine to the port where the image is going to publish the service in order to access it from the outside. The last line simply lists the commands array to be executed (it is the set of the starting command and its parameters).
At the end of this process the it will be possible to build the image and to run it with the simple docker commands: *docker build image_name* and *docker run image_name*. So the application will be accessible by navigating to the local port defined inside the Dockerfile.

4.6.1 Managing different images with Minikube orchestrator

Minikube is a tool that makes the execution of Kubernetes easy inside the local machine. Minikube runs a single-node Kubernetes cluster simulation inside a Virtual Machine (VM) a multiple node cluster. It is meant for users looking to try out Kubernetes or develop with it day-to-day, but the same concepts can be applied in the cloud when the application has to be deployed and published online. On *Google Cloud Platform*, for example, the process is made even easier because it is possible to activate Kubernates clusters just via the Dashboard web interface. The steps needed to run a Docker image in a Kubernates cluster are the following:

- First the Minikube VM and the Kubernetes service has to be stated. This can be done just running the command

```
        minikube start
```

- Then it is necessary to write a deployment file declaring which are the desired setting for the deployment. This file can be a json file or a yml file. It contains the number of replicas, the name of the deployment and the name of the application. In addition is mandatory to declare which Docker image has to be used to build the pods. This can be done by writing the local uri or it can be a magnetic link to a Docker image hosted in a Docker image repository. In this case the Home Controller image has been pushed to *DockerHub* a service provider offered by Docker for finding and sharing images which works like GitHub for code repositories.
  Finally we have to explicitly declare which is the port provided by the image to be access from outside the container. This port must coincide with the port defined in the Dockerfile used to build the image.

The following is an example of deployment.yml file:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: home-controller-deployment
  labels:
    app: homeController
spec:
  replicas: 3
  selector:
    matchLabels:
      app: homeController
  template:
    metadata:
      labels:
        app: homeController
    spec:
      containers:
      - name: homeController
        image: mattemendu/homeController:v1.0
        ports:
        - containerPort: 8080
```

Listing 12: Deployment.yml file example

- The information about the deployment and the consequent pods can be viewed by:
  - the Kunernates Dashboard which inside Minikube can be accessed by running the command

    ```
                minikube dashboard
    ```

    and going to the given URL
  - by running the commands

    ```
                kubectl get deployments deployment_name
    ```

    and

    ```
                kubectl describe deployments deployment_name
    ```

- Then it is needed to create a Service object that exposes the deployment, for the Home Controller a Load Balance service has been implemented by running

  ```
         kubectl expose deployment deployment_name --type=LoadBalancer
      --name=service_name
  ```

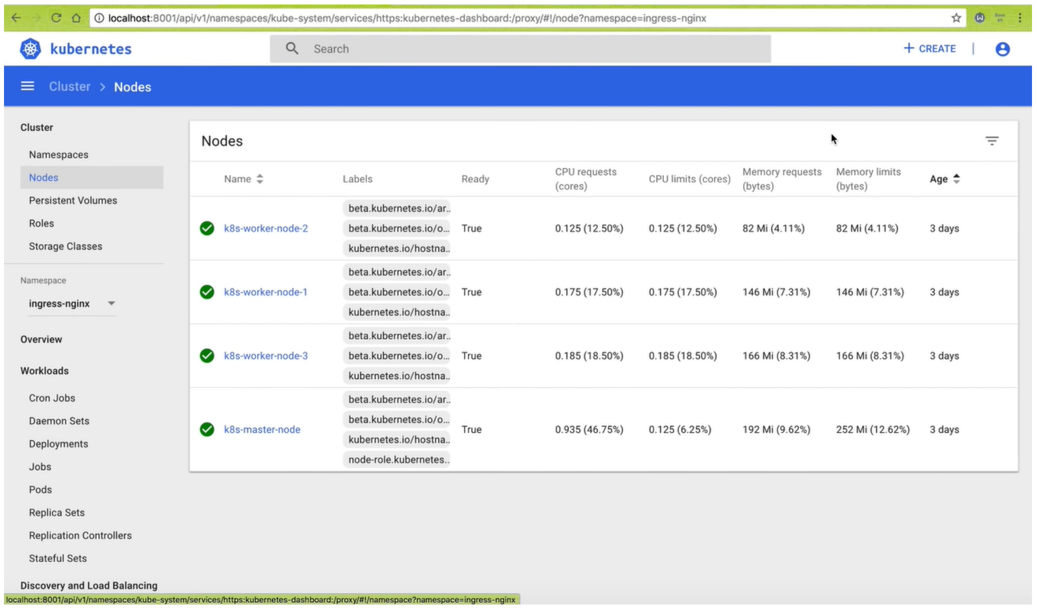
  the running services are listed real-time on the Dashboard, anyway more details can be get by running

  ```
         kubectl get services service_name
  ```

- The last step is to run the service and go to the URL to access the port exposed by the cluster. In order to do that you can simply run

