POLITECNICO DI TORINO

Master's Degree In Mechanical Engineering

Master's Degree Thesis

Simulating an air conditioning system with object-oriented modeling approach using Modelica on Wolfram SystemModeler

Supervisors

Prof. MATTEO FASANO

Engr. JAN BRUGÅRD

Engr. ANKIT ANURAG NAIK

Candidate

LUIGI AUGELLO

April 2021

Summary

The main purpose of the work carried out in this thesis is the use of the object-oriented approach for air conditioning systems modeling, including all the equipment that compose them, relatively to a residential apartment through the usage of the modeling environment Wolfram SystemModeler.

After a short digression about the different approaches useful to model such systems and the presentation of some conducted works in the thermal physics field already existing in literature, the general concepts behind Modelica language, with which SystemModeler works, are exposed.

By the application of such modeling tool, a case study concerning the heating of a building using a heat pump-to-fan coils system is implemented, through which is possible focusing the attention on all the theoretical aspects related to thermal and physical processes that characterize both its behaviour and operations, with the further aim to confirm the high convenience on the usage of such modeling approach thanks to the countless benefits that it is able to offer.

Exploiting the simulation environment Simulation Center available on SystemModeler, several simulations are done to test the models developed, allowing in addition to put forward some consideration on their adaptability to different operating conditions.

The following work has been carried out thanks to Wolfram's team who let me use their software during my internship at their office in Linköping, Sweden.

ACKNOWLEDGMENTS

Finally I am arrived to the finish line; I am delighted and excited on thinking about all the moments spent to reach it, beautiful and others less beautiful, and I intend to express gratitude to all those people who made it unforgettable and less tiring thanks to their presence. First of all my family and my parents, who have always supported me in my every choice and who have always trusted me, supporting me in everything, especially economically, for the realization of this dream. I have been lucky to have parents who are aware of my love for adventures and my openness to the world and who have never limited me in anything. A special thanks also to my little sister Martina, who has been invaluable in drafting my thesis thanks to her ability on technical drawing. Subsequently it is a must to say thanks all the roommates I have had over time: Stefano, Vito, Giuseppe, Gero and Manlio, all my companions they have accompanied me throughout my journey thanks to their support and friendship; we shared home, joys and sorrows and above all our time together. I had a wonderful time with them and I hope to always keep them in mind and in my heart. Of vital importance to me has been the group of colleagues, friends and brothers that we have created and who has represented my second family. It is impossible to mention them one by one, but surely those who are among them know they are part of it. Thanks friends, without you it wouldn't have been the same. Surely thanks to all my childhood friends, who always have been with me along this long trip and who never have abandoned me even if I has been away form my hometown for six years and now are ready to rejoice with me this incredible goal. A special thanks goes to my girlfriend Rosa who endured and supported me on my journey, who she was able to both console me when it was needed and rejoice in our victories. She gave me so much determination and the strength to not give up when I was almost ready to do so, and for that I can never thank her enough. Finally, I feel extremely proud and lucky to have finished my university career by submitting a thesis written in collaboration with Wolfram Mathcore following my internship at their offices in Linköping, Sweden. This is why I would like to thank the Prof. Matteo Fasano in the first place, without whom this experience could not have taken place and Jan, Ankit, Jorge and all the other colleagues who have allowed me to work with them this fall and have supported me in my thesis activity. It was an unforgettable experience where I had the opportunity to get involved and discover new places and people and that changed me for the better. Thanks!

«What you get by achieving your goals is not as important as what you become by achieving your goals.» Zig Ziglar

Table of Contents

Acronyms

AI	Artificial Intelligence
WSM	Wolfram SystemModeler
OOM	Object Oriented Modeling
AC	Air Conditioning
NTU	Number of Transfered Unit

Chapter 1

Introduction

1.1 Modeling an air conditioning system

The presence and proper sizing of a heating and air-conditioning system for a residential building is essential to ensure an adequate thermal comfort to the occupants. Furthermore, being such a system one of the major sources of energy consumption in a building [1], a careful design of the system allows to achieve excellent results in terms of reducing the entity of energy expenditure, with a look to the environment protection. The energy requirement of an HVAC system is tightly related to several factors; the environment climate condition where the building is located, the inner temperature value that must be maintained, the inner occupancy and, mostly, the thermal features of the building which define the way in which it exchanges heat with the external environment. Ignoring these factors, particularly how they affect each other, might lead to plant inefficiencies and, moreover, to an excessive waste of energy.

Modeling and simulation of such systems play a crucial role in the optimal design of the various equipment that compose them. The application of modeling tools that allow to dynamically model, from a thermal point of view, the building and all the components of an AC system, provides the possibility to develop sophisticated control and parameter optimization techniques able to raise the level of the whole system efficiency, as well as its predisposition to energy-sustainability.

1.2 Different modeling approaches

The importance of ensuring certain performance, dictated by an even more sophisticated technological innovation and the increasing demand of a suitable thermal comfort, have allowed the development of a large number of modeling tool for an AC system that were capable to take into account the whole factors that affect its operation. A review on the latter is given in [2]. Modeling a heating system, as shown in figure 1.1, is made difficult by the many interactions between components belonging to different domains (thermodynamic, fluid-mechanic, electrical, control system, etc.) along with the need to handle a mix of ordinary differential equations (ODE), partial differential equations (PDE) and differential algebraic equations (DAE) [3].

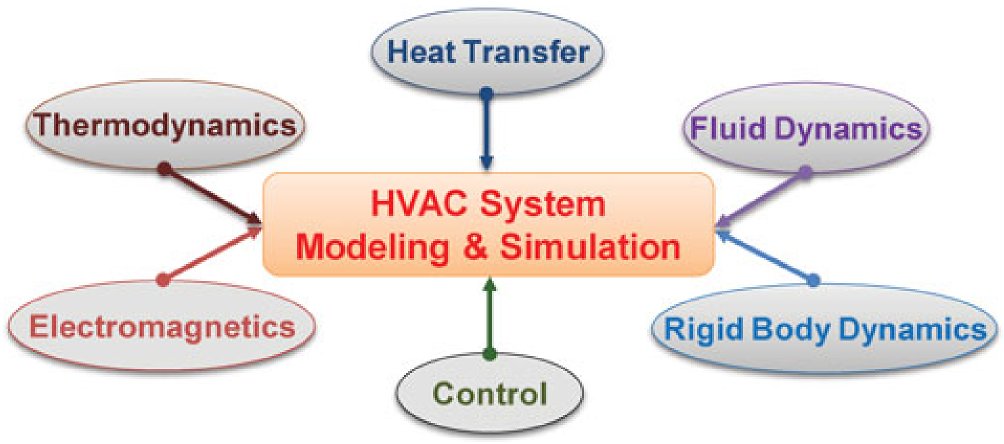


Figure 1.1: Several domains involved in the modeling of an AC system. Image taken from [3]

Traditional modeling and simulating programs are based on a language (FORTRAN, C, C++) characterized by the production of programming codes in which the sequences of instructions assign values to the variables through a precise execution order [4]. This leads to input-output systems where it is essential to manipulate the physical model equations diversifying the known variables (input) from those that must be computed (output). Such an approach provides complex models and therefore hard to handle for a hypothetical user that is not the programmer, but above all poorly interfaceable with other models. As explained by Kopei et al. [5], in the case of a *casual* modeling approach, a hypothetical component receives the x value as input, executes on it a specific mathematical operation represented by $f(x)$ and gives back the y result value as output. This involves the utilization of an *imperative* programming language, that means assign the value of the expression $f(x)$ to the y variable in a unique way. Differently, with an *a-casual* modeling approach, the common signal of two linked components may be passed in two different directions. Here, in the expression $f(x)$, the unknown variable may be both x and y. Such variables represent physical quantities and $y = f(x)$ is the law which describes the existing bond among them. This allows us to focus on the physical formulation of the system that needs to be studied, especially on the rules and processes characterizing it, rather than on the task to develop an algorithm for its resolution, minimizing the possibility to introduce errors.

Thus, equation-based and object-oriented modeling approach has been developed and widely used. Thanks to the countless advantages it offers, like modularity, models reusability, great flexibility of use and many other, this approach lends itself perfectly with the challenge to dynamically model a complex system, like an AC system.

1.3 Use of object-oriented modeling approach in AC systems: state of art

The OOM approach represents an excellent tool for engineering systems modeling. Its uniqueness lies in the possibility to break down a complex model into simpler sub-components. Each of these is mathematically described by a set of equations (differential, algebraic) that define its behaviour. Specific ports, called pins, allow the connection of the components, thus enabling key variables interact. The possibility to create new model is one of the main features of OOM approach, based on the task it should carry out, using component already exist as well and coming from different domains in order to create multi-domain models. Many languages based on OOM approach have already been developed for the simulation of complex system, such as: APMonitor, ASCEND, Modelica etc. [5]. Among these, Modelica [6] has been the utilized one in this thesis. For its detailed description of how it works, please refer to the next chapter.

In the area of AC system modeling, this approach has already largely used. Of considerable interest is the work done by Abed Al Waheed Hawila et al. [7], where a radiant floor heating system is modeled through Modelica language and using as simulation environment the well-known Dymola [8]. Figure 1.2 shows a schematic example of how the system should looks like. The top model is divided into three sub-models: the hydronic circuit, the heating system (heat pump) and, obviously, the radiant floor system. The application of radiant panels as emission devices is favoured by the excellent characteristics they have in terms of available space, high efficiency and ease of control. The dynamic model is validated by comparing the result obtained to those from experimental activity they carried out, proving ad high reproduction fidelity.

A further example available in literature of how object-oriented modeling may be used for the study of a heating system is represented by the activity proposed by R. Franke [9], where is studied a solar heating system taking as example the existing plant located in Särö in the south of Sweden. The intent is to demonstrate how the OOM approach may be used for «solving other engineering tasks such as parameter identification and system optimization». Even then, the overall model consists of the interaction of three sub-models: a load sub-system, a solar collector and a heat storage. All the components are modeled through Omola and simulated in OmSim environment [9]. The author emphasized how the symbolic system descriptions, another distinctive feature of the OOM approach, can be used in order to develop system optimization criteria, especially for the expressive flexibility that distinguishes Omola and OmSim.

Crucial is the evaluation of the thermal behaviour of the building that must be equipped by an AC system to determine the amount of thermal energy it needs. In this regard Felix Felgner et al. [10], by exploiting OOM approach advantage that it offers to focus attention on the laws that rule thermal process of heat exchange in a building, rather than on the solution algorithm implementation, built and simulated thermal dynamics of a building investigate some physical parameters and to develop logical control techniques to make the system as efficient as possible, claiming also that «the benefit of a well-organized modularity is a high degree of component reuse and clarity in the model structure makes modelling efficient and safe, even to model-users who are not familiar with the relevant physics and control theory in detail». In addition to the modeling of the building under the thermal point of view useful for accurately reproducing its behaviour, in their article F. Felgner et al. emphasize the significance of taking in account, while modeling an AC system, the influence of atmospheric agents realizing a model consisting of several algorithms able to consider the impact of solar radiation on the system behaviour [10]. Also interesting is the manner in which are modeled the pipes of the heating system; a diagram view of the model is presented in figure 1.3, where it is possible to appreciate how, following the logic of modularity that the object-oriented modeling approach possesses, all the components are connected

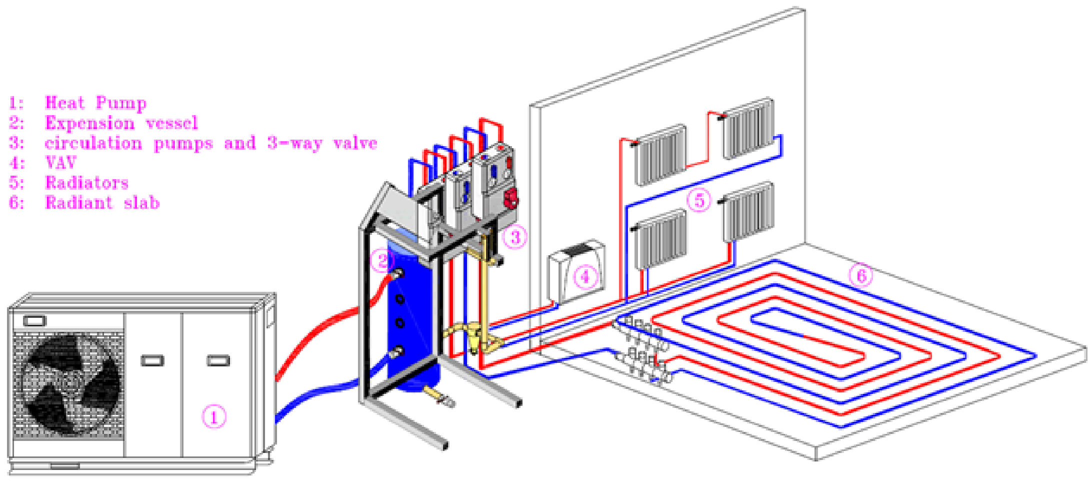


Figure 1.2: Radiant floor system layout. Image taken from [7]

each other through connectors, thereby creating a hierarchical model of a water pipe. [10]

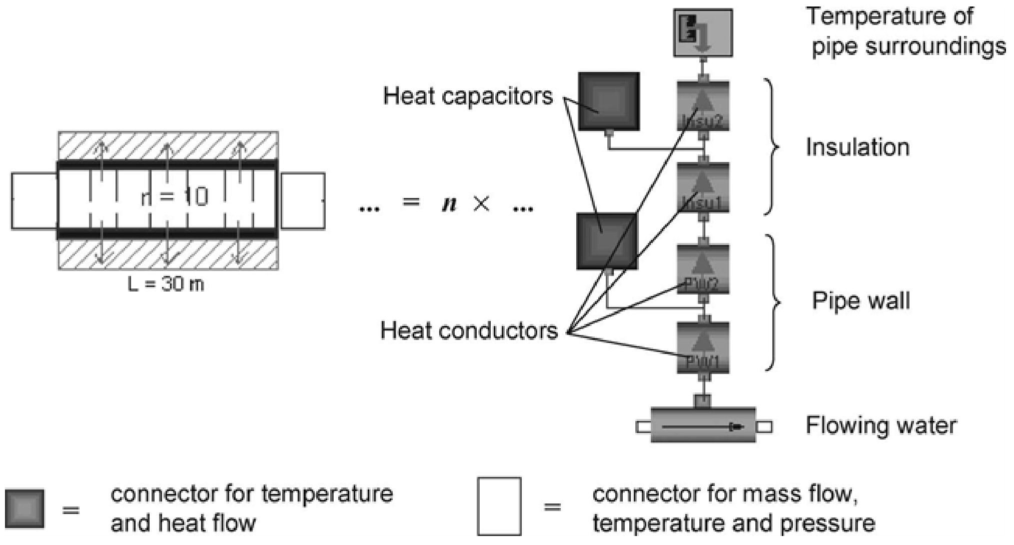

Temperature of pipe surroundings
Heat capacitors
Insulation
Insu2
Insu1
n = 10
... = n × ...
L = 30 m
Pipe wall
PW2
PW1
Heat conductors
Flowing water
▨ = connector for temperature and heat flow
□ = connector for mass flow, temperature and pressure

Figure 1.3: Model of a water pipe. Image taken from [10]

Among the other works carried out through OOM approach, inherent to AC system equipment, noteworthy examples may be the one of Mattson [doc all], who modeled a water-to-water heat exchanger via logarithmic mean difference technique, obtaining satisfactory results. And yet, Mortada et al. [3] studied and developed an air-to-air heat pump model, while Schulze [3] made a major contribution to the computation of the two-phases properties of coolants through real time models simulations achieved using Modelica via SimulationX [3].

1.4 Statement of work

The principal purpose of this thesis is modeling an air conditioning system for a residential apartment using the OOM approach, thus exploiting all the power which is capable to offer. For the modeling will be used the aforementioned Modelica language on which *SystemModeler* is based, an interactive graphical environment for the modeling and simulation of complex systems. Developed by Wolfram Research, it already includes a large number of ready-to-use components belonging to different engineering domains (mechanic, thermal, control system, etc.) stored into some libraries, which may be combined (connected) to each other and modified to adapt their behaviour to the situation under consideration.

In the specific case of this thesis, the air conditioning system of a building has been modeled on SystemModeler, consisting of a heat pump that supplies fan coils, selected as terminals, through a pipe network within which flows water, the heat carrier fluid. This is coupled with the dynamic thermal model of the building that allows to study how modifying some technical or external parameters may dynamically change the thermal power extent it needs; such is a key parameter for the subsequent sizing of the thermal machines. It will be moreover underlined the extreme flexibility such model shows allowing to rearrange the components that make up the building model in several different configurations, compared to the one covered in this work.

For the construction of all the models, components already available from CollegeThermal library developed in co-operation with Politecnico di Torino have been adopted, which represents an excellent example of how should be thermally modeled all the necessary components of a heating system. Furthermore, *ex-novo* models have been created by applying thermo-technical laws for taking account heat transfer phenomena affecting components such as heat exchangers. This will lead to a system, in order to be able to investigate the major parameters of each components in addition with the possibility to shape a control logic. Last but not least, the analysis of the model with the variations of operating conditions will allow to make judgements, for instance, about the system energy consumption, providing a solid tool for the development of a critical thought in the selection of the equipment of an air conditioning system.

Chapter 2

Methods

2.1 SystemModeler: Modelica language

Wolfram SystemModeler, developed by Wolfram Research, is an interactive graphical environment designed for computer modelling and simulation of complex and multi-level physical and other system based on the Modelica language [11]. WSM allows the user to create system models using an extensive library of ready-made physical and logical components written in the Modelica language using the drag-and-drop approach. The basis of creating models in the WSM environment is the use of the *object-oriented* Modelica language. It is a high-level declarative language for component-oriented modeling of complex systems, particularly those systems containing mechanical, electrical, electronic, hydraulic, thermal, energy components. These components can then be combined into subsystems and systems with a very complex architecture.

Modelica language is based on writing differential, algebraic, and discrete equations, instead of using assignment operations. For instance, in contrast to a typical assignment statement, such as:

$$x := 2 + y \tag{2.1}$$

where the left-hand side of the statement is assigned a value calculated from the expression on the right-hand side, an equation may have expressions on both its right- and left-hand sides, for example:

$$x + y = 3z \tag{2.2}$$

So, equations do not describe assignment but *equality* [12].

Such a modeling method does not specify a predetermined casual relationship to the calculation of input variables; in fact, it independently manipulates the equations in symbolic form, determining the order of their execution and which components in the equation will determine the input and outputs. As pointed out in [11], Modelica is often compared with object-oriented programming language, such as C++ or Java, but it differs significantly from them. Firstly, it is a *modeling* language, not a *programming* language; classes of this language are not compiled in the usual sense, but are converted into *objects*, which are subsequently used by a specialized process. Second, although classes may contain algorithmic components that are similar to operators and blocks in programming languages, their primary content is a set of equations. Thus, the program in Modelica is a system of equations.