  ```
         minikube service service_name
  ```

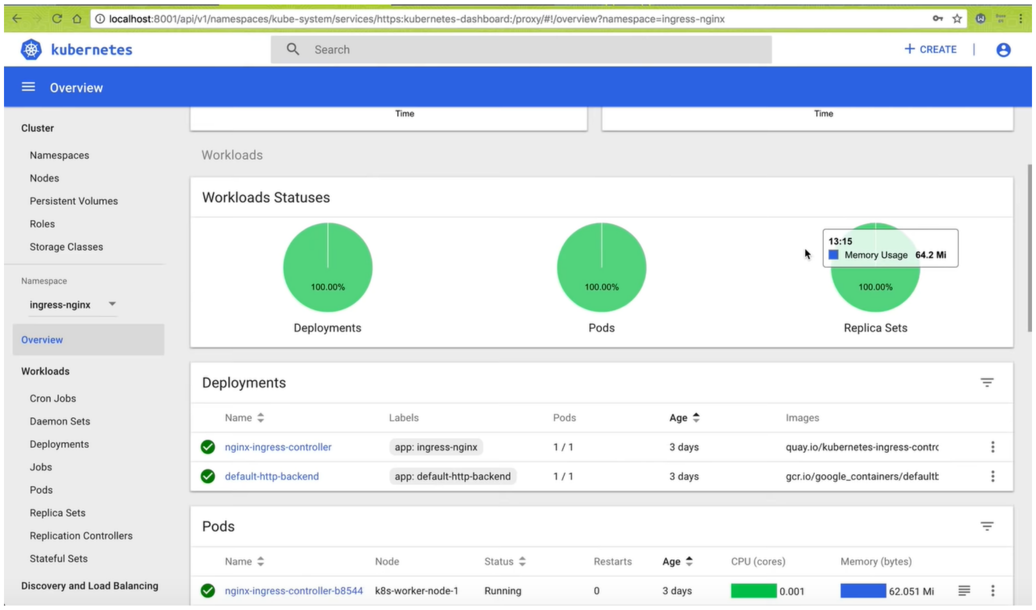
At the end the application will be accessible as if it is a single image, but in a transparent way Kubernetes will handle the pods in order to keep their number constant despite possible crashes of a single pod by deploying a new one automatically.
More fine-grained operation can be done such as increasing the number of pods when the traffic is higher than a defined threshold or when the overhead of an image is too high. This became easier to do with the Dashboard because it provides not only an higher lever of abstraction, but also shows the monitored cluster state in a graphical way.


localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/node?namespace=ingress-nginx
kubernetes
Search
+ CREATE
Cluster > Nodes
Cluster
Namespaces
Nodes
Persistent Volumes
Roles
Storage Classes
Namespace
ingress-nginx
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Discovery and Load Balancing
localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/namespace?namespace=ingress-nginx

Nodes

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
✓ k8s-worker-node-2	beta.kubernetes.io/ar.. beta.kubernetes.io/o... kubernetes.io/hostna..	True	0.125 (12.50%)	0.125 (12.50%)	82 Mi (4.11%)	82 Mi (4.11%)	3 days
✓ k8s-worker-node-1	beta.kubernetes.io/ar.. beta.kubernetes.io/o... kubernetes.io/hostna..	True	0.175 (17.50%)	0.175 (17.50%)	146 Mi (7.31%)	146 Mi (7.31%)	3 days
✓ k8s-worker-node-3	beta.kubernetes.io/ar.. beta.kubernetes.io/o... kubernetes.io/hostna..	True	0.185 (18.50%)	0.185 (18.50%)	166 Mi (8.31%)	166 Mi (8.31%)	3 days
✓ k8s-master-node	beta.kubernetes.io/ar.. beta.kubernetes.io/o... kubernetes.io/hostna.. node-role.kubernetes..	True	0.935 (46.75%)	0.125 (6.25%)	192 Mi (9.62%)	252 Mi (12.62%)	3 days

Figure 31: Kubernetes dashboard showing worker nodes



Deployments

	Name	Labels	Pods	Age	Images
✓	nginx-ingress-controller	app: ingress-nginx	1 / 1	3 days	quay.io/kubernetes-ingress-contrc
✓	default-http-backend	app: default-http-backend	1 / 1	3 days	gcr.io/google_containers/defaultb

Pods

	Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
✓	nginx-ingress-controller-b8544	k8s-worker-node-1	Running	0	3 days	0.001	62.051 Mi

Figure 32: Kubernetes dashboard showing pods and deployments workload

4.7 Data collection

Thanks to the Home Controller middleware the collection of data has been quite easy.
It collects automatically data discovering the two controllers and sending a new *reocord request* each time a record has been received.
A delay time can be set between each new request, but regarding the performed simulations the data has been collected without any delay.
On average the Home Controller collects and stores data every 10 seconds updating its own state, saving a new document inside MongoDB and appending them on csv file. Each new record has been sent to the GUI interface too. This is done in order to maintain the synchronization with the user session.

4.7.1 First simulation: temperature boundaries

Since the final goal is to simulate a real day of different cities around the world, the increasing and decreasing capacities of ScaledHome in terms of temperature change have to be tested in order to map them to real world temperatures.
This has been achieved by setting a simple simulation which first turn on the the lamp to increase the temperature and then when the maximum bound has been reached it turns off the lamp and turns on the external fan. The bounds are detected by identifying consequent records containing the same external temperature. Several simulation has been run in order to find the real bounds where the outside temperature reaches an equilibrium and no more actions on the actuators can be performed to change the state.
This first data analysis step had a great importance in the project because it enabled the identification of relevant patterns and correlations which have been then explored during the development of the project by employing ML techniques.
The following table illustrates the maximum and minimum temperatures which have been identified and the time needed by ScaledHome to reach one temperature bound given as initial temperature the other one.

	Temperature value	Δt
MAX	39±1°C	1100±20 seconds
MIN	21±1°C	150±20 seconds

The graph below shows one of the data-sets collected in the form of plot graph. It has been obtained by using a python library called *Mathoplotlib* which provides useful tools for data visualization.
Because of the the accuracy of the sensor a comfort range which is smaller of the actual range available has been considered for the simulation purposes. So, in the next sections the range considered has been set between $23°C$ and $38°C$.

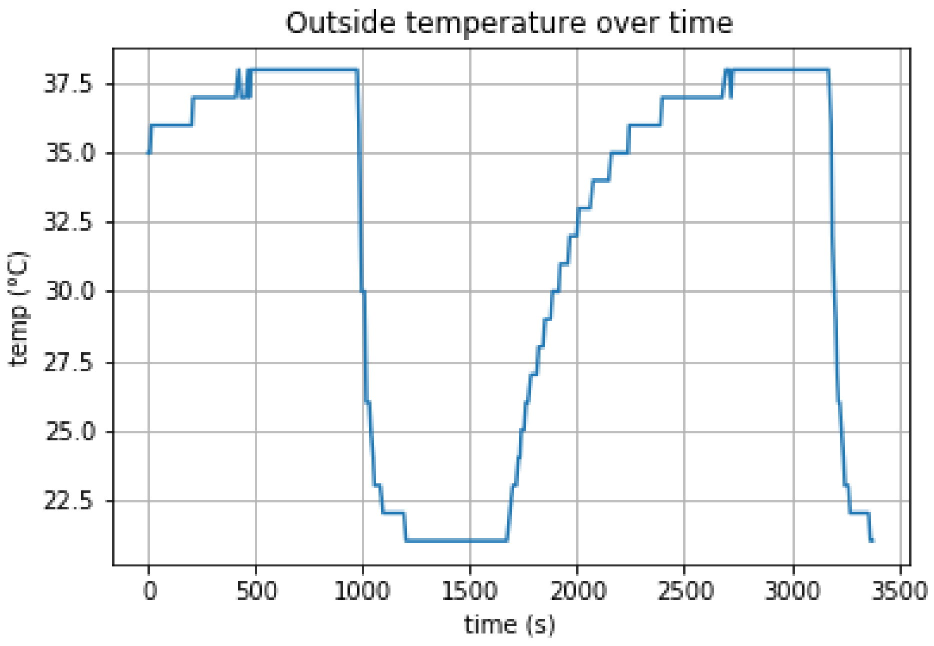


Figure 33: Temperatures collected by the search boundaries simulation

4.7.2 Temperature regression for increasing and decreasing temperature functions

Using another popular Python library for Machine Learning applications called $scikit - learn$ combined with $numpy$ for data analysis it has been possible to find with polynomial regression models the approximated functions which fits the decreasing and increasing temperature trends.
The following graphs represents the regression functions and their improved accuracy increasing the degree of the polynomial regression functions.

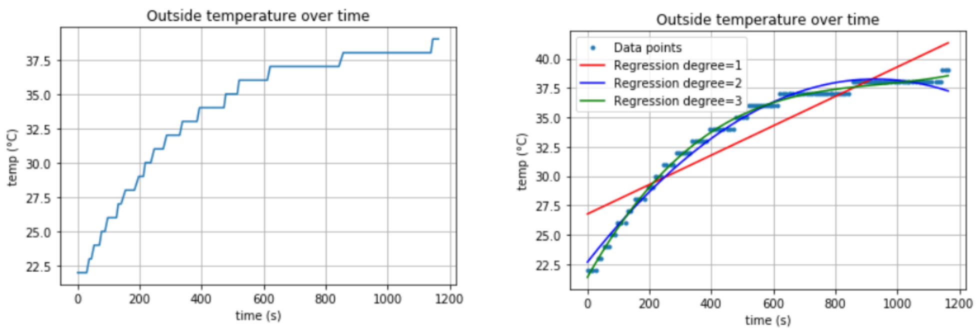


Figure 34: Temperature increasing temperature regression functions

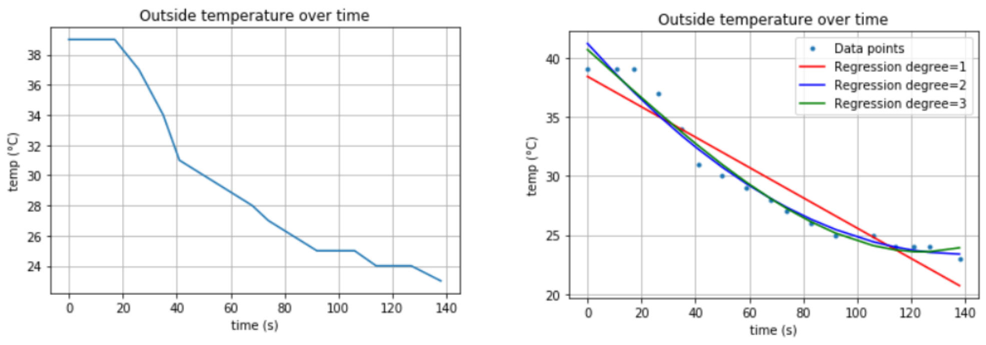


Figure 35: Temperature decreasing temperature regression functions

4.7.3 Temperature and Humidity correlation

Before moving forward performing new simulations, we have analysed the correlations between the state metrics inside the house collected by the sensors. The followings heat-maps show these relationships indicating with a bigger square a corresponding greater correlation among the data. These have been obtained by still using *Mathplot library* and *numpy*.

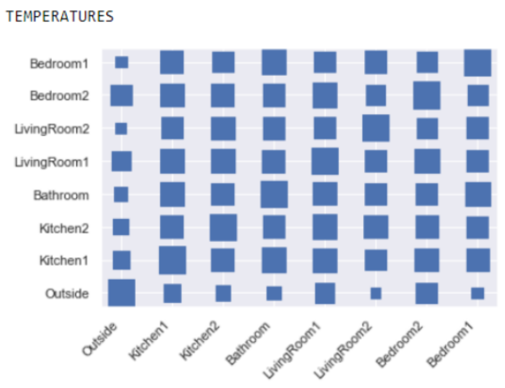


Figure 36: Temperature heat-map

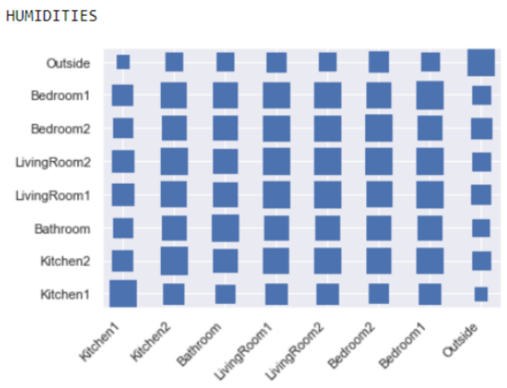


Figure 37: Humidity heat-map