Modelica is an ideal language for modeling the behavior of engineering systems in nearly any engineering domain being technically very capable. Through complex ready-made algorithms, Modelica compilers allow engineers to focus on the mathematical description of the behavior of high-level components. This allows users to carry out high-performance numerical experiments, without going into solving differential-algebraic equations, working numerical solvers, generating code, post-processing, etc [11].

2.2 Object-oriented approach and a-causal modeling

As stated by P. Fritzson in [13]: «object-oriented modeling languages try to separate the notion of *what* an object is from *how* its behavior is implemented and specified in detail.» The Modelica view on object-orientation in the Modelica language emphasizes structured mathematical modeling. Object-orientation is viewed as a structuring concept that is used to handle the complexity of large system descriptions. A Modelica model is primarily a declarative mathematical description, which simplifies further analysis. Dynamic system properties are expressed in a declarative way through equations. The concept of declarative programming is inspired by mathematics, where it is common to state or declare what holds, rather than giving a detailed stepwise algorithm on how to achieve the desired goal as is required when using procedural languages. This relieves the programmer from the burden of keeping track of such details. Furthermore, the code becomes more concise and easier to change without introducing errors [13].

Thus, the declarative Modelica view of object-orientation, from the point of view of object-oriented mathematical modeling, can be summarized as follows:

- Object-orientation is primarily used as a structuring concept, emphasizing the declarative structure and reuse of mathematical models. The three ways of structuring are hierarchies, component-connections, and inheritance.

- Dynamic model properties are expressed in a declarative way through equations.

- An object is a collection of instance variables and equations that share a set of data.

Acausal modeling is a declarative modeling style, meaning modeling based on equations instead of assignment statements. Equations do not specify which variables are inputs and which are outputs, whereas in assignment statements variables on the left-hand side are always outputs (results) and variables on the right-hand side are always inputs. Thus, the causality of equation-based models is unspecified and becomes fixed only when the corresponding equation systems are solved. This is called 'acausal modeling'. The term physical modeling reflects the fact that acausal modeling is very well suited for representing the physical structure of modeled systems. The main advantage with acausal modeling is that the solution direction of equations will adapt to the data flow context in which the solution is computed. The data flow context is defined by stating which variables are needed as outputs, and which are external inputs to the simulated system [13].

One of the main advantages with Modelica, thanks to the composability advantage of an acasual modeling, is the ease of constructing multi-domain models simply by connecting components from different application domain libraries.

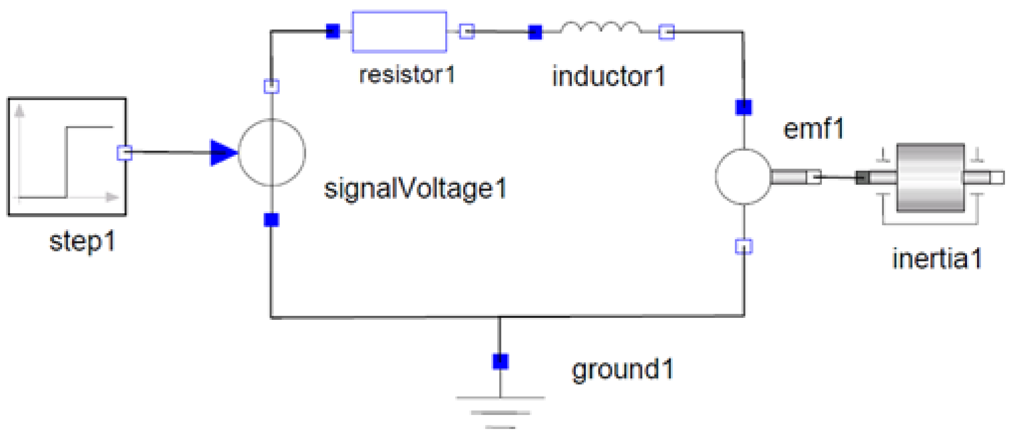


Figure 2.1: Multi domain model of a DC motor on SystemModeler.

The particular model shown in Figure 1.1 contains components from the three domains, mechanical, electrical, and signal blocks, corresponding to the libraries Modelica.Mechanics, Modelica.Electrical, and Modelica.Blocks.

2.3 Modelica operating basic concepts

The basis of modeling in the WSM environment and the fundamental structural units of modeling in the Modelica language are the *classes*, also called models [13]. From a class definition, it is possible to create any number of objects that are known as instances of that class. Classes can contain equations which provide the basis for the executable code that is used for computation in Modelica. Every object has a class that defines its data and behavior. A class has three kinds of members:

- Data variables associated with a class and its instances. Variables represent results of computations caused by solving the equations of a class together with equations from other classes. During numeric solution of time-dependent problems, the variables stores results of the solution process at the current time instant.

- Equations specify the behavior of a class. The way in which the equations interact with equations from other classes determines the solution process, i.e., program execution.

- Classes can be members of other classes.

Components are simply instances of Modelica classes. Those classes should have well-defined interfaces, sometimes called ‘ports’, in Modelica called connectors, for communication and coupling between a component and the outside world. A component is modeled independently of the environment where it is used, which is essential for its ’reusability’. This means that, in the definition of the component including its equations, only local variables and connector variables can be used. No means of communication between a component and the rest of the system, apart from going via a connector, should be allowed. A component may internally consist of other connected components, i.e., ‘hierarchical modeling’ [13].

Complex systems usually consist of large numbers of connected components, of which many components can be hierarchically decomposed into other components through several levels.

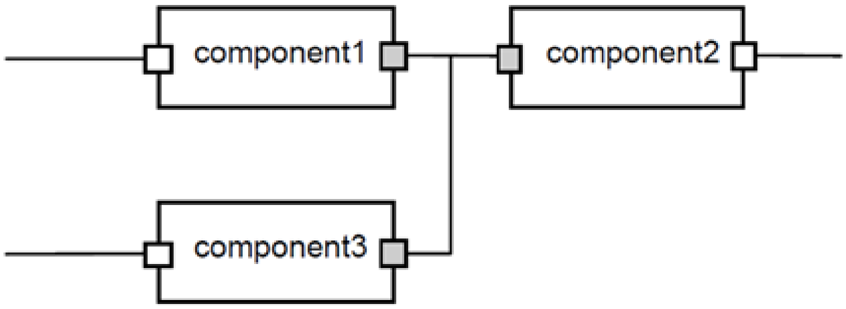


Figure 2.2: Simple components. Image taken from [13]

Each rectangle shown in Figure 2.2 represents a physical component, e.g., a resistor, a capacitor, a transistor, a valve, etc. The connections represented by lines in the diagram correspond to real, physical connections. For example, connections can be realized by electrical wires, by the mechanical connections, by pipes for fluids, by heat exchange between components, etc. The connectors, i.e., interface points, are shown as small square dots on the rectangle in the diagram. Variables at such interface points define the interaction between the component represented by the rectangle and other components [13].

Connectors are instances of connector classes, which define the variables that are part of the communication interface that is specified by a connector [13]. Thus, *a connector is a way for one model to exchange information with another model.* For example, observing the figures below, PIN is a connector class that can be used to specify the external interfaces for electrical components that have pins.

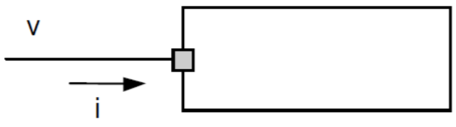


Figure 2.3: Component with electrical PIN connector. Image taken from [13]

```
connector Pin
  Voltage      v;
  flow Current i;
end Pin;
```

Figure 2.4: Modelica PIN code

Connections between components can be established between connectors of equivalent type (both `Voltage` and `Current` are `Real` in this case, but with different associated units, respectively [V] and [A]). Modelica supports equation-based acausal connections, which means that connections are realized as equations. For acausal connections, the direction of data flow in the connection need not be known. This is possible because acausal connector definitions focus on physical information exchanged, not the direction that information flows. The result is that it is possible to create component models around the idea of physical interactions without requiring any *a priori* knowledge about the nature (i.e., directionality) of the information exchange [13].

A connector class can contain two distinct classes of variables, according with the acausal approach to physical modeling:

1. **Across variables** (also called *Potential*). Differences in the values of across variables across a component are what trigger components to react. Typical examples of across variables are temperature, voltage and pressure. Differences in these quantities typically lead to dynamic behavior in the system [13].

2. **Through variables** (also called *Flow*). Flow variables normally represent the flow of some conserved quantity like mass, momentum, energy, charge, etc. These flows are usually the result of some difference in the across variables across a component model. For example, current flowing through a resistor is in response to a voltage difference across the two sides of the resistor. Modelica follows a convention that a positive value for a through variable represents flow of the conserved quantity **into** a component [13].

Two types of coupling can be established by connections depending on whether the variables in the connected connectors are nonflow (across), or declared using the flow prefix:

1. Equality coupling, for nonflow variables, according to Kirchhoff's first law.

2. Sum-to-zero coupling, for flow variables, according to Kirchhoff's current law.

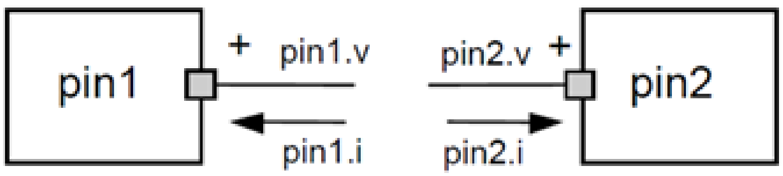


Figure 2.5: Connection between pins. Image taken from [13]

Connection equations are used to connect instances of connection classes. A connection equation connect (`pin1,pin2`), with `pin1` and `pin2` of connector class `Pin`, connects the two pins so that they form one node. This produces two equations:

$$pin1.v = pin2.v$$

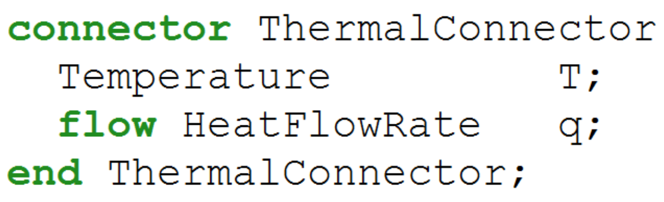
$$pin1.i + pin2.i = 0$$

The first equation says that the voltages of the connected wire ends are the same (across variables, equality coupling). The second equation corresponds to Kirchhoff's second law, saying that the currents sum to zero at a node (assuming positive value while flowing into the component). The sum-to-zero equations are generated when the prefix flow is used (through variables, sum-to-zero coupling). Similar laws apply to flows in piping networks and to forces and torques in mechanical systems [13].

2.4 An example: heat transfer domain

To practically explain the concepts exposed in the previous paragraph, some components of the thermal domain, the main library used in the development of this thesis, are briefly introduced. In order to do this, an example taken from M. Tiller book 'Modelica by example' [14] is proposed, available for free online in HTML version, containing several examples on how the logic behind Modelica is applied for studying, modeling and simulating engineering systems.

A connector for modeling lumped heat transfer is not much different from an electrical connector, see figure 2.6.



```
connector ThermalConnector
  Temperature          T;
  flow HeatFlowRate    q;
end ThermalConnector;
```

Figure 2.6: Modelica code of a thermal connector

Instead of `Voltage` and `Current`, this connector includes `Temperature` and `HeatFlowRate`. The connector includes one through variable (q, indicated by the presence of the flow qualifier) and one across variable (T, indicated by the lack of any qualifier). In fact, the type of the variable with the flow qualifier, HeatFlowRate, is the time derivative of thermal energy. An example of a component model will be a model of lumped thermal capacitance with uniform temperature distribution. The equation we wish to associate with this component model is:

$$C\dot{T} = Q_{flow} \tag{2.3}$$

```
model ThermalCapacitance
  parameter Capacity C;
  parameter Temperature T0;
  node"connector";
 initial equation
  node.T = T0;
 equation
  C*der(node.T) = node.Q_flow;
end ThermalCapacitance;
```

Figure 2.7: Model of a thermal capacity

The Modelica model representing this equation is quite simple: where C is the thermal capacitance and $T0$ is the initial temperature. Note the presence of the node connector in this model. This is how the ThermalCapacitance component model interacts with the "outside world" [14]. The temperature at the node, `node.T`, will be use to represent the temperature of the thermal capacitance. The flow variable, `node.Q_flow`, represents the flow of heat into the thermal capacitance. It is possible to see this when looking at the equation for the thermal capacitance:

$$C \cdot der(node.T) = node.Q\_flow;$$

Note that when `node.Q_flow` is positive, the temperature of the thermal capacitance, `node.T`, will increase. This confirms that it has been followed the Modelica convention that flow variables on a connector represent a flow of the conserved quantity, heat in this case, into the component [14]. This model contains only the thermal capacitance element and no other heat transfer elements (e.g., conduction, convection, radiation). Using the graphical annotations in the model it can be rendered as:

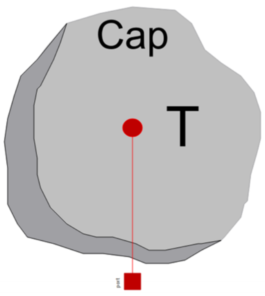


Figure 2.8: Icon view of a thermal capacity

Since no heat enters or leaves the thermal capacitance component, `cap`, the temperature of the capacitance remains constant.

In order to add heat transfer, it is possible to define another component model to represent heat transfer to some ambient temperature. Such a model could be represented in Modelica as shown in Figure 2.9.

```
model ConvectionToAmbient
  parameter CoefficientOfHeatTransfer h;
  parameter Area A;
  parameter Temperature T_amb;
  port_a;
equation
  port_a.Q_flow = h*A*(port_a.T-T_amb);
```

Figure 2.9: Text view of `ConvectionToAmbient` model.

This model includes parameters such as the heat transfer coefficient, h, the surface area, A and the ambient temperature, T_amb. This model is coupled to other heat transfer elements through the connector `port_a`. Observing the equation section in the figure 2.9, it is possible to say that when `port_a.T` is greater than T_amb, the sign of `port_a.Q_flow` is positive. That means heat is flowing into this component. In other words, when `port_a.T` is greater than T_amb, this component will take heat away from `port_a` and vice versa [14].

Having such a component model available enables us to combine it with the `ThermalCapacitance` model and simulate a system using the following Modelica code:

```
model CoolingToAmbient
  ThermalCapacitance cap(C=0.12, T0=363.15)
  ConvectionToAmbient conv(h=0.7, A=1.0, T_amb=298.15)
 equation
  connect(cap.node, conv.port_a)
end CoolingToAmbient;
```

Figure 2.10: Text view of `CoolingToAmbient`

In this model, it is possible to see that two components have been declared, `cap` and `conv`. The parameters for each of these components are also specified when they are declared. The diagram view shown in 2.11 is representative for the `CoolingToAmbient` model:

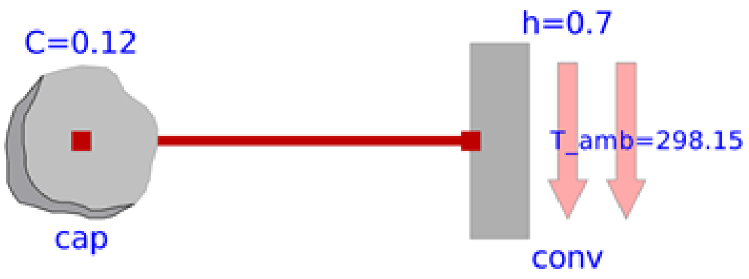


Figure 2.11: Diagram view of the model. Image taken from [14]

What is really remarkable about this model is the equation section:

$$connect(cap.node, conv.port_a);$$

This statement introduces one of the most important features in Modelica: while the connect operator looks like a function, it is much more than that. It represents the equations that should be generated to model the interaction between the two specified connectors, `cap.node` and `conv.port_a` [14]. In this context, a connection does two important things. The first thing it does is to generate an equation that equates the "across" variables on either connector. In this case, that means the following equation:

- $cap.node.T = conv.port\_a.T$ "Equating across variables";

In addition, a connection generates an equation for all the through variables as well. The equation that is generated is a conservation equation. We can think of this conservation equation as a generalization of Kirchoff's current law to any conserved quantity [14]. Basically, it represents the fact that the connection itself has no "storage" ability and that whatever amount of the conserved quantity, in this case heat, that flows out of one component must go into the other(s). So, in this case, the connect statement will generate the following equation with respect to the flow variables:

- $cap.node.Q\_flow + conv.port\_a.Q\_flow = 0$ "Sum of heat flows must be zero";

All the flow variables are summed. In this simple case, with only two components, it can be seen clearly that if one value for `Q_flow` is positive, the other must be negative. In other words, if heat is flowing out of one component, it must be flowing into another. These conservation equations ensure that we have a proper accounting of conserved quantities throughout our network and that no amount of the conserved quantity gets "lost" [14].

A very simple way to summarize the behavior of a connection, in the context of a thermal problem, is to *think of a connection as a perfectly conducting element with no thermal capacitance* [14].

Simulating the `CoolingToAmbient` model above, the following temperature transient is shown in figure 2.12:

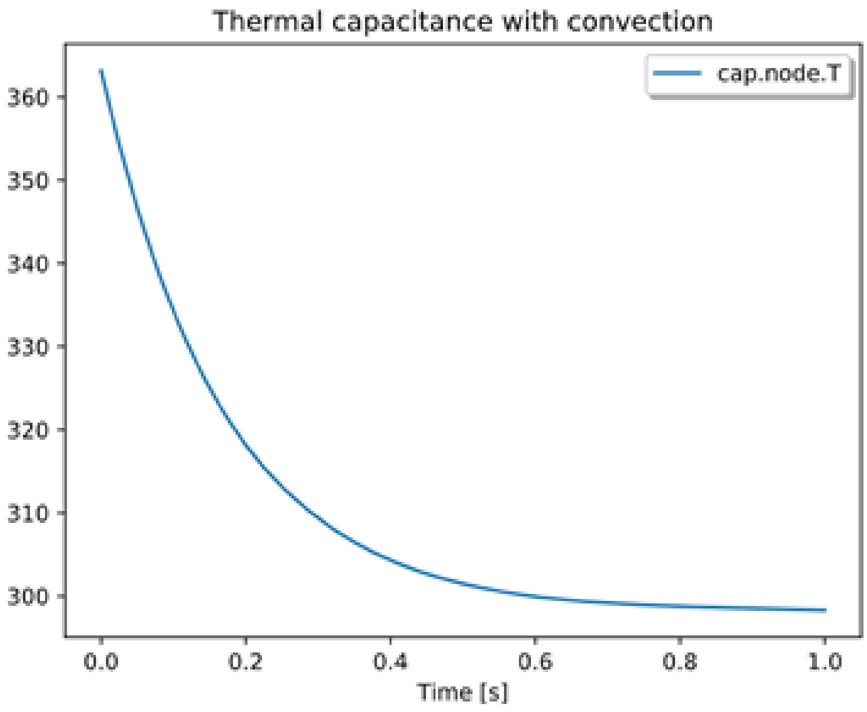


Figure 2.12: Capacitance's temperature plot. Image taken from [14]

Two different effects into one component have been lumped, but this limits the reusability of the component models. Let's refactor the code to separate these effects out. The first new component is a `Convection` model. In this case, no assumptions about the temperature at either end are done. Contrarily, just that each is connected to something with an appropriate thermal connector is assumed. This model contains two equations. The first equation, $port\_a.Q\_flow + port\_b.Q\_flow = 0$ (conservation of energy) represents the fact that this component does not store heat. The equation enforces the constraint that whatever heat flows in from one connector must flow out from the other [14]. The next equation, $port\_a.Q\_flow = h * A * (port\_a.T - port\_b.T)$ (heat transfer equation) captures the heat transfer relationship for convection by expressing the relationship between the flow of heat through this component and the temperatures on either end [14]. Now that the convection model is ready to be used, it is necessary something to represent the ambient conditions. What we want is something that does not change temperature at all, as if it had a *C* value that was infinitely large. This kind of model is commonly called an "infinite reservoir" model [14].

```
model AmbientCondition
  parameter  T_amb;
  node "connector"
 equation
  node.T = T_amb;
end AmbientCondition;
```

Figure 2.13: Text view of `AmbientCondition` model.