As expected the outside temperature has a greater correlation with the rooms closer to the lamp and the heater while has almost no correlation with distant rooms as kitchen. Regarding the correlation among the inside rooms they have a greater value when the doors are open. This simple and maybe unnecessary information is instead meaningful to prove if the sensors are working properly.

4.8 Simulation of a real day in Milan

Thanks to the data obtained by Milano Weather Station published on Hardvard Dataverse [25] it has been possible to scale the temperatures in Milan in the reachable range of ScaledHome.
Basically, given:

- the actual lower and upper bounds in Milan, mil_{min} and mil_{max}
- the corresponding bounds inside the ScaledHome system, sh_{min} and sh_{max}
- the actual value in Milan to be scaled, M_i

the mapping can be done in accordance with the formula below:

$$SH_i = \frac{M_i - mil_{min}}{mil_{max} - mil_{min}} \times (sh_{max} - sh_{min}) - sh_{min} \tag{17}$$

where SH_i is the value inside the ScaledHome system with reference to M_i.
This consideration was not enough to guarantee the correctness of our mapping because the capabilities of the actuators used to increase or decrease the temperature were not modular. In fact, they have only a binary state. Taking into account the time needed by the ScaledHome system to change its temperature from one bound to the other, we grouped the temperatures in Milan creating time frames of 3 hours each and we set a simulation interval used by the system to reach a new temperature by the previous one. We estimated the duration of the simulation interval by interpolating the increasing and decreasing regression functions. This is meant to give to the lamp and the fan enough time to reach the desired temperature. Then we had to find a way to reach a temperature in a time interval longer than the one needed to reach the target.

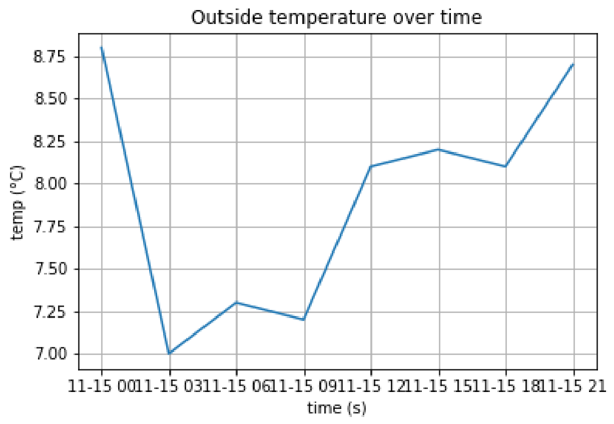


Figure 38: Real Milan data

4.8.1 Scaling approach: Hysteresis

Hysteresis is a well known concept in control theory. It consists of setting an accepted range including the target value where no additional actions have to be performed. For example, considering a temperature controller if the target temperature T_x is higher than the actual temperature T_a it would be necessary to perform actuator actions in order to increase the temperature.
Given a defined accepted bias b, T_x will be in the range $[T_x - b, T_x + b]$ and for that reason the state of the actuators will not be changed until the temperature will reach $T_x + b$ as the upper bound of hysteresis interval while that state will not be changed until the temperature will reach $T_x + b$ in case T_x is lower than T_a.

4.8.2 Two different simulations

Taking into account the hysteresis concept we run a first simulation where we kept the temperature still once we reached the target.

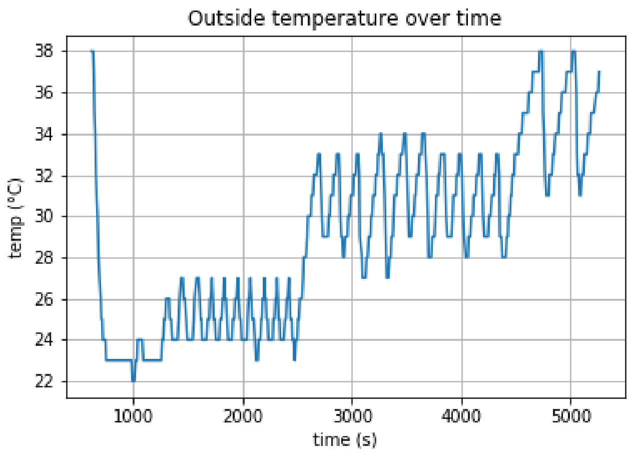


Figure 39: First simulation without sub time frames

Since the matching of these results with the actual data was not satisfying we decided to divide each time frames in sub-time frames reaching for all of them sub-targets obtained dividing the temperature difference by the number of sub-time intervals.

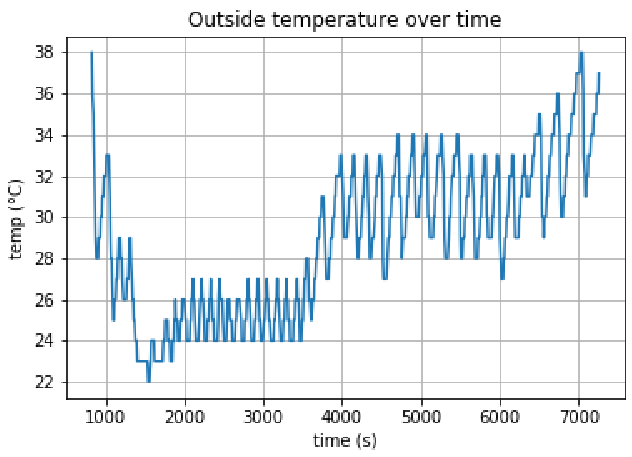


Figure 40: Second simulation with sub interval frames

The comparison between these two different approach is shown by the following graphs.

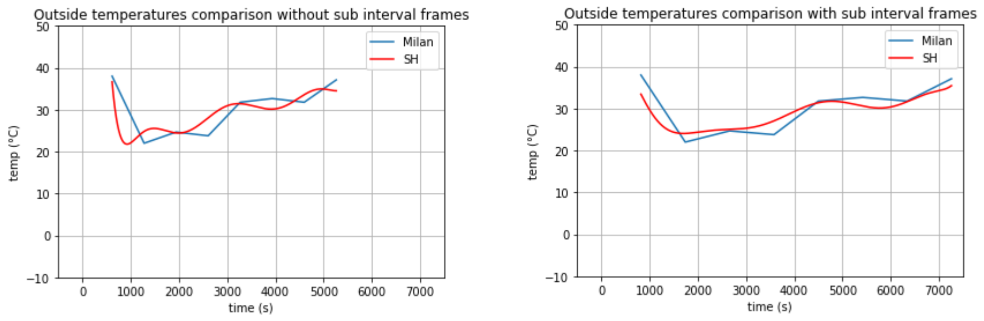


Figure 41: Different approaches for simulating the outside temperature

As it can be seen the second approach leads to a better matching between the data in Milan and the data reproduced by ScaledHome.
As final result we have been able to simulate a real world scenario inside the ScaledHome system.

4.9 Machine learning models

At this point since we were able to simulate a real world scenario the next step has been to train multiple ML models in order to find the best prediction of the house state S_{t2} given a previous state S_{t1} and the action performed.
In order to do that we collected more than 4000 records changing the state of the actuators in a random way every 60 seconds.
We fed the models by given them every record r_i containing all the features we wanted to consider as input values and the corresponding record r_{i+t} containing the features we wanted to predict in the next record as target values.
The t value is a settable parameter as well as the input features and the target features. This flexibility has been introduced in order to consider these values as hyperparameters of the models, since we do not know a priori which are the most relevant features.
We then trained four different machine leaning models:

- K-Nearest Neighbours model
- Support Vector Regression model
- Deep neural network model
- Long Short Term Memory model

4.9.1 KNN: K Nearest Neighbours

KNN is a non-parametric learning algorithm. The k-nearest neighbors algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar values exist in close proximity, in other words, similar values are near to each other. It basically captures the idea of similarity (sometimes called distance, proximity, or closeness) by calculating the distance between points on a graph.
It can be implemented by following the steps below:

- Load the data
- Initialize K to your chosen number of neighbors
- For each example in the data
  - Calculate the distance between the current example from the data and the K closest neighbours.
  - Add the distance and the index of the example to an ordered collection
- Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
- Pick the first K entries from the sorted collection
- Get the labels of the selected K entries

Depending on the case KNN can be used for regression of classification. If we are interested in regression (like in the ScaledHome case) the mean of the K labels will be returned, while on the other hand KNN will return the mode of the K labels in case of

classification.
The K value and the method used to calculate the distance between samples are hyperparameters and they cannot be learnt by the model, on the contrary they have to be defined manually at development time.
The KNN model has been implemented by using the *scikit−learn* Python library which provides high level implementations of many machine learning models.
The following class shows a possible implementation which will be used for tuning the hyperparameters.

```python
from sklearn.neighbors import KNeighborsRegressor

class KNNConfig(GeneralModel):
    def __init__(self, dataset_uri, feature_cols, target_cols,
    prediction_index, n_neighbors, distance_metric):
        super(KNNConfig, self).__init__(dataset_uri, feature_cols,
    target_cols, prediction_index)
        self.n_neighbors = n_neighbors
        self.distance_metric = distance_metric
        self.model = KNeighborsRegressor(n_neighbors=self.n_neighbors,
    metric=self.distance_metric)

    def train_on(self, x, y):
        self.model.fit(x, y)

    def predict_on(self, x):
        return self.model.predict(x)

    def get_restoring_path(self):
        dir_name = os.path.dirname(__file__)
        return os.path.join(
            dir_name,
            'saved_models',
            f'KNNConfig_k{self.n_neighbors}'
        )
```

Listing 13: KNN with settable hyperparameters

4.9.2 SVR: Support Vector Regression

SVR is a different kind of Support Vector Machine, it differs from SVM because it is not used for classification but for regression. While in simple regression we try to minimise the error rate, in SVR we try to fit the error within a certain threshold.
The terms we have to define are:

- Kernel: it is the function used to map a lower dimensional data into a higher dimensional data.
- Hyperplane: in SVM this is basically the separation line between the data classes. Although in SVR we are going to define it as the line that will will help us predict the continuous value or target value
- Boundary line: in SVM there are two lines other than Hyper Plane which creates a margin . The support vectors can be on the Boundary lines or outside it. This boundary line separates the two classes. In SVR the concept is same.
- Support vectors: these are the data points which are closest to the boundary. The distance of the points is minimum or least.

When we are moving on with SVR is to basically consider the points that are within the boundary line. Our best fit line is the line hyperplane that has maximum number of points. The decision boundary is our margin of tolerance that defines which points we are going to consider.

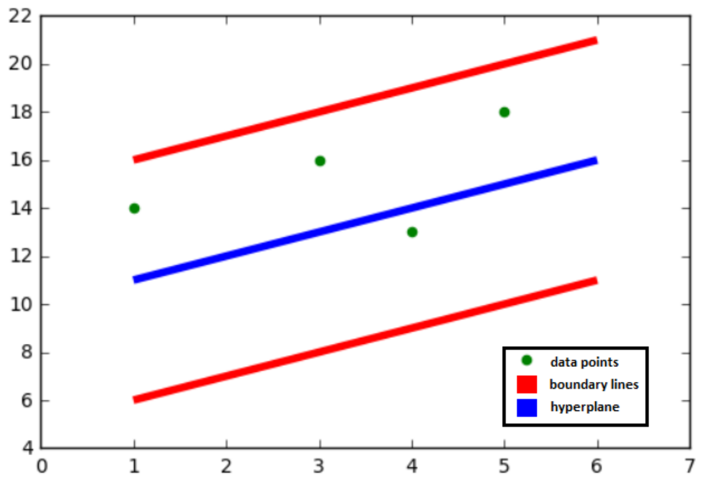


Figure 42: SVR hyperplane and boundary lines

Because SVR can consider just one target value at the time, the following implementation shows how one SVR model has to be defined for each of the target values.

```python
from sklearn import svm
class SVRConfig(GeneralModel):
    def __init__(self, dataset_uri, feature_cols, target_cols,
    prediction_index):
        super(SVRConfig, self).__init__(dataset_uri, feature_cols,
    target_cols, prediction_index)

        self.models = [svm.SVR(gamma='auto') for i in range(len(target_cols
    ))]

    def train_on(self, x, y):
        for i in range(len(self.target_cols)):
            self.models[i].fit(x, y[:, i])

    def predict_on(self, x):
        y_hat = np.zeros(shape=(x.shape[0], len(self.target_cols)))
        for i in range(len(self.target_cols)):
            y_hat[:, i] = self.models[i].predict(x)
        return y_hat

    def get_restoring_path(self):
        dir_name = os.path.dirname(__file__)
        return os.path.join(
            dir_name,
            'saved_models',
            f'SVRConfig'
        )
```

Listing 14: SVR with settable hyperparameters

4.9.3 DNN: Deep neural network regression

A deep neural network is a specific kind of NN where the architecture of the model consists of several and consequent hidden layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output.
The number of epochs corresponds to how many times the back-propagation process will be performed. In general terms, the higher the number of epochs the better will be the accuracy of the network, this means that with an infinite number of epochs any NN will theoretically achieve a zero error prediction on test set. Of course this is not possible, even without considering the vanishing or exploding gradient or the overfitting problems some unexpected record or an insufficient amount of data will lead to a wrong prediction.
DNNs must consider many training parameters, such as the number of layers and number of neurons per layer, the learning rate, and initial weights. Sweeping through the parameter space for optimal parameters may not be feasible due to the cost in time and computational resources. Various tricks, such as batching (computing the gradient on several training examples at once rather than individual examples) can speed up computation. Large processing capabilities of many-core architectures (such as GPUs using $Invidia\ CUDA$) have produced significant speedups in training, thanks to the suitability of such processing architectures for the matrix and vector computations.
A settable architecture for the DNN predictor has been implemented and it has to be noticed that the hyperparameters can be passed to the class in order to identify the bests.
As it can be noticed we used *keras* to implement the DNN. It is a high level library initially independent from Tensorflow, but now it is completely integrated inside the the framework. The reason behind *keras* is to provide an high level interface to build and train machine learning models without taking care of the implementation details like back-propagation and predictions. Different layers can be stack on top of a sequential architecture defining for each of them the shape and the activation functions.

The following code shows the resulting implementation of the NN developed.

```python
from keras.models import Sequential
from keras.layers import Dense

class NNConfig(GeneralModel):
    def __init__(self, dataset_uri, feature_cols, target_cols, prediction_index, loss_function,n_layers):
        super(NNConfig, self).__init__(dataset_uri, feature_cols, target_cols, prediction_index)
        self.model = Sequential()
        self.model.add(Dense(128, input_dim=len(feature_cols), kernel_initializer='normal', activation='relu'))
        for _ in range(n_layers):
            self.model.add(Dense(128, kernel_initializer='normal', activation='relu'))
        self.model.add(Dense(len(target_cols), kernel_initializer='normal'))
        self.model.compile(loss = loss_function, optimizer='adam')

    def train_on(self, x, y):
        self.model.fit(x, y, epochs=5000)

    def predict_on(self, x):
        return self.model.predict(x)