Since we are talking about the heat transfer domain, this model is an infinite reservoir for heat and no matter how much heat flows into or out of this component, its temperature remains the same [14]. Using these new `Convection` and an `AmbientCondition` models, it is possible to reconstruct the simple system level heat transfer model using the following:

```
model Cooling
  ThermalCapacitance cap(C=0.12,T0=363.15)
  Convection convection(h=0.7, A=1.0)
  AmbientCondition amb(T_amb=298.15)
equation
  connect(convection.port_a, cap.node)
  connect(amb.node, convection.port_b)
end Cooling;
```

Figure 2.14: Text view of `Cooling` model.

When rendered, the model looks like as shown in Figure 2.15.

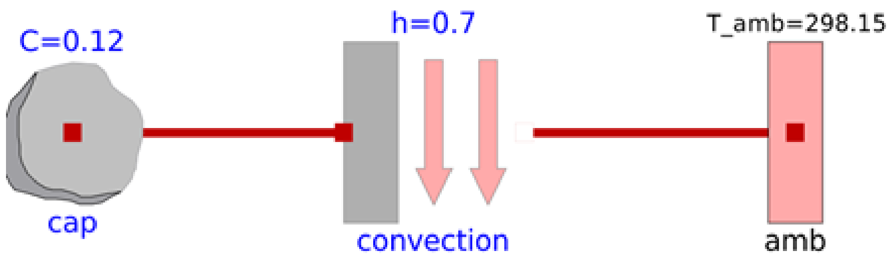


Figure 2.15: Diagram view of Cooling model. Image taken from [14]

This may not seem like much of an improvement. Although we went to the trouble to break up the `ConvectionToAmbient` model into individual `Convection` and `AmbientTemperature` models, we still end up with the same fundamental behavior it is possible to see in 2.12. The big benefit of breaking down `ConvectionToAmbient` into `Convection` and `AmbientTemperature` models is the ability to recombine them in different ways [14]. A new rearrangement could be the one shown in Figure 2.16. With these components it is possible to make arbitrarily complex networks of components and still never have to worry about formulating the associated equations that describe their dynamics. Everything that is required to do this has already been captured in our component models. This allows to focus on the process of creating and designing systems and leave the tedious, time-consuming and error prone work of manipulating equations behind [14].

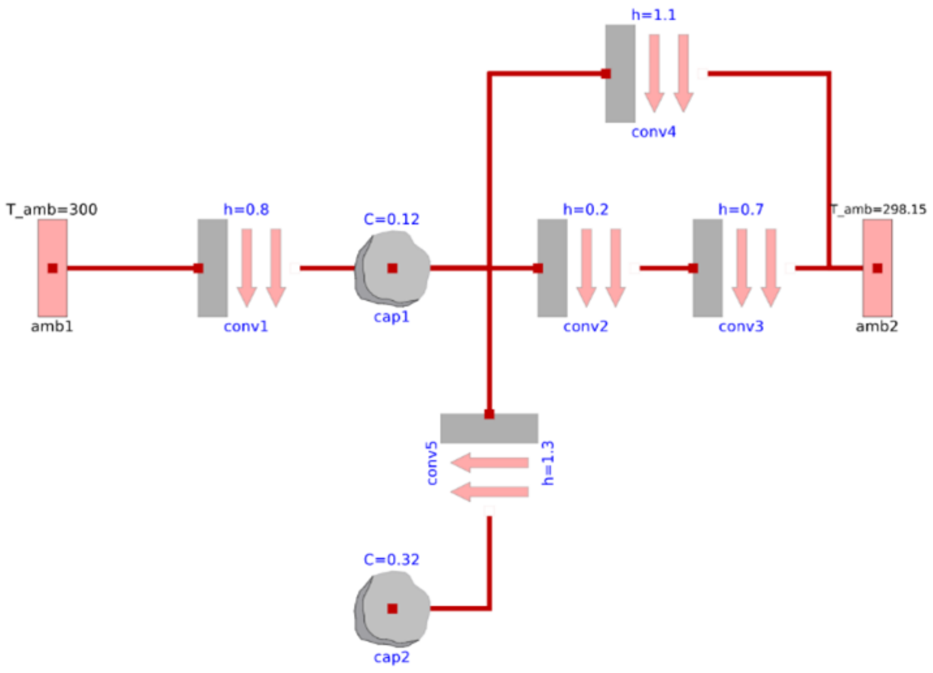


Figure 2.16: Possible components rearrangement. Image taken from [14]

Such example, created and proposed by M. Tiller, has as its goal the clarification of the generic concepts on which is based Modelica and its operation. In the next chapter, the analyzed case study in this thesis will be presented, together with all the models developed on SystemModeler, which will allow its modeling.

Chapter 3

Heat transfer modeling

For the investigation of an air conditioning system and its modeling through *Wolfram SystemModeler*, it is necessary to establish the operational conditions where the system will be working and therefore the definition of a case study which introduces specific design constraints to be followed for modeling activity. Summarily, the aim is studying and implementing the air temperature conditioning for a residential building, by using a heating system composed of a heat pump that provide to feed fan coils, chosen as terminal devices of the required thermal power by the building, through a water distribution network. For this reason, all the components constituting the system have been modeled considering the physical processes that highlight the heat transfer having as goal the assessment of the way in which the parameters are influenced each other. Hence, a dynamic model of the building has been developed, which allows to take in account all the heat transfer phenomena and to estimate the amount of thermal power required after having defined its construction and operational features. Subsequently the modeling of three additional sub-systems has been necessary: the heat pump system, the fan coil system and the distribution system, which links them. In the next paragraphs we discuss separately and in detail each of the above mentioned components, by defining the design constraints imposed by the case study and the operative procedures from the later modeling on SystemModeler.

3.1 Building

The first system to be examined is that related to the building. It represents the place where the thermal comfort must be ensured; its modeling is primarily aimed to the thermal power needed determination, key parameter both for the refrigeration cycle design, that occurs inside the heat pump, and for that of fan coils. In this regard, all the main heat transfer mechanisms to which a building may be subject will be taken into consideration.

3.1.1 Heat transfer through opaque surfaces

In order to achieve a dynamic model representing the thermal behaviour of the building, it is necessary to study all the mechanisms with which it exchanges heat due to the temperature difference between the inner and outside environment.

Let's start by considering the heating dispersion that occurs between the inner ambient of the house and the external environment through all the opaque surfaces, or rather: outer walls, roof and floor

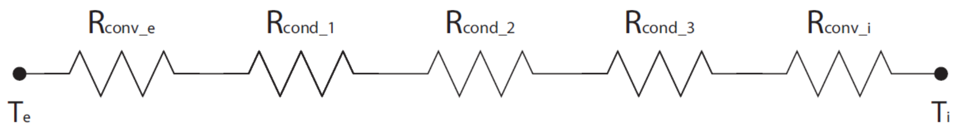


Figure 3.1: Electric analogy pattern

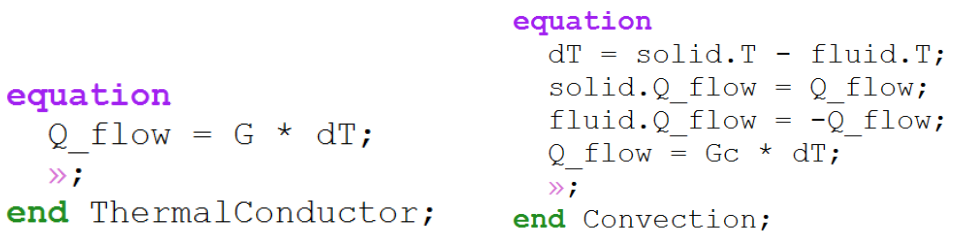
Let's consider the thermal modeling of a wall: to do this the electrical analogy method is used, where each layer of the wall is represented through a thermal resistance that allows the study of the heat transfer mechanism related to it. How it can be noticed by the figure 3.1, the external and internal convection are present as well as the conduction across three different material layers, each of them represented both by their thermal conductivity λ and thickness s whereas, by introducing a simplification, the radiation mechanism is here neglected. By estimating the specific thermal resistance values related to each mechanism, $\frac{\lambda}{s}$ for the conduction, and $\frac{1}{h}$ for the convection, it is possible compute the total transmittance offered by the wall stratigraphy through the following equation:

$$U_{tot} = \frac{1}{\frac{1}{h_e} + \Sigma\frac{s}{\lambda} + \frac{1}{h_i}} = \frac{1}{R_{tot}} \qquad \left[\frac{W}{m^2K}\right] \tag{3.1}$$

Once that value is known, together with the geometric features of the considered wall that define its surface A, the thermal power that it lets flows is computed as:

$$\Phi = U_{tot} A (T_i - T_e) \tag{3.2}$$

Such scenario is implemented on SystemModeler by using and connecting the components shown in the picture 3.3, representing the *diagram view* of the entire `Wall` model, ready-to-use and contained in the Modelica's Thermal library. On the extreme left and right we can find respectively the components `outsideConvection` and `insideConvection` that allow to model the external and internal convective heat transfer to the wall, that in turn consists of three different components: `externalInsulation`, `innerMaterial` and `internalInsulation`; they represent the conductive heat transfer across the three different layer of wall materials. Their relation is governed by the equations contained shown in the figure below.



```
equation
  Q_flow = G * dT;
  »;
end ThermalConductor;
```

(a) Conduction class

```
equation
  dT = solid.T - fluid.T;
  solid.Q_flow = Q_flow;
  fluid.Q_flow = -Q_flow;
  Q_flow = Gc * dT;
  »;
end Convection;
```

(b) Convection class

Figure 3.2: Equation section of the text view of `ThermalConductor` and `Convection` components

All the components just described possess two **Pin** each, one red and one white, that allow the interaction with the others classes. In particular, these are thermal connectors: as already explained in the previous chapter 2.6, they include the thermal flux (through variable) and the temperature (across variable) as main parameters. The thermal flux value that the components let pass is proportional to the product between the difference temperature dT of the two above mentioned pins and, $G{=}\frac{\lambda A_{wall}}{s}$ that is the thermal conductance of the three materials for the conduction, or $G_c{=}h_{i,e}A_{wall}$ for the convection, representing the convective thermal conductance transmitted as parameter to the convective elements thanks to two constant blocks. It is possible interactively changing such parameters in order to study different situations, with for instance different values of conductivity λ or of thickness s or of the area of wall A_{wall} etc.

For considering the capacity of each material layer of which consist the wall, three `heatCapacitor` components are added, one for each layer. This kind of component, already discussed in 2.4, is described by its thermal capacity computed as $C = mcp$ where cp is the specific heat of the material and the mass m is computed by knowing the volume and the density of the corresponding layer, all editable parameters depending on which material we want implement. These capacitors are connected to the respective layers through the only one pin at their disposal. After having defined all the components and their parameters, they are connected each other through their connectors (pins) and this generates some connection equations between the variables of each individual components A.1 obtaining a unique model which represents the manner in which an external wall lets heat through. Providing as input two different values of extreme temperatures through the two connectors `outsideTemperature` and `insideTemperature` visible in the figure 3.3, depending on the geometry and on the thermal parameters chosen, the `Wall` model allows to estimate the heat flux dispersed and the temperature profile in the considered conditions, thus its thermal behaviour.

Once created, it is stored inside the BUILDING library, into MATTSURFACES sub-package ready to be used and graphically represented by its icon view:

In fact, is exactly by applying the re-usability characteristic of the object-oriented modeling approach model that the `Roof` model has been realized, thermally described identically to what has been done for the external wall; its diagram and icon view are shown in the following figure:

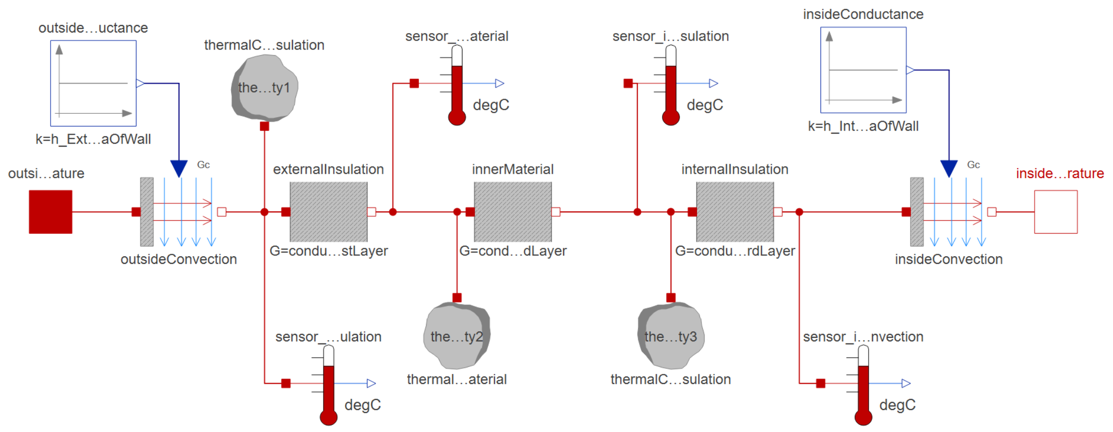
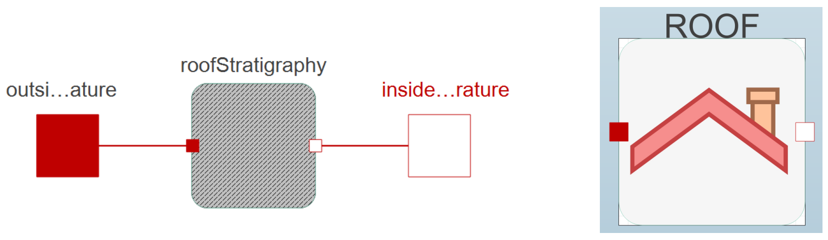


Figure 3.3: Diagram view of a `Wall` model



(a) Roof diagram view (b) Roof icon view

Figure 3.5: Graphical representation of `Roof` model

As it can be seen, just dragging and dropping the `Wall` model from the BUILDING library to a new model, giving him `roofStratigraphy` as name and creating two new pins allows to create the complete roof class. Again, the two connectors present in the icon view enable the connection whit the temperatures values necessary to its thermal modeling.

Finally, the `Floor` modeling is identical to the one used for the wall except for the lack of the external convection and for the presence of a single material's layer, as appreciable by the figure 3.6. In this case, a constant temperature value is defined representing the ground temperature which is supposed never change. Indeed, only one thermal pin is present that will allows, as we will see later, the connection with the 'inner' temperature that is the temperature of the air inside a room. In this case as well, the capacity of the ground to retain heat is modelled thanks to the presence of a `heatCapacitor` working exactly as in the `Wall` model.

For the wall, and so the roof, we have three different material layers; we need to define parameters as conductivity, density, thickness etc. in order to thermally determine the stratigraphy. In the following table all the necessary parameters chosen for the thermal characterization of each layer are presented, where ρ is the density, λ is the thermal conductivity, s the layer's thickness and cs the specific heat

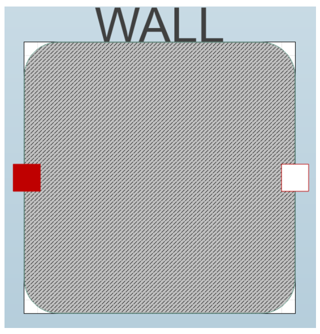


Figure 3.4: Icon view of the `Wall` model

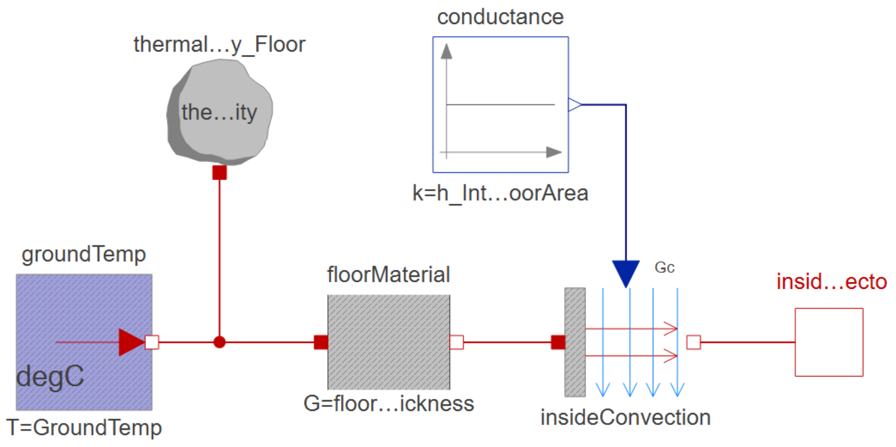


Figure 3.6: Graphical diagram view representing the `Floor` thermal model

LAYER	$\rho\ [\frac{kg}{m^3}]$	$\lambda\ [\frac{W}{mK}]$	$s\ [m]$	$cs\ [\frac{J}{kgK}]$
External Insulation	90	0.037	0.02	1030
Inner Material	1681	0.6	0.05	840
Internal Insulation	410	0.086	0.01	1000

Table 3.1: Wall materials properties

for each material. Regarding the `Floor` model, the parameters and the properties of its single layer are shown in the table 3.2. As mentioned earlier, the ground temperature is considered constant and equal to 12 C°. In all the models, for the internal and natural convection, two constant values $h_i = 10\ \frac{W}{m^2K}$ and $h_e = 40\ \frac{W}{m^2K}$ respectively are guessed.

Finally, all the models belonging to the MattSurface subpackage are done.

3.1.2 Heat transfer through doors and windows

Let's now pass to the windows and doors modeling, extremely important elements that have to be considered for the computation of the total dispersed thermal flux by the building. As regards to the door, guessing it constituted by a single wood layer and by applying the electric analogy once more, the two convective and the only one conductive specific thermal resistance are estimated using the well-know expressions:

LAYER	ρ $[\frac{kg}{m^3}]$	λ $[\frac{W}{mK}]$	s $[m]$	cs $[\frac{J}{kgK}]$
Floor material	1680	0.07	0.05	840

Table 3.2: Floor material properties

$$R_{conv_{e,i}} = \frac{1}{h_{e,i}} \left[\frac{m^2K}{W}\right] \qquad R_{cond} = \frac{s}{\lambda} \left[\frac{m^2K}{W}\right] \tag{3.3}$$

Knowing them, the total transmittance U_{door} for the door is computed as:

$$U_{door} = \frac{1}{R_{tot}} \left[\frac{m^2K}{W}\right] \tag{3.4}$$

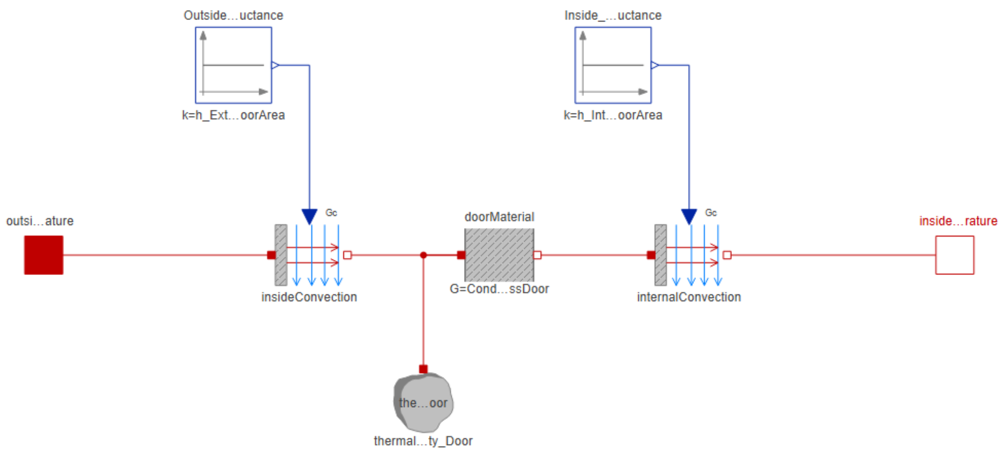
where the total resistance is the sum of all the resistance $R_{tot} = R_{conv_{e,i}} + R_{cond}$.

Such component is thermally modelled on SystemModeler in an analogous way to what has been done for the outer wall, or rather considering the external and internal convection and the conduction across the only material layer `doorMaterial` in which the door consists and creating a new class called `Door` by utilising and connecting the already used components for the convection and conduction modeling of the wall, as shown in the figure 3.7a. Its capacity to retain heat is taken in consideration thanks to the presence of a `heatCapacitor` component, where the core parameter is $C = mc_s = \rho V c_s$ where ρ is the material density, V the door volume and c_s the material specific heat. In this case, the door geometry is guessed to be constant and equal to $A_{door} = 1{,}68\, m^2$ so the volume is equal to the product between A_{door} and the thickness of the door. Through the thermal connectors present in the icon view shown in figure 3.7b, the door component is connected with the temperature values that determine its thermal behaviour or with other thermal models, as we'll see later, just dragging and dropping it into a new class. Even for the door, all the material properties defined by the case study are now presented in the table 3.3, to definitively fix all the characteristics.

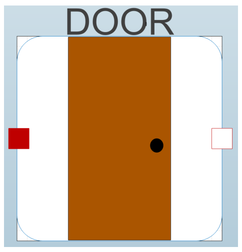
LAYER	ρ $[\frac{kg}{m^3}]$	λ $[\frac{W}{mK}]$	s $[m]$	cs $[\frac{J}{kgK}]$
Door material	500	0,13	0,06	1600

Table 3.3: Door material properties

Let's now consider the window component; through it heat is dispersed via conduction and convection across the frame, which is considered made of both



(a) Door diagram view



(b) Door icon view

Figure 3.7: Graphical representation of `Door` model

matt and transparent surfaces, via radiation through the transparent surface only and, lastly, via ventilation. Each of these mechanisms is now individually analyzed.

Starting by the frame, its thermal characterization consists on the determination of the total transmittance U_w by using the electric analogy strategy, exactly how it has been done for the the previous components. However in this case U_w is computed through a specific expression that takes in consideration both of the glazed part and the matt part of which the frame consists. That expression is the following:

$$U_w = \frac{A_f(v\,U_v + t\,U_t) + \Psi\,p}{A_f} = 1{,}773 \quad \left[\frac{W}{m^2K}\right] \tag{3.5}$$

where:

- A_f is the total window surface;
- v is the glass surface portion respect with the total surface A_f;

- t is the frame surface portion respect with the total surface A_f;
- Ψ represents the linear transmittance of the glass edge;
- U_v represents the glass transmittance;
- U_t represents the frame transmittance;
- p is the window perimeter;

Such specific parameters are defined by windows manufacturers complying the constraints imposed by the legislation. The selected values of these parameters are shown in the table below.

A_f $[m^2]$	v $[/]$	t $[/]$	Ψ $[\frac{W}{mK}]$	U_v $[\frac{W}{m^2K}]$	U_t $[\frac{W}{m^2K}]$	p $[m]$
3,2	0,8	0,2	0,06	1,6	1,8	7,2

Table 3.4: Parameter values for the window transmittance

Regarding the geometry of the window, the total surface and the perimeter have been chosen equal to the values shown in the table and has been guessed to be constant for all the windows in the building, to simplify the treatment.

Let's consider now the radiation heat transfer contribution through the glass surface; it represents one of the two positive component of the heat flux to the thermal balance, being incoming into the house rather than out-going. For the sake of simplicity, such contribution is considered constant and dependent on the incident solar radiation ψ on the considered window, on the surface A_f of the aforementioned window and on a optical coefficient G, in order that the provided heat flux can be computed as:

$$\Phi = \psi \, G \, A_f \quad [W] \tag{3.6}$$

A strong approximation has been introduced about the radiation mechanism; in fact its value has been kept constant for each direction (N,S,W,E) even if this is not comparable with the reality, where these values change continuously. Since these values are dependent by the place where the heating system will be placed, their specific definition will be presented when the apartment model will be discussed.

Lastly, let's consider the ventilation contribution; whereas ventilation is usually carried out through ventilation channels, for simplicity it was here taken into account together with the windows' model. Assuming that the ambient needs a complete circulation of air every hour, the minimum air mass flow rate required for the air renewal is computed as $G_a = \frac{V\rho}{3600}$ $[\frac{kg}{s}]$ where V is the room volume and ρ

the air density considered to be constant and equal to $\rho = 1{,}2\ [\frac{kg}{m^3}]$. At this point, the dispersed thermal power by ventilation is obtained as:

$$\Phi = G_a\, c_p\, \Delta\, T\ [W] \tag{3.7}$$

The c_p of the air is considered to be constant as well, equal to $c_p = 1005\ [\frac{J}{kgK}]$.

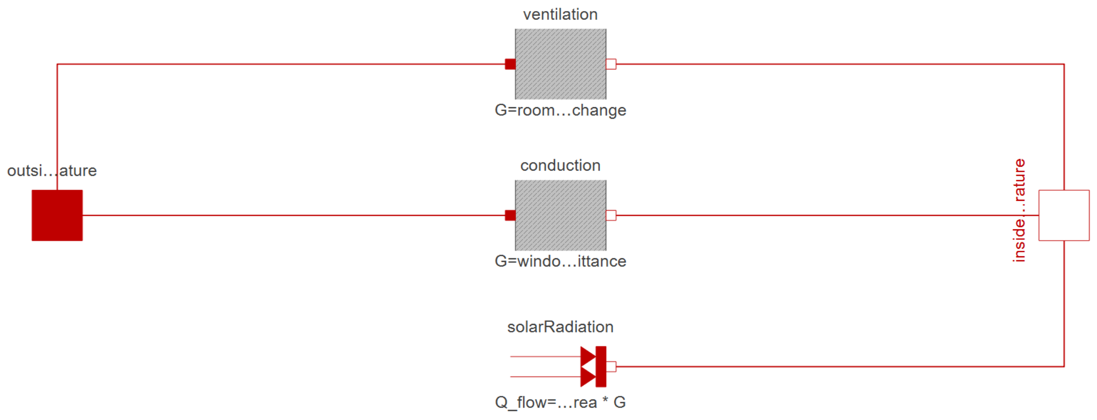


Figure 3.8: `Window` model diagram view

Such model is implemented on SystemModeler connecting three different components in parallel that allow to consider these heat transfer mechanisms. Making reference to the figure 3.8, on the top it is possible to notice the presence of a `thermalConductor` element under the name `ventilation` with which the equation 3.7 is implemented, where the product between G_a and c_p in considered to be a 'fictitious' conductance. Also here there are two thermal pins: the red one allows the connection with the outside temperature value through the thermal connector `outsideTemperature`, while the white one with the inside temperature value through `insideTemperature`.

In the middle, we see another `thermalConductor` element, named `conduction`, with which the dispersion across the frame is considered; the core parameter for a component like this on SystemModeler is the conductance G, that here is equal to U_w (3.5) multiplied for the window surface A_w. Through its two pins, it is connected to the same ports previously mentioned.

The last element on the bottom, presents in the THERMAL library of Modelica and in the window model under the name `solarRadiation`, allows to a certain constant thermal power amount defined *a priori* to be injected to the component it is connect with. It is used to model the solar radiation contribution, supposed constant, by implementing the previously presented equation 3.6. As it can be observed, it possesses only one pin that allows its connection with the thermal port `outsideTemperature`; this means that it does not need a connection with the inner temperature value, since the thermal power it let pass does not depend on a temperature difference.

Summarising, the model just realized is able to take into account of the way, even if with some simplification, with which the window element exchange heat.

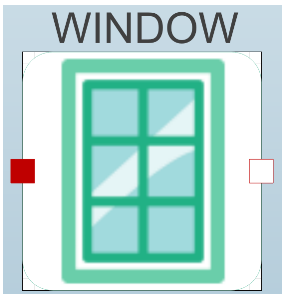


Figure 3.9: `Window` model icon view

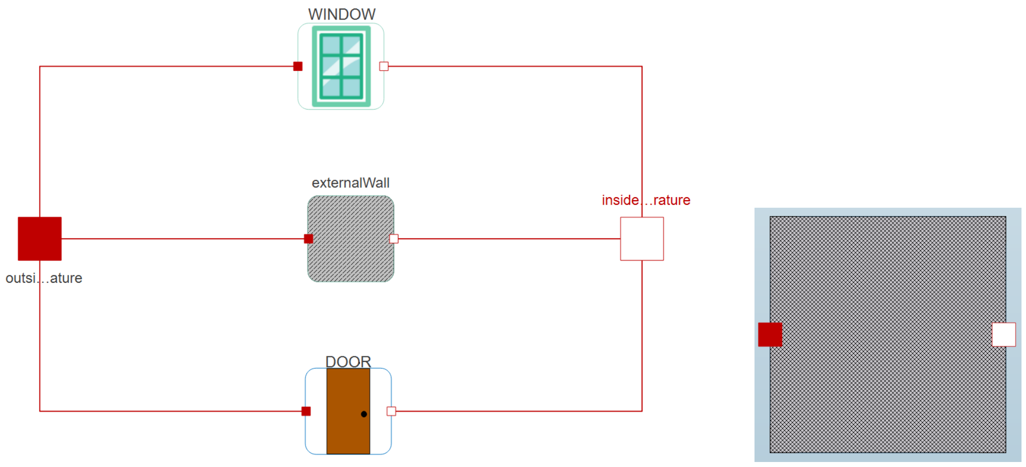
By observing the icon view of such model 3.9, we can see the pins that are the thermal ports to which conduction, ventilation and radiation are connected. Through the pins, the window element can be connected to the temperatures values or with others thermal components to create new more complex model, HIERARCHICAL, as we will see soon.

3.1.3 External and Internal wall models

Now that we have modeled the outer wall, the door and the window, by applying one of the main feature of SystemModeler and of the OOM in general it is possible the developing of two *hierarchical* models: the `ExternalWall` and the `InternalWall`. By referring to the figure 3.10a, the new `ExternalWall` model consists on the connection of the `Wall`, `Door` and `Window` models to two thermal ports. This is like having a model of a wall where a door and a window are present and depending if these elements are effectively present in the considered external wall, there is the possibility to consider them or not. In this way, instead to have three different models, we have only one model that contains all them.

Let's make an example useful to explain the logic under its operating. Suppose we want model an outer wall in which a window is present but not a door; in the new `ExternalWall` model, contrary to the only `Wall` model, the net surface is equal to the total wall area deprived of that of the door and that of the window. Since in our example there is a window but not a door, a null value for the parameter DOORAREA will be set, in order that all its thermal dynamic, addicted to this parameter, will be canceled. The right parameters for the window (area, perimeter, volume of the room, solar radiation etc.) will instead be set. In doing so, it is taken into consideration that there is a window in that wall, which obviously exchanges heat in a different way than the wall surface. The effective wall area will be equal to its total, minus the window area; the net heat flux exchanged by the wall will depends on this value. The other contribution to the total heat flux is provided by the window model.

Thus, by using this new model `ExternalWall` and by setting equal to 0 both the area of the door and the area of the window, such model will behave exactly as the `Wall` model. This confers an extreme modeling flexibility, as we will see in the `Rooms` and in the `Apartment` modeling.



(a) External wall diagram view **(b)** External wall icon view

Figure 3.10: Graphical representation of `externalWall` model

In addition to having walls in contact with the outside environment, in a house each room shares one or more walls whit the other rooms, thus they are internal walls. Surely we have to include a door that allows switching from a room to another one. Hence even for this situation, to consider the presence of a door in an internal wall and the fact that wall and door exchange heat differently, the `InternalWall` model has been developed. The model consists of the connection of a `Wall` and a `door` model, graphically represented through a diagram view similar to the `externalWall` model except for the presence of a window. By setting the right value for the area of the door, the net wall surface useful to the heat transfer is the total minus the door's area, which provides a different values of thermal power dependent on its modeling.

3.2 Rooms

Once all the models of the main components responsible for heat transfer between the inside and the outside of a room or a building in general are obtained, the re-usability and modularity features that distinguish the OOM approach are applied to create larger models. The idea is now to create a model representing the thermal behavior of a generic room, where all the parameters of all the component in which it consists of can be modified, in order to adapt this generic model to a specific situation. Thus, just one model under the name `GeneralRoom` has been realized in which all the core parameters (geometry of each wall, presence of windows or not, presence of internal wall or not, presence of people etc.) can be set depending on the room that then we wish to implement, that it is the kitchen or the bathroom or any other room.

Starting from a single model, all the rooms in the building are realized and then, as we will show soon, they are connected with each other to assemble the overall model. This results in an extraordinary modeling flexibility, with the possibility to create a moltitude of different building with various geometries by applying and adapting case by case just one model.

Let's see how the `GeneralRoom` model has been realized: in the diagram view

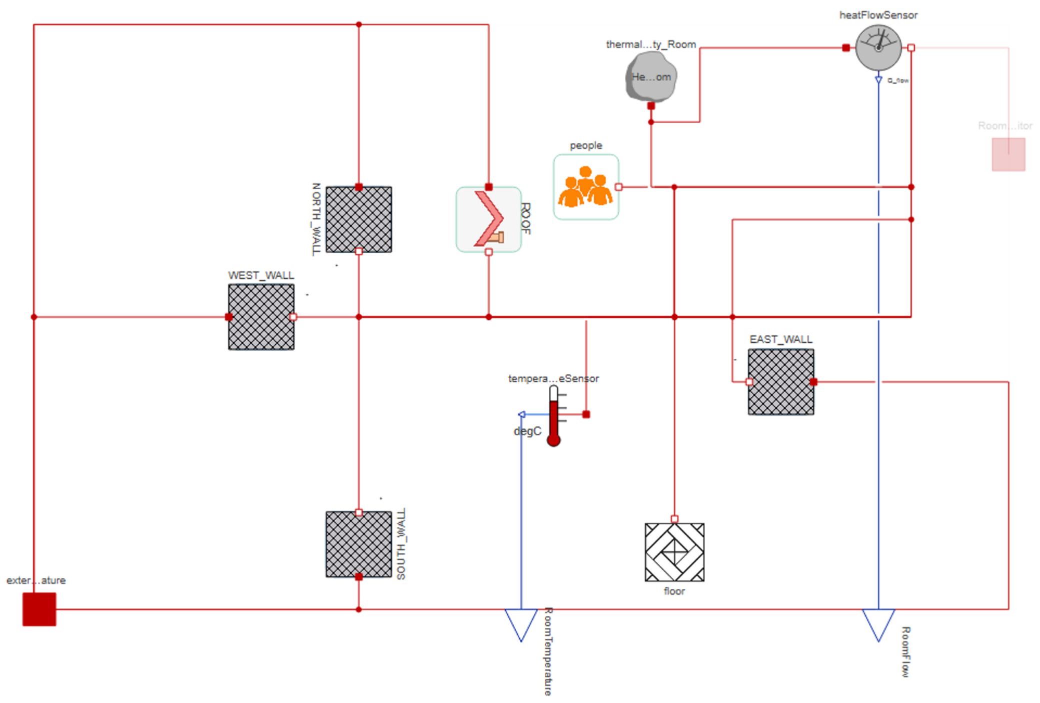


Figure 3.11: `GeneralRoom` model diagram view

shown in figure 3.11, the generic room consists of the connection of four wall models representing the outer walls, one floor and one roof components. Each of these, as abundantly discussed, present two thermal pins; the red one connects each model to the external temperature value through the common thermal connector `externalTemperature` noticeable on the lower left. Instead, the white one is connected to a `heatCapacitor` element, named `thermalCapacityRoom`, thermally representing the air inside the room and its capacity to retain heat; this component is indeed represented not only by its thermal capacity but by its initial temperature as well. By providing the external temperature through the `externalTemperature` connector, and by defining the initial temperature of the heat capacitor, which represents the internal temperature, the total flux dispersed by every room caused by a temperature difference between the inner and the outer environments is collected in the `heatCapacitor` component. Thus, `heatCapacitor` can be considered the thermal representation of the room's behavior, taking account of the temperature variationcaused by heat dispersion. Thanks to the apply of a temperature sensor and a heat flow sensor connected with the `heatCapacitor` component, we can keep track of these values and produce some plots that show their trend as a function of time, as will be demonstrated later. The presence of a thermal connector called `RoomCapacitor` is crucial, the red one at the right on the icon view 3.12, which is only connected to the `heatCapacitor` of the room and that will allows, as we will see, the connection of two rooms that share a wall, that it is therefore an internal wall. Here, then, the model of a generic room obtained thanks to the combination of the previously developed models, where by selecting each component presents inside it is possible to vary all the parameters, geometric and related to the heat transfer mechanisms, to build the thermal model of the considered room. In turn, this model graphically represented by its icon view, is reused to realize the thermal apartment model, object of next section.

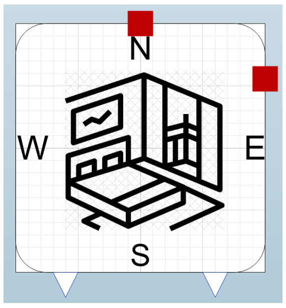


Figure 3.12: `GeneralRoom` model icon view

3.3 Apartment

Finally, after having thermally modeled the main components of a room and combined them in order to assemble the `GeneralRoom` model, this may in turn be used to realize the thermal model of a residential apartment. For this purpose, the building layout definition imposed by the case study considered in this thesis activity is needed.