    def get_restoring_path(self):
        dir_name = os.path.dirname(__file__)
        return os.path.join(
            dir_name,
            'saved_models',
            f'NNConfig_k'
        )
```

Listing 15: DNN with settable hyperparameters

4.9.4 LSTM: Long Short Term Memory

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed-forward neural networks, LSTM has feedback connections. It is not meant to process single data points, but to consider entire sequences of data (such as speech or video). In our case it has processed sequences of records in order to predict the consequent sample. The network keeps track of previous sequences remembering the relationships among data points already seen. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.
LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs.
This kind of network is widely used in the natural language research area because it can take into account the sparsity of the concepts expressed by a human speech which is something that a standard neural network is not able to do.

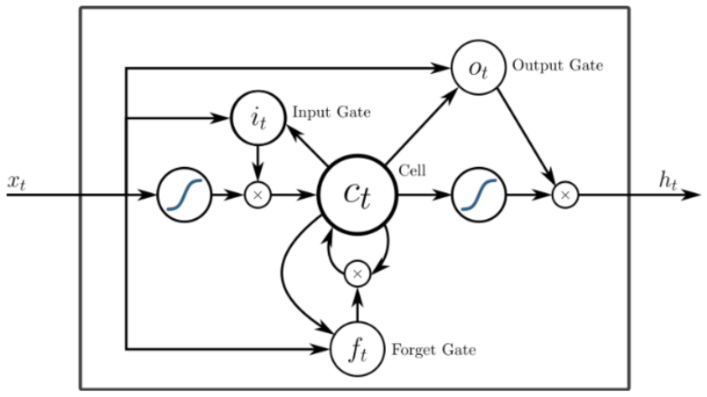


Figure 43: LSTM model architecture

In order to predict new values using an LSTM model the shape of the data fed to the network has to be changed. Basically, an additional dimension will be needed. This dimension is due to the fact that instead of feeding to the network single data points, sequences of entries has to be reshaped by windowing. No change is needed with reference to the shape of target data points, but we have to consider less data points because the first n of them are included inside the first window sequence where n is the size of the window.

The code below represents the structure of the Python class which implements the LSTM model.

```python
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM

class LSTMConfig(GeneralModel):
    def __init__(self, dataset_uri, feature_cols, target_cols, prediction_index, n_layers, n_epochs, n_neurons, loss_function):
        super(LSTMConfig, self).__init__(dataset_uri, feature_cols, target_cols, prediction_index)
        self.prediction_index = prediction_index
        self.n_layers = n_layers
        self.n_epochs = n_epochs
        self.n_neurons = n_neurons
        self.loss_function = loss_function
        self.x_test = create_sequence_from_flat_data(self.x_test, self.prediction_index)
        self.x_val = create_sequence_from_flat_data(self.x_val, self.prediction_index)
        self.x_train = create_sequence_from_flat_data(self.x_train, self.prediction_index)
        self.y_train = self.y_train[self.prediction_index:,:]
        self.y_val = self.y_val[self.prediction_index:,:]
        self.y_test = self.y_test[self.prediction_index:,:]
        self.data_set_parts = {
            'train': {
                'x': self.x_train,
                'y': self.y_train
            },
            'val': {
                'x': self.x_val,
                'y': self.y_val
            },
            'test': {
                'x': self.x_test,
                'y': self.y_test
            }
        }
        self.model = Sequential()
        self.model.add(LSTM(self.n_neurons, input_shape=(prediction_index, len(feature_cols))))
        self.model.add(Dense(len(target_cols)))
        self.model.compile(loss=self.loss_function, optimizer='adam')

    def train_on(self, x, y):
        self.model.fit(x, y, epochs=self.n_epochs, batch_size=5, verbose=2)

    def predict_on(self, x):
        return self.model.predict(x)

    def get_restoring_path(self):
        dir_name = os.path.dirname(__file__)
        return os.path.join(
            dir_name,
            'saved_models',
            f'LSTMConfig'
        )
```

Listing 16: LSTM with settable hyperparameters

4.10 Hyperparameters tuning

Each hyperparameter of the model plays an very important role since it cannot be learnt by the network itself.
Good hyperparameters will lead to an accurate model in a short amount of time while bad hyperparameters can affect the network and lead to local minimums in the residual function.
In order to find the best hyperparamters and so compare the implemented models to find the best for perdiction purposes we implemented a Hyperparameters searcher which compile different versions of the same model using different parameters. The searcher tries all the different combinations of hyperparameters by adopting a grid search approach, but a possible optimization of the tuning algorithm consists of searching the best hyperparameters with an heuristic tree search approach.
Thanks to the fact that every model can be saved as a file and used to predict new data without new training, the Hyperparameter searcher saves the model with the higher accuracy and its hyperparameters.
To support the handling of the implemented models a parent class called "General Model" has been inherited by each specific model. This class provides the common utility functions such as the function used to split the data into training, validation and test set as well as the scaling function and the functions needed to save and retrieve the trained model as file.
Considering the splitting function the partitioning is done without shuffling the data because in this case we are interested in the chronological order between the data points. Anyway the data shuffling is a good practice in scenarios where we do not have this kind of concern since it helps the network to generalize the pattern to new data samples.
Scaling the input data is another important aspect because in our case the data-set consists of data points with different domains like the temperature in the continuous range [21°C, 38°C] and the state of the actuators with binary value {0,1}. Without the scaling the input data the network will give a greater importance to the features with an higher value compared with the ones with lower values, in other words some features will dominate on the others just because they have a bigger value and not because of their relevance. The output features can be scaled too, but with is usually done in order to improve the performances since it does not affect the accuracy of the network.
The tuning of the hyperparameters is the most expensive step in terms of computational effort, but it has the great advance that it can be easily parallelized since the search does not require any kind of synchronization between the searching agents. Many frameworks and GPU architectures are now designed to support this kind of optimized tuning.
In order to speed up the tuning we have used the free tier of Google Cloud platform which provides GPU resources for machine learning training applications. Another free solution is the one provided by Google Colab, a platform for sharing Jupyter Notebooks among the members of the same team based on Google Drive storage which provides GPU resources for running machine learning models. Jupyter Notebook is a web based interactive python shell which is very well suitable to testing scenarios, but it is not the best solution if the model requires an high number of resources or if the complexity of the learning system is remarkable.
The following code shows the General Model class and its common utility function inherited or overridden by its child models.

```
class GeneralModel(object):
    def __init__(self, dataset_uri, feature_cols, target_cols,
    prediction_index):
        self.feature_cols = feature_cols
        self.target_cols = target_cols
        self.prediction_index = prediction_index
        self.x_train, self.x_val, self.x_test, self.y_train, self.y_val,
    self.y_test = \
            self.split_dataset_train_val_test(dataset_uri, feature_cols,
    target_cols, prediction_index)

        self.scaler = StandardScaler()
        self.x_train, self.x_val, self.x_test = self.scale(self.x_train,
    self.x_val, self.x_test)
        self.data_set_parts = {
            'train': {
                'x': self.x_train,
                'y': self.y_train
            },
            'val': {
                'x': self.x_val,
                'y': self.y_val
            },
            'test': {
                'x': self.x_test,
                'y': self.y_test
            }
        }


    def train(self, include_val=False):
        x, y = self.x_train, self.y_train
        if include_val:
            x, y = np.concatenate((x, self.x_val), axis=0), np.concatenate
    ((y, self.y_val), axis=0)
        self.train_on(x, y)


    @abstractmethod
    def train_on(self, x, y):
        pass


    def predict(self, x, scale=False):
        if scale:
            x = self.scaler.transform(x)
        return self.predict_on(x)


    @abstractmethod
    def predict_on(self, x):
        pass



    @abstractmethod
    def get_restoring_path(self):
        pass


```

```python
    def split_dataset_train_val_test(self, dataset_uri, feature_cols, target_cols, prediction_index):
        x, y = handle_data(dataset_uri, feature_cols, target_cols, prediction_index, as_numpy=True)
        x_train_val, x_test, y_train_val, y_test = train_test_split(x, y, test_size=0.2, shuffle=False)
        x_train, x_val, y_train, y_val = train_test_split(
            x_train_val,
            y_train_val,
            test_size=0.125,
            shuffle=False
        )
        return x_train, x_val, x_test, y_train, y_val, y_test


    def scale(self, train, val, test):
        self.scaler.fit(train)
        train = self.scaler.transform(train)
        val = self.scaler.transform(val)
        test = self.scaler.transform(test)
        return train, val, test


    def evaluate(self, dataset_part, flat=True):
        x, y = self.data_set_parts[dataset_part]['x'], self.data_set_parts[dataset_part]['y']
        y_hat = self.predict(x)
        return mean_squared_error(y, y_hat, multioutput='raw_values')
```
Listing 17: General Model class

The Hyperparameter searcher evaluates the best model by calculating the summation of the mean squared errors. After the tuning the best model is then trained once again using both the trainig and the validation set. In fact, since the best hyperparameters have already been found there is no need to keep some data apart for validation. It has to be noticed that depending on the model this procedure has not always led to better results, sometimes the best model achieved a better accuracy just using the training set without merging the data with the validation set.

```python
class HyperParameterSearcher(object):
    def __init__(self, algorithm_class, config):
        self.config = config
        self.algorithm_class = algorithm_class
        self.best = {
        # it refers to validation
            'model': None,
            'error': list()
        }

    def update_bests(self, val_error, knn_config):
        if self.best['model'] is None or sum(self.best['error']) > sum(
    val_error):
            self.best['model'] = knn_config
            self.best['error'] = val_error


    def create_config_conbinations_as_list_of_dicts(self):
        parameters = self.config.keys()
        ll = list()
        for key in parameters:
            if type(self.config[key])!=list:
                p_list = list()
                p_list.append(self.config[key])
            else:
                p_list = self.config[key]
            ll.append(p_list)
        permutations = list (itertools.product(*ll))
        ld = list()
        for p in permutations:
            d = {}
            index = 0
            for k in parameters:
                d[k] = p[index]
                index += 1
            ld.append(d)

        return ld


    def search(self, mode='only best'):
        configurations = self.create_config_conbinations_as_list_of_dicts()
        for config_i in configurations:
            model_config = self.algorithm_class(
                **config_i
            )
            model_config.train()
            val_error = model_config.evaluate(dataset_part='val')
            if (mode == 'only best'):
                self.update_bests(val_error, model_config)
            elif (mode == 'all models'):
                pass


    def get_best(self):
        return self.best


    def search_knn():
        from models.knn.knn import KNNConfig
        dataset_uri = os.path.join(settings.PROJECT_ROOT_ADDRESS, "data/2
    _5_2020_random_actions_1h_every_60s.csv")
        n_neighbors = [3, 11, 21]
        feature_cols = [settings.INPUT_FEATURE_NAMES]
        if 'TIME' in settings.INPUT_FEATURE_NAMES:
            feature_cols[0].remove('TIME')
        target_cols = [settings.TARGET_FEATURE_NAMES]
        if 'TIME' in settings.TARGET_FEATURE_NAMES:
            target_cols[0].remove('TIME')
        distance_metrics = ['euclidean','manhattan']
        prediction_index = [6,7,8,9,10]
        common_config = {
            'dataset_uri': dataset_uri,
            'n_neighbors': n_neighbors,
            'feature_cols': feature_cols,
            'target_cols': target_cols,
            'distance_metric': distance_metrics,
            'prediction_index': prediction_index
        }
        knn_searcher = HyperParameterSearcher(KNNConfig, common_config)
        knn_searcher.search()
        bests = knn_searcher.get_best()
        print(bests['model'].get_hyperparameters())
        print('[TEST error - without validation]',bests['model'].evaluate(
    dataset_part='test'))
        bests['model'].train(include_val=True)
        print('[TEST error - with validation]',bests['model'].evaluate(
    dataset_part='test'))
        bests['model'].save()
        path = bests['model'].get_restoring_path()
        knn_config = KNNConfig.load(path)
        print('[TEST error - loaded model]',knn_config.evaluate(
    dataset_part='test'))


    def search_svr():
        from models.svm.svm import SVRConfig
        dataset_uri = os.path.join(settings.PROJECT_ROOT_ADDRESS, "data/2
    _5_2020_random_actions_1h_every_60s.csv")
        feature_cols = [settings.INPUT_FEATURE_NAMES]
        if 'TIME' in settings.INPUT_FEATURE_NAMES:
            feature_cols[0].remove('TIME')
        target_cols = [settings.TARGET_FEATURE_NAMES]
        if 'TIME' in settings.TARGET_FEATURE_NAMES:
            target_cols[0].remove('TIME')
        prediction_index = [6]
        common_config = {
            'dataset_uri': dataset_uri,
            'feature_cols': feature_cols,
            'target_cols': target_cols,
            'prediction_index': prediction_index
        }
        svr_searcher = HyperParameterSearcher(SVRConfig, common_config)
        svr_searcher.search()
        bests = svr_searcher.get_best()
        print(bests['model'].get_hyperparameters())
        print('[TEST error - without validation]', bests['model'].evaluate(
    dataset_part='test'))
        bests['model'].train(include_val=True)
        print('[TEST error - with validation]', bests['model'].evaluate(
    dataset_part='test'))
        bests['model'].save()
        path = bests['model'].get_restoring_path()
        knn_config = SVRConfig.load(path)
        print('[TEST error - loaded model]', knn_config.evaluate(
    dataset_part='test'))


    def search_dnn():
        from keras.losses import mean_squared_error, huber_loss
        from models.dnn.dnn import DNNConfig
        dataset_uri = os.path.join(settings.PROJECT_ROOT_ADDRESS, "data/2
    _5_2020_random_actions_1h_every_60s.csv")
        feature_cols = [settings.INPUT_FEATURE_NAMES]
        if 'TIME' in settings.INPUT_FEATURE_NAMES:
            feature_cols[0].remove('TIME')
        target_cols = [settings.TARGET_FEATURE_NAMES]
        if 'TIME' in settings.TARGET_FEATURE_NAMES:
            target_cols[0].remove('TIME')
        prediction_index = [6]
        loss_function = [mean_squared_error, huber_loss]
        n_layers = [1, 2, 3, 4, 5]
        n_epochs = [100, 300, 500]
        common_config = {
            'dataset_uri': dataset_uri,
            'feature_cols': feature_cols,
            'target_cols': target_cols,
            'loss_function': loss_function,
            'n_layers': n_layers,
            'prediction_index': prediction_index,
            'n_epochs': n_epochs
        }
        nn_searcher = HyperParameterSearcher(DNNConfig, common_config)
        nn_searcher.search()
        bests = nn_searcher.get_best()
        print('BEST PARAMS')
        print(bests['model'].get_hyperparameters())
        bests['model'].save()
        path = bests['model'].get_restoring_path()
        nn_config = DNNConfig.load(path)

        nn_config.train(include_val=True)
        print('[TEST error - without validation]', bests['model'].evaluate(
    dataset_part='test'))
        print('[TEST error - with validation]', nn_config.evaluate(
    dataset_part='test'))

    def search_lstm():
        from keras.losses import mean_squared_error, huber_loss
        from models.lstm.lstm import LSTMConfig
        dataset_uri = os.path.join(settings.PROJECT_ROOT_ADDRESS, "data/2
    _5_2020_random_actions_1h_every_60s.csv")
        feature_cols = [settings.INPUT_FEATURE_NAMES]
        if 'TIME' in settings.INPUT_FEATURE_NAMES:
            feature_cols[0].remove('TIME')
        target_cols = [settings.TARGET_FEATURE_NAMES]
        if 'TIME' in settings.TARGET_FEATURE_NAMES:
            target_cols[0].remove('TIME')
        prediction_index = [6]
        loss_function = [mean_squared_error, huber_loss]
        n_layers = [1, 2, 3, 4, 5]
        n_epochs = [100, 300, 500]
        n_neurons = [4,16]
        common_config = {
            'dataset_uri': dataset_uri,
            'feature_cols': feature_cols,
            'target_cols': target_cols,
            'loss_function': loss_function,
            'n_layers': n_layers,
            'prediction_index': prediction_index,
            'n_epochs': n_epochs,
            'n_neurons': n_neurons
        }
        lstm_searcher = HyperParameterSearcher(LSTMConfig, common_config)
        lstm_searcher.search()
        bests = lstm_searcher.get_best()
        print('BEST PARAMS')
        print(bests['model'].get_hyperparameters())
        bests['model'].save()
        path = bests['model'].get_restoring_path()
        lstm_saved = LSTMConfig.load(path)
        lstm_saved.train(include_val=True)
        print('[TEST error - without validation]', bests['model'].evaluate(
    dataset_part='test'))
        print('[TEST error - with validation]', lstm_saved.evaluate(
    dataset_part='test'))
```

Listing 18: Hyperparameter searcher class

4.10.1 Keras Tuner

I want to mention this quite new tool released the fall 2019 because it is opinion of the author that thanks to the open source community it will soon became a well known practise as well as today's ML framework such as Tensorflow and Pytorch.
Keras Tuner is an easy-to-use, distributable hyperparameter optimization framework that solves the pain points of performing a hyperparameter search. Keras Tuner makes it easy to define a search space and leverage included algorithms to find the best hyperparameter values. Keras Tuner comes with Bayesian Optimization, Hyperband, and Random Search algorithms built-in, and is also designed to be easy for researchers to extend in order to experiment with new search algorithms. It is not just compliant with Keras, but it can be set to search sci-kit parameters as well. Compared with the hyperparameter searcher we have manually developed it has as main advantage the fact that it can perform not only grid search but also other adaptive and optimized searching algothms.
The only requirement is to define the model as a function which takes as a given parameters the Keras Tuner object. Then instead of defining the hyperparameters as single values it will be possible to define a search space which will be used by the tuner to find the best values. The tuner object requires a search algorithm as input parameter and the maximum number of epochs available to complete the search. The following is a brief example of Keras Tuner usage with some sci-kit machine learning models.