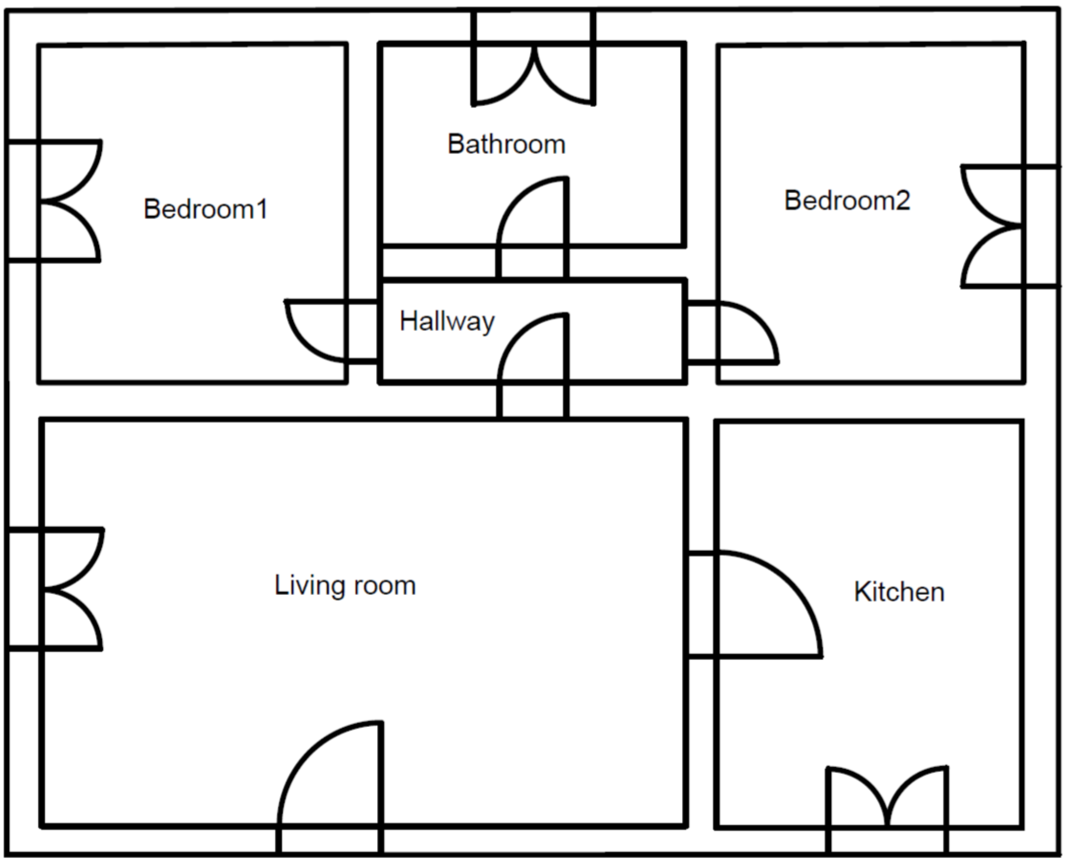


Figure 3.13: Case study apartment layout

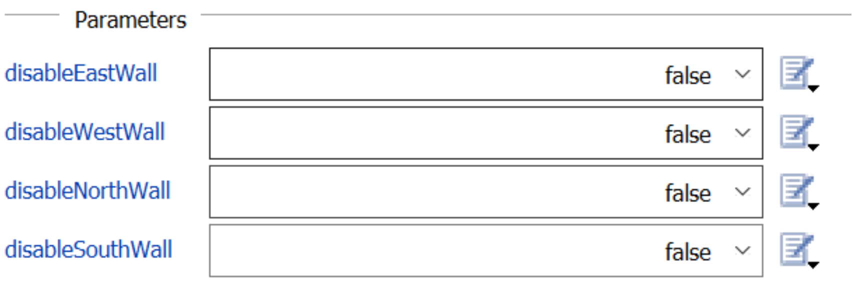
An apartment consisting of six rooms arrange on just one floor is taken: in the figure 3.13 the plan of the building can be noticed imposed by the case study, while in the table 3.5 the geometry for each room is summarized. Such layout is only one of the countless possible; as we we will see, thanks to the flexibility of SystemModeler and the OOM approach, any building's configuration will be possible. Before proceeding with the apartment modeling, let's introduce the main feature of the `GeneralRoom` model. For it in fact, the possibility to *disable* one

ROOM	Length $[m]$	Width $[m]$	Heigth $[m]$
Living room	8	5	3
Kitchen	4	5	3
Bed room 1	4	5	3
Bed room 2	4	5	3
Bath room	4	3	3
Hallway	4	2	3

Table 3.5: Geometric dimensions of each room

or more walls has been implemented simply by selecting which by the drop-down menu in the parameter tab as shown in figure 3.14. This is required to realize the connection of rooms sharing one or more walls. Looking again at the `GeneralRoom` model diagram view 3.11, it can be noticed that all the walls (S,N,W,E) are *active* and that the`RoomCapacitor` Pin is instead deactivated. By disabling one or more walls, their related models are annulled and the thermal pin `RoomCapacitor` is activated that allows the rooms connection.

Let's see how this mechanism practically works. Knowing the disposition and the geometry of the various rooms, their models (kitchen, living room, etc.) may be realized as already explained in the discussion about the Rooms section; for instance, to represent the `LivingRoom`, all the `externalWall` models must be selected one by one in the `GeneralRoom` model in order to manually set the right geometric values. The west wall presents a window, thus all the parameters for the window must be set while, since the wall does not have a door, all its parameters must be equal to zero, as already largely explained in the treatment of the `externalWall` model. The same for the `roof` and `floor` components must be done. Once setting all the parameter for each component of the considered room, its thermal model is ready. We now repeat for another room, always starting from the `GeneralRoom` model inside the ROOMS sub-package on SystemModeler, until all the rooms models are


Parameters

disableEastWall	false
disableWestWall	false
disableNorthWall	false
disableSouthWall	false

Figure 3.14: Drop-down menu to disable or not a wall

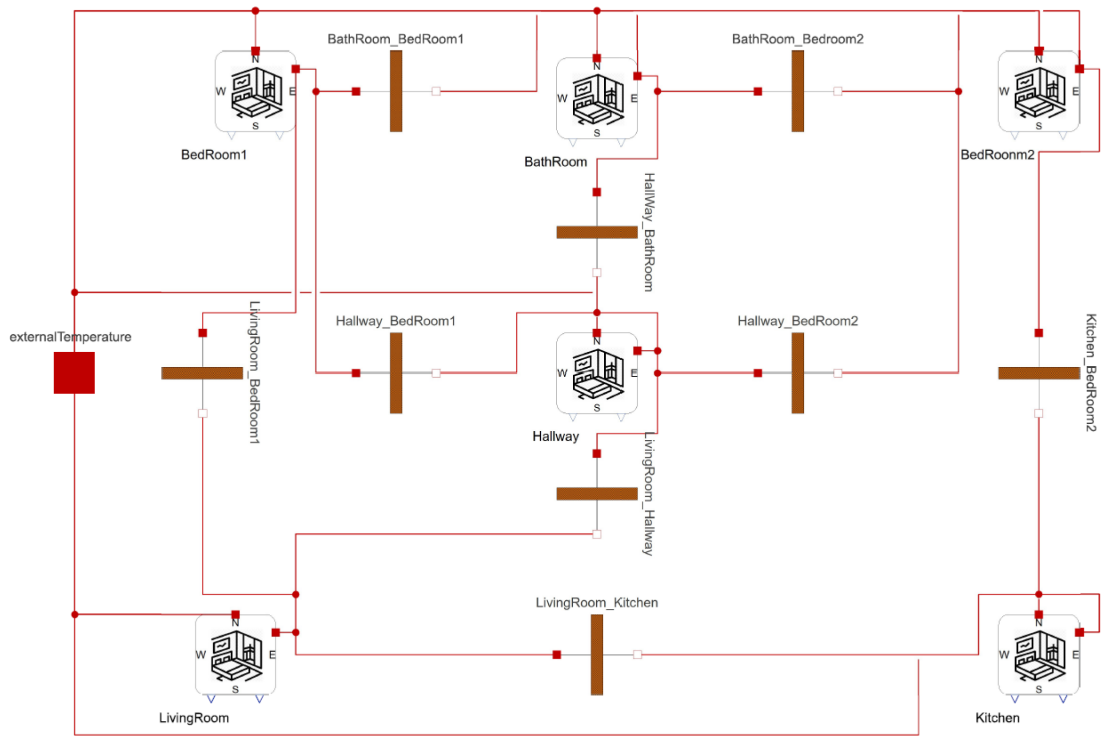


Figure 3.15: `Apartment` model diagram view

ready. At this point, following the defined layout they can be connected each other to create the Apartment model, whose diagram view is shown in the figure 3.15.

Let's take as example the `Kitchen` to explain how its internal walls have been connected to the rooms with which it shares them: the north wall of the `Kitchen` is the same to the south wall of the `BedRoom2`; thus we need to go inside the model of these rooms and disable the north wall for the `Kitchen` and the south wall for the `BedRoom2`. In this way the `RoomCapacitor` pin of each considered room is active allowing the connection of the two rooms by interposing among them an `internalWall` model, whose pins are connected one to the `RoomCapacitor`'s pin of the `Kitchen` and the other one to that of the `BedRoom2`. Doing so allows the rooms connection to diversify the way in which heat is transferred between rooms from the way it is exchanged between the internal and external environment. Following such logic, the complete apartment thermal model is obtained.

Finally we have obtained a model representative of the thermal behaviour of a well determined residential apartment. By using it alone, it is capable to provide the effective thermal power dispersed due to the difference in temperature between the internal environment and the outside climate condition, that we have not yet introduced. Indeed, to size the thermal machines (fan coils, condenser and evaporator) that have to provide the right thermal energy to guarantee the desired internal comfort, we need to estimate the total thermal load required by our apartment when it is located in a specific place. To do so, the worst condition in terms of external temperature to which the apartment may be subject must be considered, taking into account also the thermal features of the external and the internal insulation and of the inner material.

To do this, the design external winter temperature of the place where we want locate our apartment must be considered, taken from UNI 5364. Supposing we want an apartment located in Milan, which is classified to be in the climatic zone E, the extrapolated value for that temperature is equal to -5 ° C. Regarding the inner temperature, its desirable value is set to 20 ° C as required by the current legislation to guarantee the right comfort. Thus considering these conditions, by sending the external temperature signal to the `externalTemperature` pin visible in figure 3.15 and by simulating the complete apartment model it is possible plotting the value of the total thermal load required by the building as a function of time and those required by each room needed to compute the right quantity of water to sent to the fan coils, as we will see soon.

ROOM	Design Thermal Load [W]
Living room	3660.31
Kitchen	2014.78
Bed room 1	2013.57
Bed room 2	2013.57
Bath room	1069.35
Hallway	309.76
Total	11080.17

Table 3.6: Design thermal load for each room considering $T_{outside} = -5°C$

The figure 3.16 shows the plot of the trend of these values for each room, while tha table 3.6 contains the numeric values of these quantity at $t = 0$ when we have $20°C$ in the whole house and $-5°C$ externally, or rather the worst condition necessary for the thermal devices sizing.

The total thermal load starts from the *static condition* and, due to the temperature difference between internal and external environment, it decreases since with the passing of time the inner temperature of the house decreases considering that there is not any fan coils that injects heat in order to raise its value. The thermal load for the kitchen and the two bedrooms it's essentially the same because they are geometrically identical.

Regarding the solar radiation, we can now define the value for the solar radiation

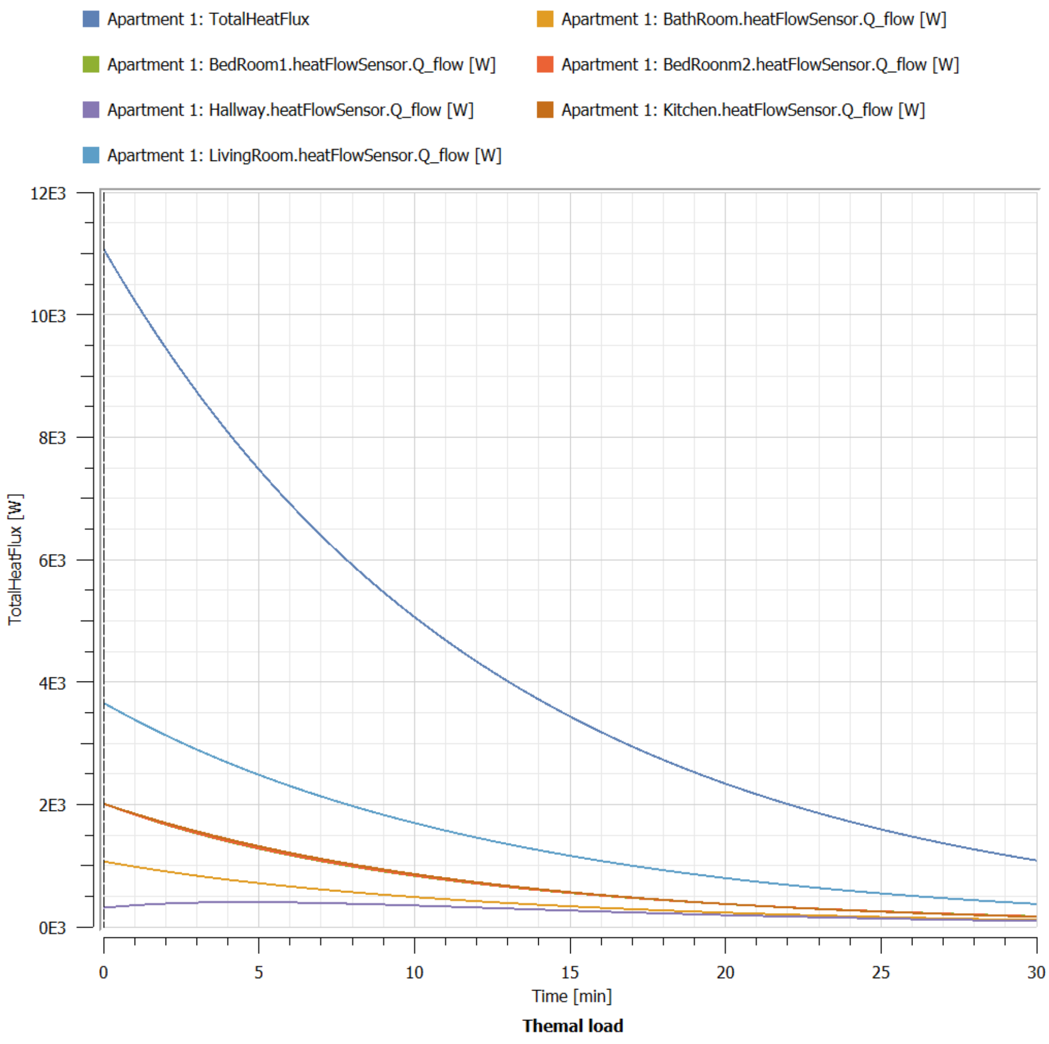


Figure 3.16: Plot of the design thermal load values for the considered apartment in Milan

needed to compute the related heat flow. What we need is the value of the specific heat flux ψ measured in $\frac{W}{m^2}$ knowing the global daily irradiance of the month with the worst values, or rather December with a value of global irradiance equal to 1,5 $[\frac{MJ}{m^2}]$. By assuming ten hours of light in a day, computing ψ by dividing the global irradiance for the light ours in a day, $\psi = 41{,}7$ $[\frac{W}{m^2}]$ is obtained. This value is kept constant and equal for each direction S,O,W,N; that's represent a strong approximation since in the reality this values change continuously and is different depending on which direction we consider.

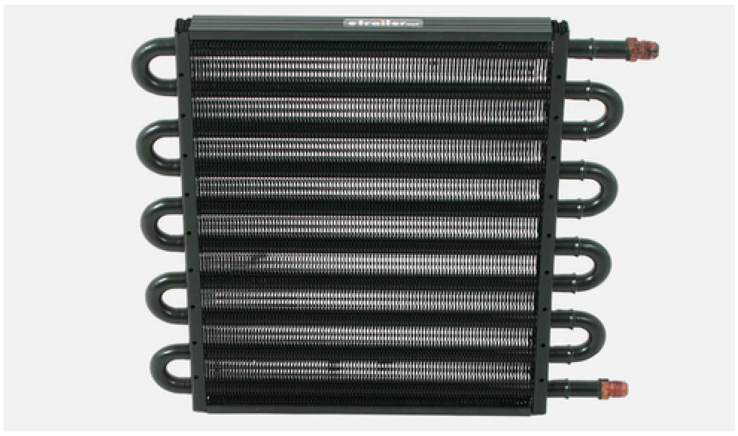
Now that we have the thermal model of the building that must be warmed, let's consider and model the equipment needed for its heating.

3.4 Fan Coils

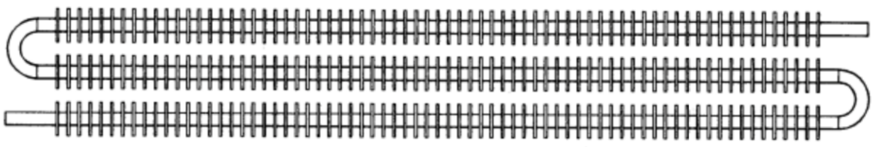
Let's pass to the study of the thermal devices needed to heat the internal ambient. Knowing that the COP of a heat pump increases with decreasing of the temperature difference between hot and the cold sources, it is good as general rule to choose as a hot source one allowing the lowest possible delivery temperature in order to maximize the COP end therefore the operating of the system. For this reason, air-to-water fan coils have been chosen as devices characterised by a delivery temperature around 45-50 °C, relatively lower compared to 70-80 °C that we should have ensure using common radiators. With this premise, the theoretical and constructive aspects necessary for the correct sizing procedure of the fan coils will be presented and lastly how they have been modeled and implemented on SystemModeler.

3.4.1 Finned tube heat exchangers

As before hinted, fan coils have been chosen as emission devices of the required thermal power by the apartment. Since for the thermal exchange the involved fluids are on the one hand the hot water and the room's air that need to be warmed on the other, they are usually designed as finned tube heat excanghers; such choice is justified by the fact that the heat transfer coefficient on the liquid (water) side is typically one order of magnitude greater that that on the gas (air) side. Thus, in order to balance the thermal conductance of both sides (same hA) and to reduce the exchanger's size, fins are applied on the air side to increase the surface of exchange and therefore the heat exchange's efficiency as well [15]. Furthermore, heat exchangers are usually classified according the *flow arrangement*; in the case of finned tube heat exchanger, it is of *cross flow* type, where the two fluids move in cross flow perpendicular each other. The figure 3.17a shows one of the possible constructive solution for a heat exchanger of this kind, together with a scheme showing how the finned tube looks like 3.17b.