```python
from sklearn import ensemble
from sklearn import linear_model

def build_model(hp):
    model_type = hp.Choice('model_type', ['random_forest', 'ridge'])
    if model_type == 'random_forest':
        with hp.conditional_scope('model_type', 'random_forest'):
            model = ensemble.RandomForestClassifier(
                n_estimators=hp.Int('n_estimators', 10, 50, step=10),
                max_depth=hp.Int('max_depth', 3, 10))
    elif model_type == 'ridge':
        with hp.conditional_scope('model_type', 'ridge'):
            model = linear_model.RidgeClassifier(
                alpha=hp.Float('alpha', 1e-3, 1, sampling='log'))
    else:
        raise ValueError('Unrecognized model_type')
    return model

tuner = kt.tuners.Sklearn(
        oracle=kt.oracles.BayesianOptimization(
            objective=kt.Objective('score', 'max'),
            max_trials=10),
        hypermodel=build_model)
X, y = ...
tuner.search(X, y)
```

Listing 19: Keras Tuner example

4.11 Models evaluation

For each model implemented we considered both the MSE and the accuracy. Regarding the latter we defined a tolerance range to validate the predicted value in such a way that a score variable has been increased every time the predicted value was inside the tolerance range. Since we were predicting temperature and the accuracy of the available sensors is 1°C the tolerance range for each actual target value y_i has be set to $[y_i - 1, y_i + 1]$.
With smaller datasets consisting about of 300 records we identified DNN as the model with the higher accuracy which achieved about 89% as accuracy score. It has been followed by the KNN model with about 70% accuracy score and then by LSTM with 60%. The model with the lowest accuracy has been SVR which achieved about 20% as accuracy score.
We then decided to increase the size of the dataset to find out if LSTM would achieved better results. In fact, since LSTM perform the learning process on sequences instead on single values we expected to reach its maximum potential with huge amount of data needed to be processed into sequences.
With about 4000 records the best models is KNN which achieved about 95% as accuracy score while SVR is still the worst by barely reaching 20%. While DNN performs almost the same, LSTM improves its score by reaching 87% accuracy as expected.
The following table reports the scores in terms of accuracy achieved by each model differentiating also each of them by their score obtained by training on just training set and then by training on validation and training set merged.

Model	Dataset size: 300 records	Dataset size: 4000 records
KNN	70% and 71% (with val)	95% and 96% (with val)
SVR	42% and 47% (with val)	22% and 19% (with val)
DNN	79% and 63% (with val)	89% and 88% (with val)
LSTM	64% and 27% (with val)	87% and 85% (with val)

It is reasonable to predict that increasing even more the size of the dataset the LSTM model will perform better while the accuracy score of the others will decrease.

4.12 Actions planner design

The final step we had accomplished is to use the best model to select the best actions to perform in order to reach a target temperature starting from an initial state.
This task is a tree search problem, because for each time interval the predictor model has to be used to select the best action that minimizes the heuristic function $h(\Delta T, E)$. The heuristic function takes into account the difference ΔT between the actual temperature reached by the planner and the target temperature and the energy consumed by the inside actuators E. These two factors should be weighted in order to prioritize the satisfaction of one of these on the other. The mathematical representation of the heuristic function is:

$$h(\Delta T, E) = min(w_1 \cdot \Delta T + w_2 \cdot E) \tag{18}$$

So in order to explore the search space we identified the following possible approaches:

- Breadth-first search
- Depth-first search
- Hybrid search
- Reinforcement learning

Due to the fact that the performable actions by the ScaledHome system have a high dimensional space, the selection of the best actions has been quite challenging. In fact, even considering the lamp and the external fan as not part of the system as well as the window of the Living room and the Ac placed inside the room as the same actuator, the number of different binary permutations is equal to 2^{16}. That means that for each level inside the search tree more than 65k combination of performable action will have to be taken into account, leading to a corresponding number of prediction and heuristic evaluation of the next state.
This became even more complex if we have to perform a depth-first search or an hybrid search instead of a breadth-search because sequences of predictions will have to be chained in order to find the best actions to perform given the state at time i. Furthermore, the chaining approach will lead to a diverging trend comparing it to the actual temperature trend. This is due to the fact that the error of each prediction will be summed up with the error of the other layers and even if the errors will partially compensate each other in general terms this will effect badly the prediction accuracy.
In this kind of scenario where the search space is remarkable a reinforcement learning approach is well suitable because the agent can explore the search space in an autonomous way even with zero a priori knowledge. As mentioned in the previous chapters in this case there is no need to train the model because it will learn by rewards and punishments based on the heuristic function the best way to reach the target.

The planner has to interact with the Home Controller to obtain the current state of the house and choose the best array of action to perform while the Environment simulation agent provides the actions needed to simulate the real environment.
The following image shows the resulting architecture of the ScaledHome system focusing on the interaction between the Home Controller and the simulation agents.

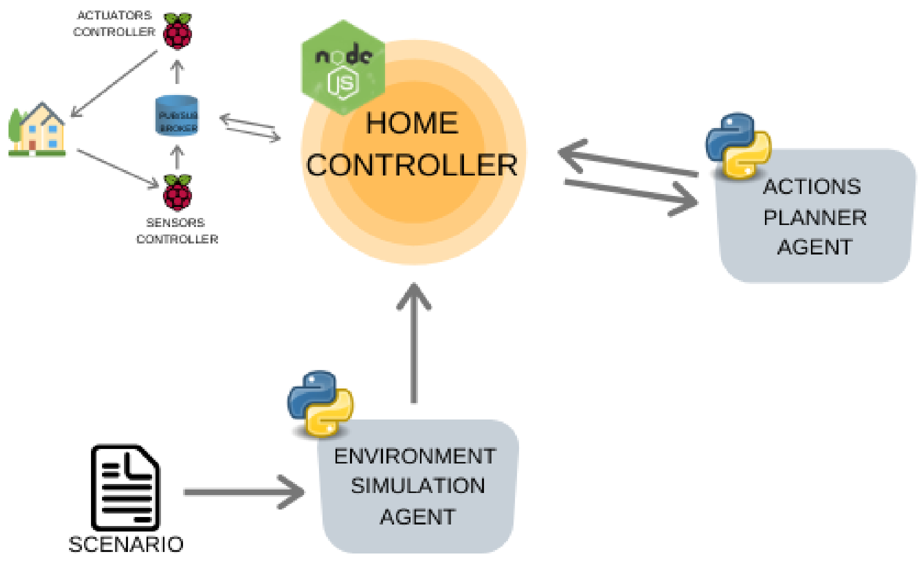


Figure 44: Home Controller interaction with simulation agents

Once the action planner obtains the current state it predicts the next state with the exploring strategies discussed in the previous section and selects the one that minimizes the working time of the actuators while reaching the closest temperature to the target. Of course this implies that it has to keep track of the working time of the AC and the heater placed inside the living room for each new action performed.

4.13 Achieved results

The resulting system is an heterogeneous structure compose by different agents where each of them has a well defined duty and there are no shared responsibilities.
The resulting architecture can be easily extended by adding new entities with the minimal effort. The system is meant to interact remotely with the ScaledHome simulation environment in order to stimulate third parties to collaborate and then improve the quality of the project.
The system has been used to collect data and then recreate a real world scenario inside the simulation environment taking into account and solving many scaling aspects.
Then the data collected has been used to train several machine learning models which after being tuned have been used to predict the consequences of the actions performed by the planner.
This has enabled the action planner agent to explore the space of the possible actions and then to provide at each time interval the best actions to perform. This is meant to reach the desired house state while saving energy by using as less as possible the inside actuators.

4.14 Possible extensions

The following are some of the most reasonable extensions that can be introduced inside the system, it is needed to stress that these changes will not lead to an essential modification of the architecture since flexibility of the system has been one of the main requirements.

4.14.1 Parallel hyperparameter tuning

Since Kubernates eases the managing of complex and distributed architectures it would be interesting to deploy all the entities of the ScaledHome system as containers in order to manage them from a single orchestrator. In this case the tuning of the hyperparameters could be parallelized in order to increase the training speed of the models. We identified Kubeflow, a Kubernates version meant to ease the deployment of ML applications, as a very promising tool to implement.

4.14.2 Prediction chaining

Instead of considering just the actions at the current level of the search tree the Action planner could explore in a more optimized way the search space with an hybrid approach. This will most likely decrease the time needed by the ScaledHome system to reach the target and consequently reducing the energy consumption.