(a) Constructive example. Source `www.badrinheatexchangers.com`



(b) Scheme. Image taken from [16]

Figure 3.17: Finned tube heat exchanger example

Fins can be of various types, but the most common in the air conditioning industry are surely the circular ones presented in the figure 3.18.

Sizing procedure's purpose of such an exchanger is the tube's length evaluation needed for the required heat transfer, thus the exchange surfaces, and moreover the estimation of the heat transfer coefficients h_{inner} and h_{outer} both for the water and air side.

Regarding the inner convection on the water side, since we have an outflow into a circular pipe, the *Dittus-Boelter* correlation will be used for estimating the *Nusselt* number that will allow the computation of h_{inner}. Speaking about the air side convection the evaluation for h_{outer} is made more complex by the presence of the fins, requiring a more accurate analysis and the applying of particular correlations in order to take into consideration the specific geometry possessed by the exchanger. Hence, a specific sizing procedure has been followed, exposed in the Shah book [15] based on the $\varepsilon - NTU$ method and that is now reported step by step.

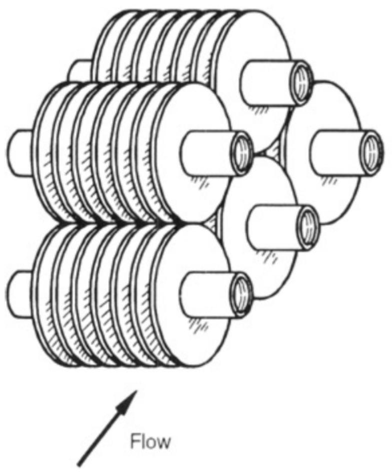


Figure 3.18: Circular fins

3.4.2 Sizing procedure

For the fan coil's exchanger sizing it is necessary to dispose the design thermal power value, hence that computed for the worst external climate condition. As such value that related to the room requiring the highest thermal power will be used, or rather the *Living Room*, as displayed in the table 3.6. The exchanger sizing for this condition will be realized and all the other related to the other rooms will be equal to the studied one. Indeed, since other rooms need a lower thermal power, having sized the fan coils for the living room gives as the security that they are able to provide the required value by the other rooms without issues.

Let's present the procedure:

1. First of all, the inlet and outlet temperatures for both the fluids are defined necessary for the sizing, notable in the scheme shown in the figure 3.19 that simply represent the exchanger. As it can be seen, the water inlet temperature is fixed to 45°C; a difference of temperature of 10°C between intlet and outlet is supposed, obtaining a temperature of 35 degrees outgoing. Regarding the air, the guessed values of 15°C at the inlet and 30°C at the outlet have been chosen to be very conservative.

   By knowing now the jump in temperature along the water side and the considered thermal power value, the needed water flow rate is computed using the following thermal balance:

$$\dot{m}_w = \frac{\Phi_{LR}}{cp\Delta T} = 0{,}0884\ \frac{kg}{s} \tag{3.8}$$

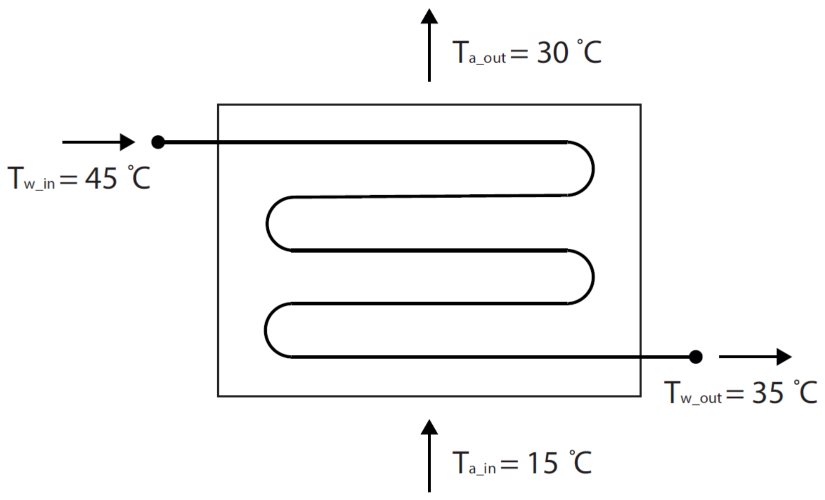


Figure 3.19: Scheme of the coil

where $cp = 4186\ \frac{J}{kgK}$ is the specific heat of the water considered to be constant, $\Delta T = 10°C$ and Φ_{LR} is the aforementioned thermal power related to the Living Room, equal to 3660,31 W. For the air flow rate, its value is considered to be constant equal to, a typical value taken from the indications contained in the Caleffi handbook [16]. This represents a simplification, since the fan's velocity,supposed to be unique, can't be varied; in the reality it can be modified in order to consider different fan's velocity values and so different levels of air convection. By being in possess of the flow rate values of the fluids, their thermal capacity referred to the unit time are now computed as:

$$C_{water} = \dot{m}_w\ cp_w = 370\ \frac{J}{Ks} = C_{max} \tag{3.9}$$

$$C_{air} = \dot{m}_a\ cp_a = 251{,}25\ \frac{J}{Ks} = C_{min} \tag{3.10}$$

where $cp_{w,a}$ are the specific heat mean values; that of water is the same to the already presented value, that one for the air is equal to 1005 $\frac{J}{kgK}$.

Now it is possible compute the heat capacity ratio c^* required by the procedure, as:

$$c^* = \frac{C_{min}}{C_{max}} = 0{,}679 \tag{3.11}$$

2. Once c* has been computed it is used for the evaluation of NTU. The exchanger's efficiency ε is computed by the following:

$$\varepsilon = \frac{q}{q_{max}} = \frac{q}{C_{min}(T_{h,in} - T_{c,in})} = 0{,}49 \tag{3.12}$$

where q represents the real thermal power the exchanger provides, whereas q_{max} is the maximum exchangeable thermal power, or rather that one with the maximum temperature difference.

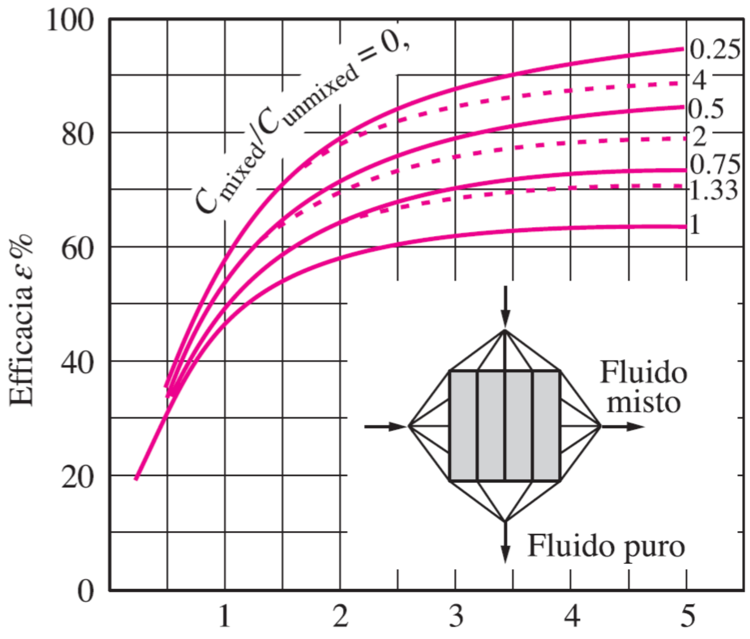


Figure 3.20: NTU for a cross-flow arrangement [15]

The NTU value is a function of ε, c^* and of the flow arrangement. Several charts are available in literature allowing NTU evaluation based on the aforementioned parameters, as the one presented in the figure 3.20. In the y-axis the ε value can be find, known, and some curves dependent on c^*; by intersecting these values the right value of NTU is obtained in abscissa, equal to 0,8. By applying two expression present in the Shah book procedure, NTU for both air side and water side are computed:

$$NTU_{water} = 10\,c\,NTU = 5{,}43 \tag{3.13}$$

$$NTU_{air} = 1{,}1\,NTU = 0{,}88 \tag{3.14}$$

3. Now, the core mass velocity is computed by solving the equation proposed by Kays and London [15]:

$$G = \left[\frac{2g_c}{\left(\frac{1}{\rho}\right) Pr^{\frac{2}{3}}} \frac{\eta_o \Delta p}{ntu} \left(\frac{j}{f}\right)\right]^{\frac{1}{2}} \qquad \left[\frac{kg}{m^2 s}\right] \tag{3.15}$$

where η_o is the fin's efficiency assumed equal to η_o=0,8, Δp is a design parameter and $\frac{j}{f}$ is the Colburn factor j divided by the Fanning friction f factor supposed to be equal to 0,25 for both sides as suggested by the procedure, ρ is the mean fluid density and Pr is the Prandtl number. The evaluation of G_w and G_a allows to compute the Reynolds number distinctive of both water and air flow conditions. In the following table all the parameters needed to compute these values are presented, assessed at the mean temperature value of 40 °C for the water and 20 °C for the air.

Parameter	Water	Air
μ $[Pa\,s]$	$6{,}53\,10^{-4}$	$1{,}81\,10^{-5}$
k $[\frac{W}{mK}]$	0,6	0,026
c_p $[\frac{J}{kgK}]$	4186	1005,5
ρ $[\frac{kg}{m^3}]$	992,2	1,205
Δp $[kPa]$	5	8,79
Pr [/]	4,56	0,70

Thus, the values G_w and G_a are obtained, equal to:

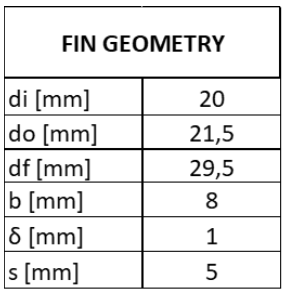
$$G_{water} = 407{,}62 \quad \frac{kg}{m^2 s} \tag{3.16}$$

$$G_{air} = 77{,}82 \quad \frac{kg}{m^2 s} \tag{3.17}$$

4. Finally, the Reynolds number for both sides are computed. The general expression that allows this is:

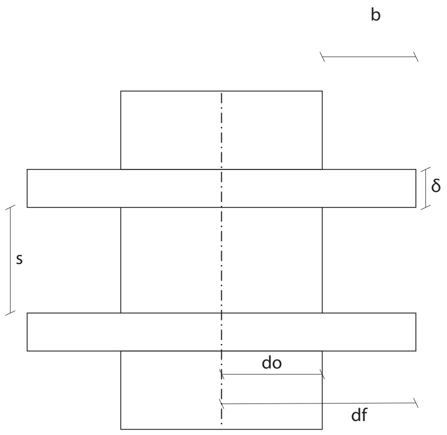
$$Re = \frac{G\,D_h}{\mu} \tag{3.18}$$

The only one unknown variable is the hydraulic diameter D_h for both sides. Its evaluation requires the knowledge of the geometrical features of the finned



FIN GEOMETRY	
di [mm]	20
do [mm]	21,5
df [mm]	29,5
b [mm]	8
δ [mm]	1
s [mm]	5

(a) Fin data geometry



(b) Fin scheme

Figure 3.21: Fin geometry

tube listed in the figure 3.21. Regarding the water side, the D_h is equal to the inner diameter of the tube, thus $D_h = D_i$.

Whereas for the air side, an expression taken from literature [17] to estimate the D_h has been taken in consideration:

$$D_h = \frac{[\frac{{d_f}^2 - {d_0}^2}{2s} + \frac{d_f \delta}{s} + d_o(1 - \frac{\delta}{s})]}{1 + \frac{(d_f - d_o)}{s}} = 24{,}57 \quad mm \tag{3.19}$$

Hence, the Reynolds numbers are:

$$Re_w = \frac{G_a \; D_{h_a}}{\mu_w} = 12484{,}43 \tag{3.20}$$

$$Re_a = \frac{G_a \; D_{h_a}}{\mu_w} = 105667{,}64 \tag{3.21}$$

5. The estimation of the heat transfer coefficients h_{inner} and h_{outer} related to the fluids outflow are now carried out; in order to do this, specific correlations between the dimensionless numbers `Re` and `Pr` must be applied, thanks to which the computing of the `Nu` may be done. Once `Nu` is known for both side, the heat transfer coefficients are computed by solving the following expressions:

$$h = \frac{Nu \, k}{D_h} \tag{3.22}$$

where k represents the thermal conductivity of the considered fluid. For the water side, since there is the motion of a liquid inside a circular pipe characterized by a Reynolds number higher than 10000, that means the flow's regime is turbulent, the *Dittus-Boelter* correlation has been used considering that the fluid is heated:

$$Nu_w = 0{,}023\ Re^{0{,}8}\ Pr^{0{,}4} = 68{,}61 \tag{3.23}$$

Hence, the h_{inner} is equal to:

$$h_{inner} = \frac{Nu_w\ k_w}{D_{h_w}} = 2058{,}31 \qquad \left[\frac{W}{m^2K}\right] \tag{3.24}$$

Regarding the air side instead, an appropriate correlation has been applied in order to consider the external outflow on a finned-tube that takes in consideration the presence of fins on the outer surface of the pipe proposed by Young and Bring and reported in [18].

The numerical computed value for Nu in this case is:

$$Nu_a = 0{,}134\ Re^{0{,}681}\ Pr^{\frac{1}{3}} \left(\frac{s-\delta}{b}\right)^{0{,}20} \left(\frac{s-\delta}{\delta}\right)^{0{,}113} = 319{,}48 \tag{3.25}$$

Even here, once `Nu` is known, the value of h_{outer} is:

$$h_{outer} = \frac{Nu_a\ k_a}{D_{h_a}} = 337{,}98 \qquad \left[\frac{W}{m^2K}\right] \tag{3.26}$$

6. Now, the fin's efficiency η_f and the overall efficiency η_o are computed. The expressions that allow this are:

$$m = \sqrt{\frac{2\ houter}{\delta\ \lambda_{fin}}} = 111{,}88 \qquad \eta_f = \frac{tanh(m\ b)}{m\ b} = 0{,}797 \tag{3.27}$$

where b and δ are the same exposed in the table 3.21a and $\lambda_{fin} = 54\ \frac{W}{mK}$ represents the thermal conductivity of the material of which the finned tube is built, or rather steel. By imposing a ratio between the fins surface and the total outer surface $\frac{A_f}{A} = 0{,}5$ η_o is computed as:

$$\eta_o = 1 - (1-\eta_f)\ \frac{A_f}{A} = 0{,}898 \tag{3.28}$$

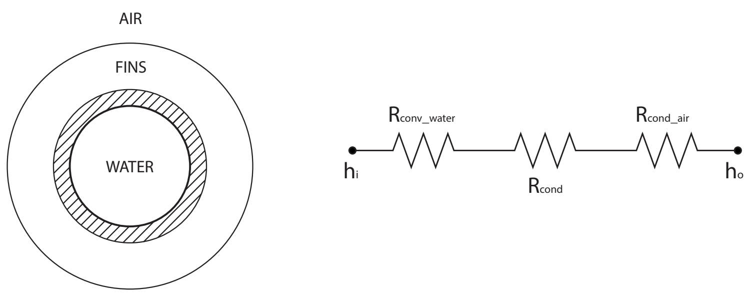


Figure 3.22: Transmittance scheme

7. Once the heat transfer coefficients for both sides, together with the geometrical and thermal features of the fins and of the tube, are known the calculation of the total transmittance can be done; by referring to the figure 3.22, it can be written:

$$U_{inner} = \frac{1}{R_t\ A_{inner}} = U_{outer} = \frac{1}{R_t\ A_{outer}} \tag{3.29}$$

$$U_{inner} = \frac{1}{\frac{1}{h_{inner}} + \frac{\ln(\frac{d_o}{d_i})\ A_{inner}}{2\ \pi\ k\ L} + \frac{A_{inner}}{A_{outer}\ h_{outer}\ \eta_o}} \tag{3.30}$$

By replacing $A_{inner} = \pi L D_i$ and by imposing a ratio between the inner and outer surfaces equal to 0,2, the computed value for the total inner transmittance U_{inner} is:

$$U_{inner} = 808{,}45 \qquad \left[\frac{W}{m^2K}\right] \tag{3.31}$$

8. At the end, by knowing the transmittance value, the length L of the tube needed to ensure the right heat transfer is evaluated by the definition of NTU considering the one related, and already computed, to the water side, since we know U_{inner}:

$$A_{inner} = \frac{NTU_W\ Cmin}{U_{inner}} = 1{,}68\ m^2 \tag{3.32}$$

$$A_{inner} = \pi L D_i \qquad L = \frac{A_{inner}}{\pi D_i} = 26{,}86\ m \tag{3.33}$$

The procedure ends with the finned-tube length's evaluation, whereas the dimensions of the casing may be chosen by the manufacturers catalogue depending on the effective footprint of the tube, which is bent to form the coil.

The finned tube heat exchanger has been sized following the procedure contained in the Shah book, albeit with some simplifications, and by respecting the constraints imposed by the case study. Now, it is possible to proceed with its modeling on SystemModeler.

3.4.3 Fan coil modeling

By following the sizing procedure and applying all the results obtained, a representative model of the fan coil thermal behavior has been built. The figure below shows the complete model developed on SystemModeler in its diagram view:

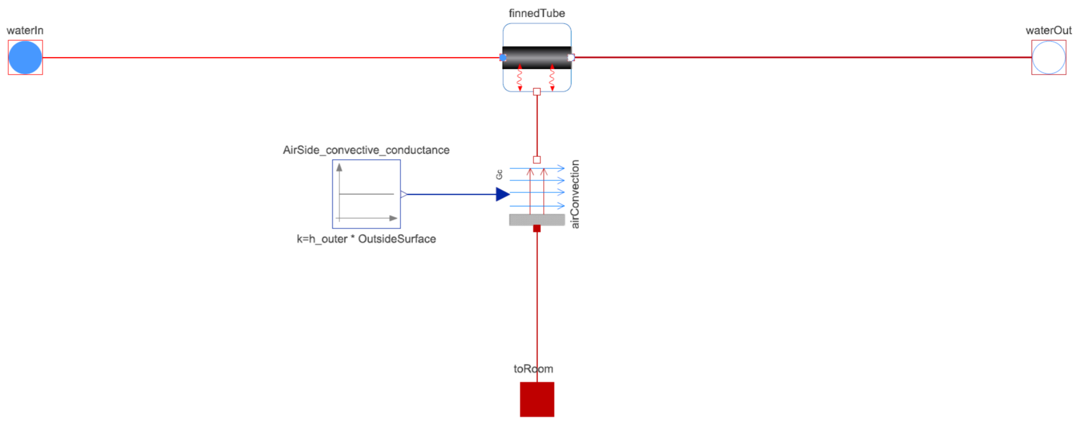


Figure 3.23: `Fan Coil` model diagram view

As can be seen, there are basically two main components connected each other: `finnedTube` and `AirConvection` represented by their icon view.

The first one is related to the heat transfer that occurs inside the tube due to the flowing of a specific water volume flow rate. This heat transfer consists on forced convection between water and the pipe due to the water motion inside it, and on the conduction across the pipe material even if it is negligible with respect of the convection contribution. Such mechanisms are implemented on SystemModeler with the creation of the aforementioned `finnedTube` class representing the manner in which the inner part of the finned tube exchange heat, by the connection of thermal components, some already known and other introduced now for the first time, as shown in the figure 3.24.

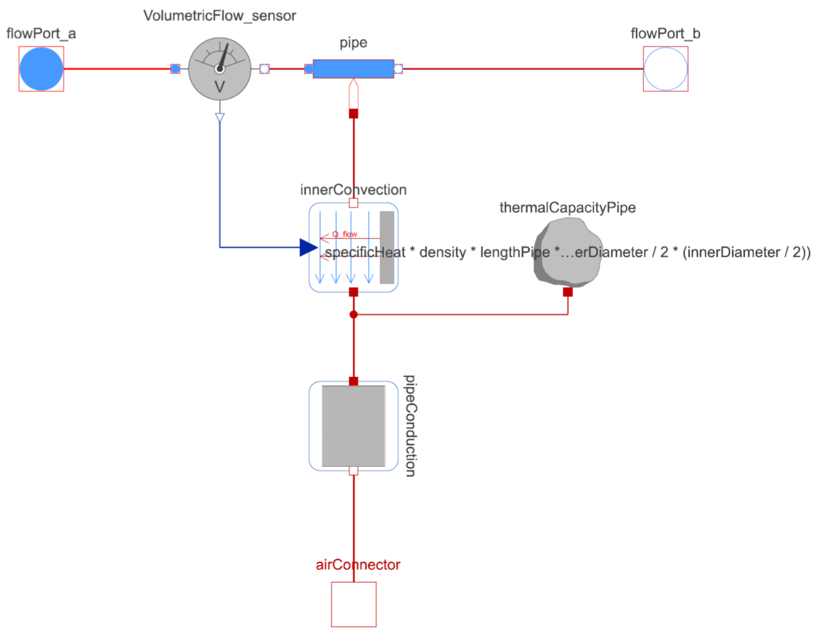


Figure 3.24: `finnedTube` model diagram view

On the top it can be observed a component named **pipe** that physically represents the tube inside which flows the water, connected to a volumetric flow sensor that sends the volume flow rate measured to the `innerConvection` model to compute the velocity of the water flux and so the *Reynolds* and so the *Nusselt* using the *Dittus-Boelter* correlation to finally calculate the h_{inner}. The text view of the model that explain all the relationship among all the variables for the inner convection is presented in the Appendix (A.2). These two components present two connectors (pins) that are different respect to the already used and are called FLOW PORT, one white and one blue as can bee seen in the figure, necessary to model the manner in which the heat is transported by a fluid, thus it allows to consider the fluid motion and the corresponding heat transfer. In the pipe component there is also the possibility to activate a `heatPort` connector that allows to link it to the `innerConvection` component representing the thermal modeling of the convection inside the tube, which in turn is connected with the element `PipeConduction` that models the conduction across the pipe material. The capacity of the pipe material to retain heat is taken into consideration by utilising the customary component `heatCapacitor`.

Thanks to these connections, the thermal power possessed by the hot water flowing inside the tube passes through all the components contained in the `finnedTube` model and arrives to the thermal connector `airConnector` that allows the connection with the the second component of the fan coil model `airConvection`, where is modelled the convection between the `finnedTube` and the surrounding area using as convective heat transfer coefficient the one computed in the sizing procedure. In fact, returning to the diagram view of the fan coil model 3.23, we can note how the two main components are thermally connected each other through their thermal pins.

Hence, in summary, the hot water comes from the flow port `waterIn` and reaches the component `finnedTube` that has only one thermal connector (the white one) with which the connection with `airConvection` model is realized by linking the two white pins. Its red pin is then linked with a thermal port called `toRoom`, that will allow to transport the heat power possessed by the water to the air of the room with an increasing of its temperature due to the heat exchange mechanisms modelled. The exhaust water flows through the white flow port `waterOut` after the heat exchange is happened.

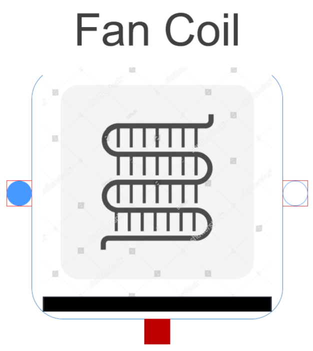


Figure 3.25: `FanCoil` model icon view

At this point the fan coil model is ready and it is graphically represented by its icon view; by observing it in figure 3.25 we can see its two flow pins through which the water is injected and its red thermal connector `toRoom` that allows the thermal connection of the fan coil model with the room it has to warm.

Developing the fan coil model allows to equip the already realized `GeneralRoom` model, and to create with extremely ease the `GeneralRoomWithFanCoil` model graphically represented in the figure 3.26. It represents an extension of the previous one; the just developed fan coil model is connected to the `heatCapacitor` component through the red thermal connector and presents also the two flow port which then will allow the connection with the water distribution network.

Now, the room contains a thermal device that may rise up its air temperature to ensure always the right comfort. This model will be resumed even to better explain its operating in the assembly of the top model.

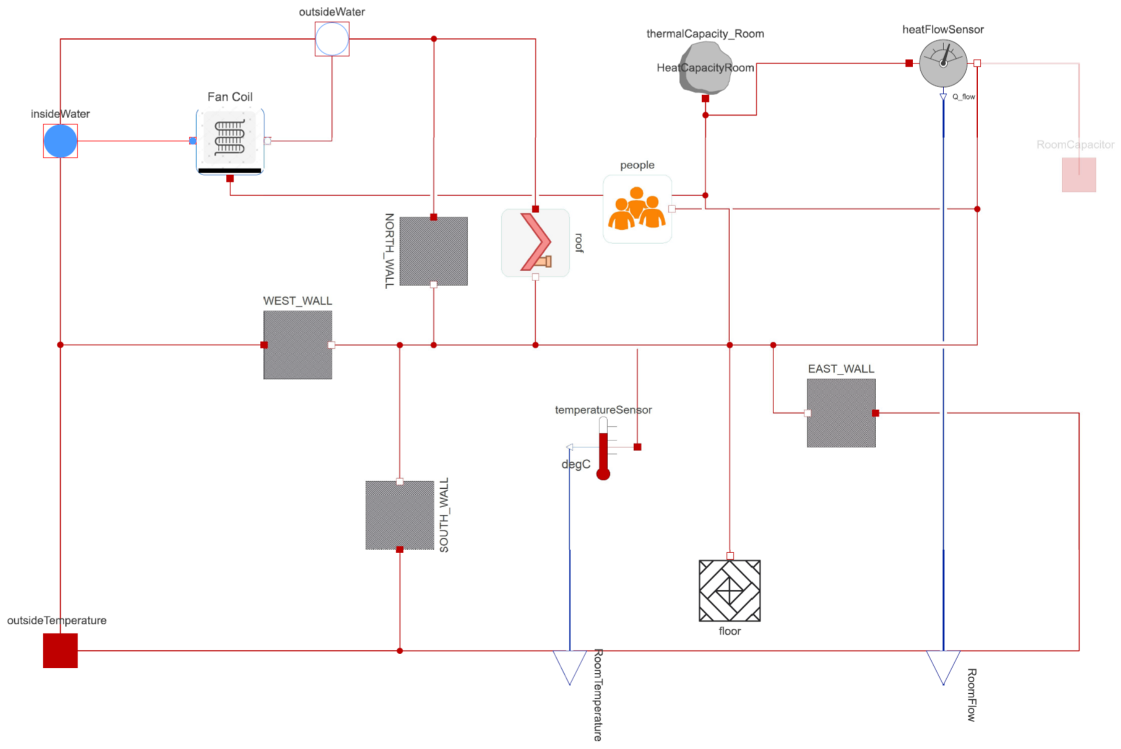


Figure 3.26: `GeneralRoomWithFanCOil` model diagram view

3.5 Distribution network

The third sub-system we need to model for the AC system is that related to the water distribution network. It may be considered the junction component between the heat generator, the heat pump, and the apartment, providing the hot water warmed by the heat produced in the condenser of the heat pump to each fan coil presents in every room. It basically consists of two different circuits, one for the delivery and one for the return of the water. The water runs through the delivery network, providing thermal energy to the fan coils to ensure the desired comfort for the house; then it comes out along the return network and the cycle is repeated. To make sure that all the system works well, it is necessary to carry out a good sizing procedure in order that the right amount of water for each device is injected, to guarantee the wellness conditions. First of all, the layout of these networks needs to be defined, the length of each pipe both for the delivery and the return; the figure 3.27 shows pipes layout while the table presented in figure 3.28 contains the total guessed lengths for every pipe. Now follows the sizing procedure that has as purpose the estimation of the total mass flow rate of water required and of the total head losses for the pipes calculation that helps to choose the right circulating

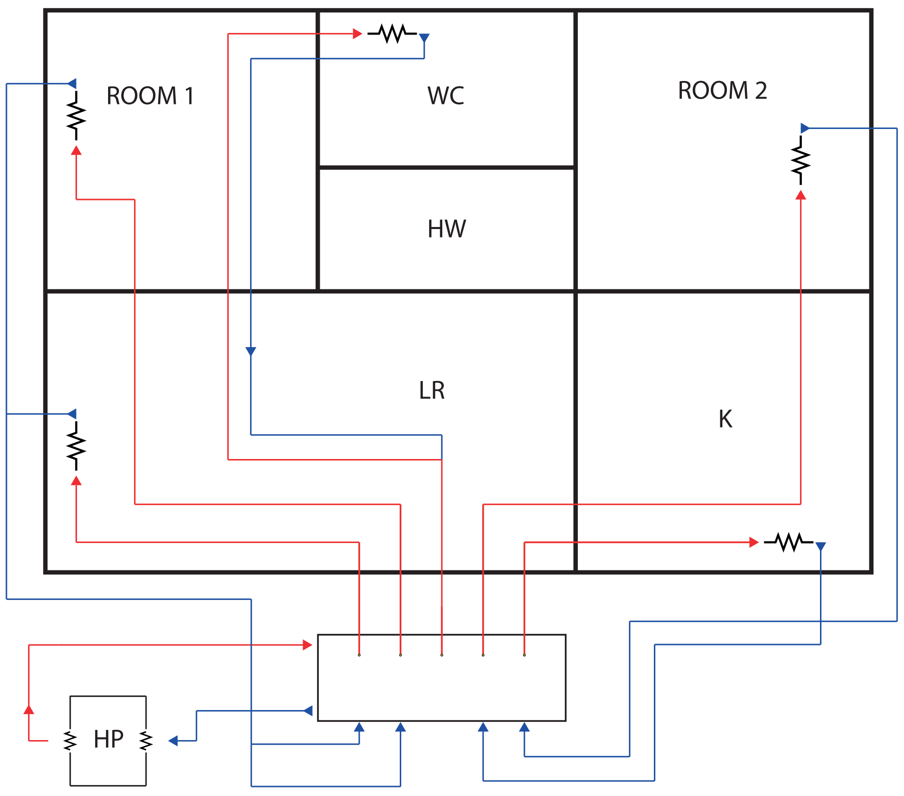


Figure 3.27: Pipes layout

pump for the system.

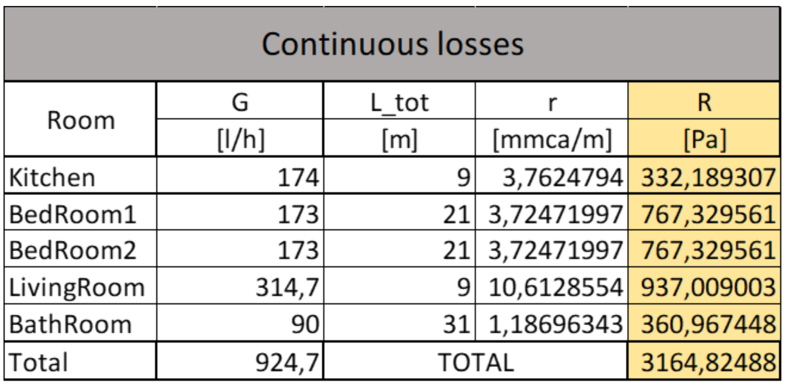
3.5.1 Network sizing

The essential quantities for the sizing of a distribution network are the flow rate and the head; the first represents the total quantity of water flowing into the various branch of the network, measured in liters per hours. The head is the pressure of the water in a certain section of the circuit, measured in kPa. The goal is to compute the dispose of water flow rate and the total head losses. Regarding the water flow rate, let's consider the maximum value, the design one, of the thermal power required by each room imposed by the case study 3.6. Knowing them, is it possible to compute the needed mass flow rate expressed in $\frac{kg}{s}$ by applying the energy balance:

$$\dot{m}_w = \frac{\phi_{room}}{cp_w \; \Delta T} \tag{3.34}$$

where ϕ_{room} is the design thermal power value of the considered room, $cp = 4186 \, \frac{J}{kgK}$ the eater specific heat, and ΔT is the guessed difference of temperature that should occur in each fan coil, equal to $10°C$.

In the following table all these computed values are summarized.



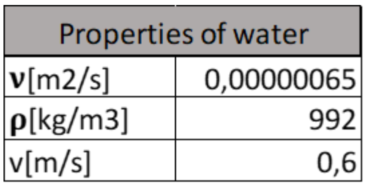
Continuous losses				
Room	G	L_tot	r	R
	[l/h]	[m]	[mmca/m]	[Pa]
Kitchen	174	9	3,7624794	332,189307
BedRoom1	173	21	3,72471997	767,329561
BedRoom2	173	21	3,72471997	767,329561
LivingRoom	314,7	9	10,6128554	937,009003
BathRoom	90	31	1,18696343	360,967448
Total	924,7	TOTAL		3164,82488

Figure 3.28: Continuous losses

Once the total water flow rate is known, under the assumption of using copper ducts with values of external and internal diameter equal respectively to 20 and 18 mm, by using diagrams proposed in the Caleffi handbook [16] a value for the velocity of the water flow equal to 0.6 $\frac{m}{s}$ is assumed for the computation of both the distributed and concentrate losses. The distributed losses are caused by the friction between the water and the inner part of the ducts; for their calculation an expression taken from the Caleffi manual regarded copper ducts with low roughness is used:

$$r = 14{,}7 \; \upsilon^{0{,}25} \; \rho \; \frac{G^{1{,}75}}{D^{4{,}75}} \qquad \left[\frac{mmc.a.}{m}\right] \tag{3.35}$$

where υ is the kinematic viscosity while ρ the density and their values are presented



Properties of water	
ν[m2/s]	0,00000065
ρ[kg/m3]	992
v[m/s]	0,6

Figure 3.29: Properties of water

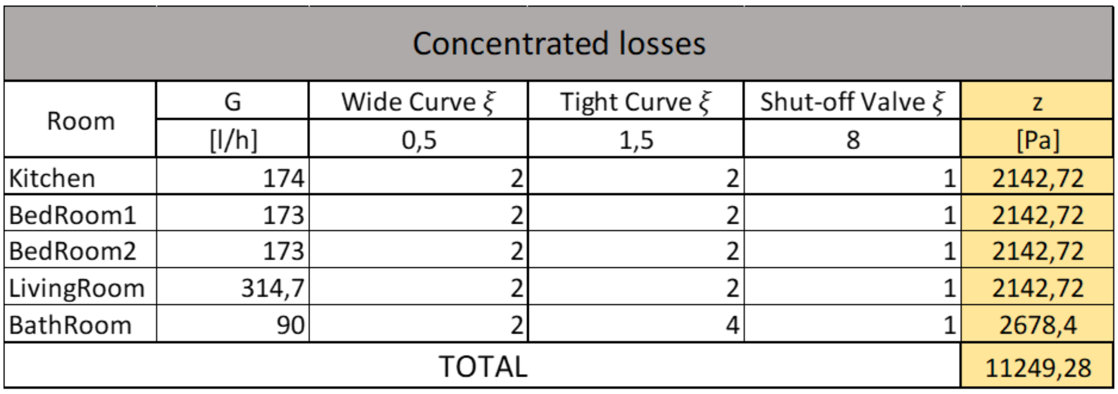
in the figure 3.29.

Then, by multiplying it for the length of each branch and for 9,81 $[\frac{m}{s^2}]$ in order to perform the conversion, the values expresses in Pa are obtained and presented in the table 3.28

Subsequently we can proceed with the estimation of the concentrated losses entity: their presence is due to the obstacles that the water meets in its road as hydraulic devices (valves) or for the changes of the network geometry. For their computation the following expression from Caleffi is used:

$$z = \xi \; \rho \; \frac{v^2}{2} \qquad [Pa] \tag{3.36}$$

where the parameter ξ depends on the shape of the localized loss: the figure 3.30 shows the assumptions made for these changes in the network together with the related values of ξ taken from Caleffi handbook, together with the numerical values obtained.



Concentrated losses					
Room	G	Wide Curve ξ	Tight Curve ξ	Shut-off Valve ξ	z
	[l/h]	0,5	1,5	8	[Pa]
Kitchen	174	2	2	1	2142,72
BedRoom1	173	2	2	1	2142,72
BedRoom2	173	2	2	1	2142,72
LivingRoom	314,7	2	2	1	2142,72
BathRoom	90	2	4	1	2678,4
TOTAL					11249,28

Figure 3.30: Concentrated losses calculations

Finally, the total head losses is obtained as the sum of the two computed:

$$Total\ head\ losses = 3164{,}82 + 11249{,}28 = 14414{,}10 \quad Pa \tag{3.37}$$

This parameter is used to build the characteristic curve of the circuit and may be overlapped to the one of the pump to choose its right size. Now that we have all the operating and constructive aspects of the network, let's proceed with its modeling.