Furthermore, a reinforcement learning approach is very promising and it could lead to better tree search performances since the dimension of the space is well suitable to an autonomous exploration technique.

4.14.3 Context forecasting

The Action Planner could be integrated in such a way that it is able to gather information about the weather forecasting providers in order to take into account the predicted outside metrics and save even more energy. For example if the weather forcasting about the next few hours predicts a temperature increase it could be possible to open the windows of the house instead of turning on the heater to save energy and increase the inside temperature.

Conclusion

Nowadays residential houses accounts a relevant amount in terms of energy consumption. This is due to the fact that they are not mere buildings any more since they are composed by several smart agents able to interact with the outside environments and work autonomously in order to reach a target state in a collaborative way.
Actuators and sensors can be designed in order to communicate each other though a centralized infrastructure in charge of collecting data and identifying the best actions needed to satisfy the requirements desired by the inhabitants of the house.
The resulting system would be able to collect data and perform autonomous action by taking into account the energy usage in order to reach the target state.
A scaled model approach will benefit from lower costs and the possibility to simulate different real world scenarios inside the same simulation system.
The work done has given a relevant improvement to the smart home area providing an high scalable and distributed simulation system which allows different teams to use it in order to collect data and run different simulations.
Thanks to the SOA approach adopted the ScaledHome system is capable of simulating custom outside environments and to react to them in order to reach the house state desired by its inhabitants.
The flexibility of the system enables the integration of additional sensors, actuators and business logic with the minimal effort by providing an high level interface to interact with. In fact, since the interaction between the agents is handled in a dynamic way by well defined entities this can be done without changing the underlying system infrastructure.
Several machine learning models has been implemented and compared to provide a practical application of the data collected and the consequent evaluation has led to select among them the most suitable to the specif simulation scenario.
The whole system has been developed taking into account some of the main software design principles applying them to a specific and heterogeneous scenario where many different technologies have been used to handle a set of different agents which interact each other in order to satisfy the common requirements. The development process is compliant to the software engineering pipeline since it follows the recommended procedure which leads the project from the requirements analysis to the final product.
The work done is a well balanced achievement between the software engineering principles, the machine learning concepts and their combined application in a real research environment.
The obtained result represents a great achievement for the host team since it provides a flexible architecture that can be used for many different purposes related to the simulation aspects in a smart home system.
Since the important outcomes achieved in both the distributed system and machine learning areas a corresponding paper is going to submitted to the MSWiM 2020 conference within the next couple of months.
The skills learnt by the candidate and the resulting product are an irrefutable proof of how the methods and the theory studied during these academic years can be merged and applied not only to example application, but to real user cases meant to improve the quality of our society.
Without any doubt the quality of the project is due to the fact that different backgrounds coming from the Italian and American sides have been used and applied together to reach a common goal.

References

[1] U.S. Energy Information Administration statistics.

[2] Amir Mosavi. Energy consumption prediction using machine learning, a review. 2019.

[3] A. Anvari-Moghaddam. Optimal smart home energy management considering energy saving and a comfortable lifestyle. pages 1–1, 2016.

[4] Joonseok Park. CASS: A Context-Aware Simulation System for Smart Home. pages 461–467, 2007.

[5] W. Lee, S. Cho, P. Chu, H. Vu, Sumi Helal, W. Song, Y.-S. Jeong, and K. Cho. Automatic agent generation for iot-based smart house simulator. *Neurocomputing*, 209:14–24, 10 2016.

[6] Dan Gessler. Mississipi river scaled model.

[7] Laboratory for Advanced Software Systems. UMassTraceRepository.

[8] N. Lima B. Jobson M. Kirk P. OKeeffe S. Pressley V. Walden B. Lamb B. Lin, Y. Huangfu and D. Cook. Automatic agent generation for iot-based smart house simulator. *Journal of Sensor and Actuator Networks*, 6:13, 2017.

[9] S. Ahmad W. Jin, I. Ullah and D. Kim. Occupant comfort management based on energy optimization using an environment prediction model in smart homes. *Sustainability*, 11:997, 2019.

[10] Fernando F. Mateo, Juan José Carrasco, Abderrahim Sellami, Mónica Millán-Giraldo, Manuel Domínguez, and Emilio Olivas. Machine learning methods to forecast temperature in buildings. *Expert Systems with Applications*, 40:1061–1068, 03 2013.

[11] Dong Chen and David E. Irwin. Weatherman: Exposing weather-based privacy threats in big energy data. *2017 IEEE International Conference on Big Data (Big Data)*, pages 1079–1086, 2017.

[12] Tobias T. Teich, Falko Roessler, Daniel Kretz, and Susan Franke. Design of a prototype neural network for smart homes and energy efficiency. *Procedia Engineering*, 69:603–608, 12 2014.

[13] Eunju E. Kim, Sumi Helal, and Diane Cook. Human activity recognition and pattern discovery. *IEEE pervasive computing / IEEE Computer Society [and] IEEE Communications Society*, 9:48, 01 2010.

[14] D. J. Cook, M. Youngblood, E. O. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja. Mavhome: an agent-based smart home. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003).*, pages 521–524, March 2003.

[15] R. Fritz and D. J. Cook. Identifying varying health states in smart home sensor data : An expert-guided approach. 2017.

[16] Ane Alberdi, Alyssa Weakley, Maureen Schmitter-Edgecombe, Diane J Cook, Asier Aztiria, Adrian Basarab, and Maitane Barrenechea. Smart home-based prediction of multidomain symptoms related to alzheimer's disease. *IEEE journal of biomedical and health informatics*, 22(6):1720—1731, November 2018.

[17] Jessamyn J. Dahmen, Brian L. Thomas, Diane J. Cook, and Xiaobo Wang. Activity learning as a foundation for security monitoring in smart homes. *Sensors*, 17(4), 2017.

[18] T. Boubez C. Brown D. Chappell J. deVadoss T. Erl N. Josuttis D. Krafzig M. Little B. Loesgen A. Thomas Manes J. McKendrick S. Ross-Talbot S. Tilkov C. Utschig-Utschig H. Wilhelmsen A. Arsanjani, G. Booch. SOA manifesto.

[19] Letha Hughes. Etzkorn. *Introduction to Middleware : Web Services, Object Components, and Cloud Computing.*

[20] mqtt.org.

[21] cloudmqtt.com.

[22] Ralph Johnson John Vlissides Erich Gamma, Richard Helm. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.

[23] Damla Turgut. CPS: Medium: ScaledHome - a physical testbed for the intelligent and controllable home of the future. 2019.

[24] Nodejs framework.

[25] ARPA. mi_meteo_8162.csv. In *Milano Weather Station Data*. Harvard Dataverse, 2015.

Acknowledgements

I have to start by thanking both Prof. Paolo Bellavista and Prof. Damla Turgut for the beautiful opportunity that I received with this academic experience outside my native country.
The exciting collaboration with the host team has improved not only my technical skills, but also my personal understanding of the academic world.
My gratitude goes to each member of Prof. Turgut's team for their availability and professional support during the development of the project.
In particular I want to thank Prof. Ladislau Boloni who has followed the progress of the project providing helpful and valuable indications about the best choices to take in order to improve the quality of the final result.
A special acknowledgement goes to both Siavash Khodadadeh and to Salih Safa Bacanli for their essential help in the implementation of the machine learning models and the networking structure.
Finally yet importantly I have to mention the help given by Hassam Ullah Sheikh who has introduced to me the main theoretical concepts about the machine learning world.