3.5.2 Distribution network modeling

The implementation of the distribution network on SystemModeler predicts the realization of two different models, or classes, representative of the two different

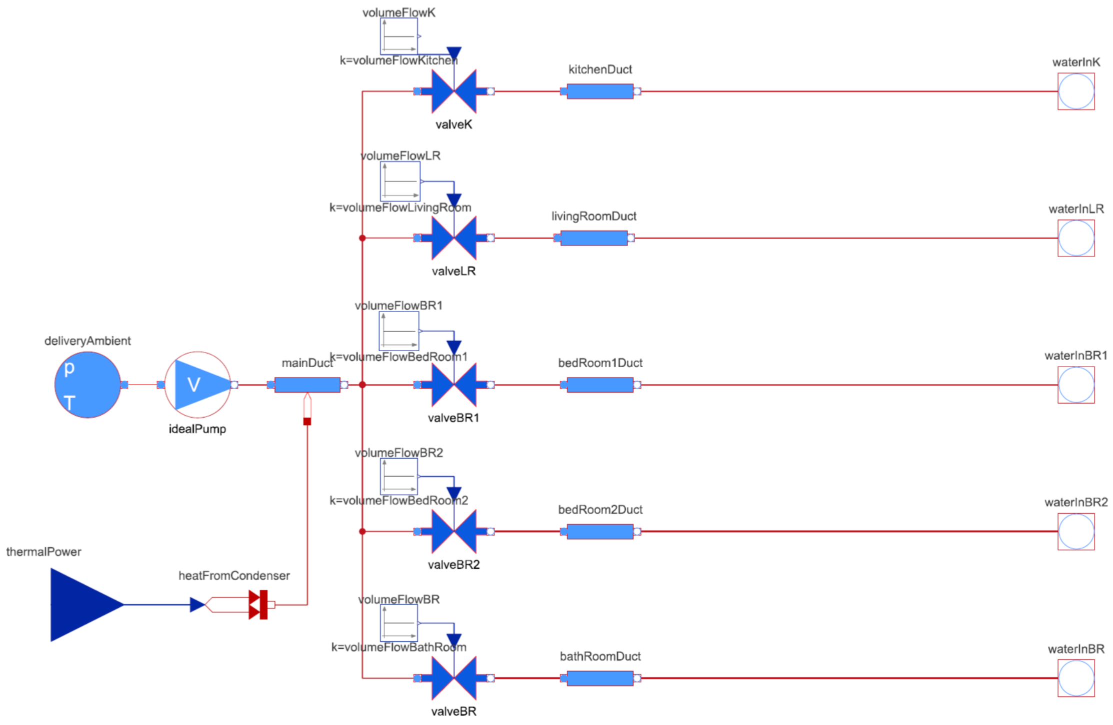


Figure 3.31: `deliveryNetwork` model diagram view

circuits, delivery and return, needed to build the correct network for the water flowing from the heat pump to every fan coils. Let's start by the one related to the delivery network. The picture 3.31 shows the diagram view of the total model `deliveryNetwork`; as can be seen there are five `pipe` models, one for each room, that thermally represent the delivery ducts that allow sending the hot water to each fan coil. They are fed by another pipe under the name `mainDuct` that is directly connect to the model of an ideal pump that allows the circulation of the water coming from the `deliveryAmbient` where a constant value for both the temperature and pressure are defined. The `mainDuct` model presents a thermal input connector essential for the heat moving; in fact, it is connect with a `PrescribedHeatFlow` under the name `heatFromCondenser` component allowing a specified amount of heat flow to be *injected* into a thermal system at a given port that corresponds to the aforementioned thermal input connector of `mainDuct` model. The numeric value of this heat flow is given by the blue arrow input named `thermalPower`, coming from the heat pump model.

Since the various rooms of the considered apartment are different in terms of dimensions, each of them needs a different amount of thermal power or rather a different value of water flow rate injected into its relative fan coil; these values have been already computed for the situation imposed by our case study in the discussion about the circuit sizing and are shown in the figure 3.28. The total required water flow rate is given as input in the `idealPump` model and by default it is divided to each branch in equal parts; to force the value to the right one for each branch, valves are used. By imposing a constant signal equal to the right amount needed for each valve model related, the required water flow rates by each room for its warming are correctly delivered.

To make the models more realistic, the actual mass of each duct is considered, computed knowing their length using the expression:

$$m = \rho\ V = \rho\ \pi({d_e}^2 - {d_i}^2)\ L \tag{3.38}$$

where $\rho = 8930\ \frac{kg}{m^3}$ is the copper density, whereas the diameters and length values have already been introduced in the previous section. By inserting this equation to the related parameter presents in the parameters tab of each pipe model, we are able to consider the effective mass of the ducts as well.

The water coming from the fan coils, after the heat transfer in order to warm the room has occurred, flows into the delivery ambient into the `returnNetwork` model, graphically represented in the figure 3.32. It can be noticed that basically it is structurally equal to the delivery except to the lack of the valves and the pump. The mass of the ducts has been considered for the return circuit as well.

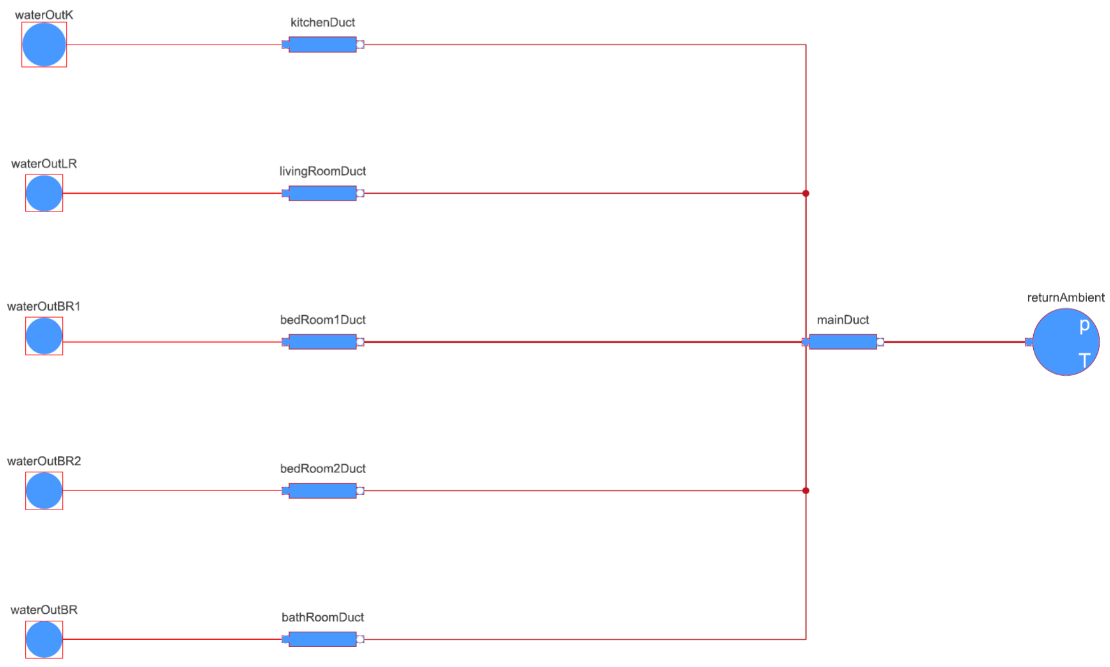


Figure 3.32: `returnNetwork` model diagram view

3.6 Heat Pump

After thermally modeling the building that needs to be heated, the thermal devices that must provide the required thermal power and the water distribution network for transport the heat transfer fluid in each local of the house, now it's the turn of the heat generator that must supply with the right amount of thermal energy all the fan coils of the system. As already mentioned in the introduction, the heat is generated thanks to the using of a heat pump system. Its operating is based on the vapour compression of a coolant fluid and on the realization of a *reverse* refrigeration cycle that this fluid runs to get the desired effect.

There are essentially two type of heat pump adapt to an air conditioning system in a building: the *air-to-air* and the *air-to-water* heat pump; in this thesis activity, the second kind has been considered, since using the external air as energy source is extremely convenient and the equipment required are simpler compared with the utilising of water or the ground as heat source. Let's briefly expose the heat pump operating; the figure 3.33 shows a simple diagram where we can note the main components in which it consists, necessary in order to realize the refrigeration cycle: a compressor, evaporator, condenser and a throttling valve.

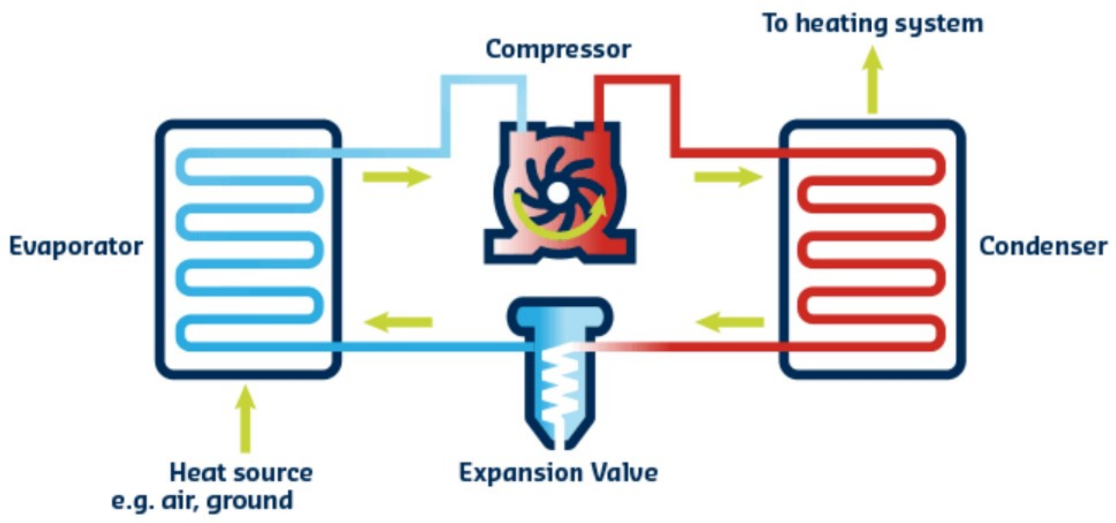


Figure 3.33: Heat pump scheme. Source [19]

These four basic components are separated into two section, indoor and outdoor, depending on which is the heat pump mode operating In the heating mode, the only one considered in this thesis, the outdoor section consists of the expansion valve and the evaporator representing the low pressure part of the system. The indoor section consists of the compressor and the condenser representing the system's high pressure side. Heat energy, made available by the external air, is absorbed in the outdoor unit by the evaporator, where the coolant is in form of gas in balance with its liquid; this heat absorption turn the coolant in gas, a cold gas. By operating the compressor, the gas pressure increases and also its temperature, turning into hot gas. This hot gas exchanges heat with the water flowing into the condenser, that is later send to through the distribution circuit to the fan coils. This heat transfer causes the coolant gas condensing into warm liquid; the presence of the throttling valve allows then to restore the pressure and the temperature of the liquid at the outlet of the condenser to the values which reigns into the evaporator, in order to start again the cycle.

One of the main feature of a heat pump system is the possibility to realize also the cooling mode, simply using a reverse valve that reverses the coolant flow; the outdoor coil is now the condenser while the indoor coil has the role of evaporator. This is why the heat pump is a totally versatile and efficient heating and cooling system.

The efficiency of a heat pump is measured by its **COP** representing the ratio between the amount of heat administered to the hot source and the work spent to do this. This performance indicator depends by the temperature jump between the evaporator and the condenser: higher is this jump, lower the **COP** value will be. For this reason, it is necessary using plant with low delivery temperature, such as fan coils or underfloor radiant panels. Indeed, the pressure that we must generate in the condenser depends on the temperature reigning into the condenser; so the power absorbed by the condenser increases with increasing of the delivery temperature, reducing the **COP** value and therefore the system efficiency. For the same reason, the temperature of the cold source should be the highest possible. In fact, if the evaporation temperature decreases, the pressure of the vapour inside the evaporator decreases as well and together with these the useful power that it is capable to provide. Let's now size the refrigeration cycle considering the situation imposed by the case study.

3.6.1 Refrigeration cycle sizing

In order to do this, the **R134a** coolant has been used. We have to define and fix the thermodynamic cycle that the **R134a** must perform by defining in a $p - \log h$ chart the coordinates of the cornerstones considering the conditions in which the system works.

Since the worst temperature considered as design parameter is equal to -5°C, the evaporation temperature must be lower to allows the absorption by the coolant of the thermal energy possessed by the air; so it is set equal to -10°C. Similarly, since the delivery temperature of the water at the outlet of the condenser to the fan coils is guessed to be equal to 45°C, in order to make sure that the heat possessed by the coolant in the condenser is transferred to the water, the **R134a**'s condensation temperature must be higher, thus set to 50°C. For reasons of simplification, the transformation occurring in the condenser is considered to be isentropic, thus with a isentropic efficiency equal to 1.

After imposing these design restrictions, the cycle is graphically constructed thanks to the software **CoolPack** that allows to plot this specific situation in a Mollier chart, shown in the following figure: Furthermore, **CoolPack** provides the

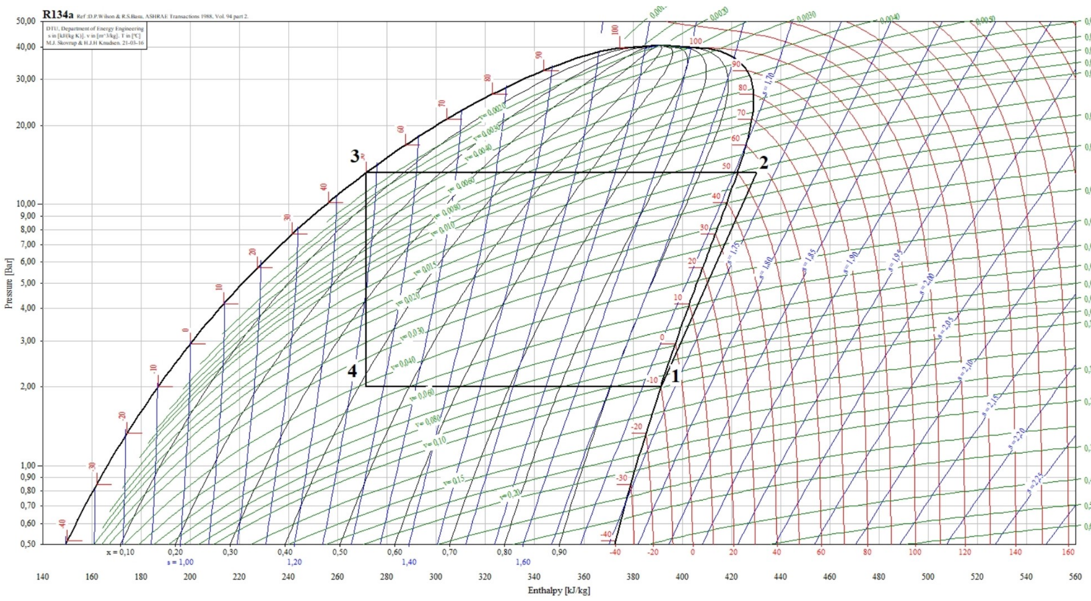
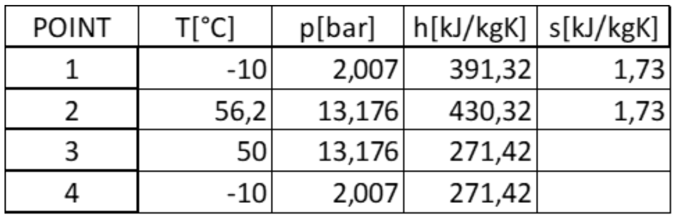


Figure 3.34: Considered cycle for the case study

values of the thermodynamic properties for the main points of the cycle, collected in the table 3.35



POINT	T[°C]	p[bar]	h[kJ/kgK]	s[kJ/kgK]
1	-10	2,007	391,32	1,73
2	56,2	13,176	430,32	1,73
3	50	13,176	271,42	
4	-10	2,007	271,42	

Figure 3.35: Cornerstones values

The cycle starts in the point 1, where the coolant is in form of saturated vapour; the transformation 1-2 is the isentropic compression, that means that the curve is plotted following the isentropic one related to a value of entropy equal to 1,73. In 2, the coolant has been compressed in order to increase its pressure up to the one imposed by the condensation temperature; consequently its temperature increases as well, obtaining a super-heated vapour. The transformation 2-3 is the condensation; here the sensible heat of the hot super-heated vapour is rejected. Then it starts to changing phase from vapor to liquid until it becomes 100 % saturated liquid refrigerant due to the condensation process at constant pressure; The point 3 represents the refrigerant's state at the outlet of the condenser. In the 3-4 transformation, the fluid crosses the throttling valve passing to the low pressure and reducing its temperature, but keeping the enthalpy value equal to the one it had in the point 3 being the transformation isenthalpic; the liquid, being not compressed anymore, returns in a vapor state. Thus, in the point 4 we have a mixture of vapor and liquid at low pressure and low temperature, ready to enter into the evaporator to starts the cycle again.

Once all the points are known, the required mass flow rate of **R134a** for the limit situation may be computed making a energy balance for the condenser:

$$\dot{m}_{r134a} = \frac{\phi_{tot}}{h_2 - h_3} = 0{,}1\ \frac{kg}{s} \tag{3.39}$$

where ϕ_{tot} is the total thermal load needed by the house in the worst climate condition considered increased by 20 % to be more conservatively.

3.6.2 Heat pump modeling

Such situation is implemented on SystemModeler by the creation of the refrigeration cycle model, shown in the figure 3.36. There are basically the models of the compressor, evaporator and condenser capable to provide the values of the quantities related to them thanks to the equations they have inside. All the related codes with the equations of each component are presented in Appendix (A.3) (A.4) (A.5).

The enthalpy values are obtained thanks to the presence of the sub-package `CoolantProperties`, depicted in the figure 3.37, containing the models that allow to extrapolate the needed values from some tables simply introducing the evaporation temperature or the condensation pressure values to their parameters tab.

In fact, by observing the figure 3.38 the `satTemperature` component provides the saturation temperature by giving it the condensation pressure value; this allows to enter in the table containing the saturation properties of the considered coolant, **R134a** in this case, and provides the saturation temperature related to that pressure value. Next with the same logic, the component `saturatedLiquid` provides the enthalpy of the saturated liquid at that value of temperature that is fixed and enters into the evaporator model, by remembering that the enthalpy at

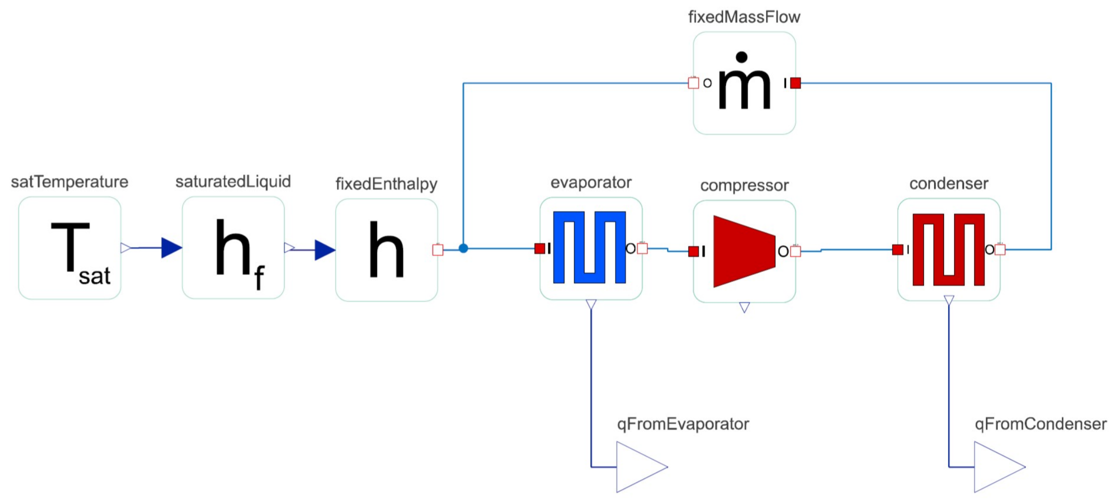


Figure 3.36: `RefrigerationCycle` model diagram view

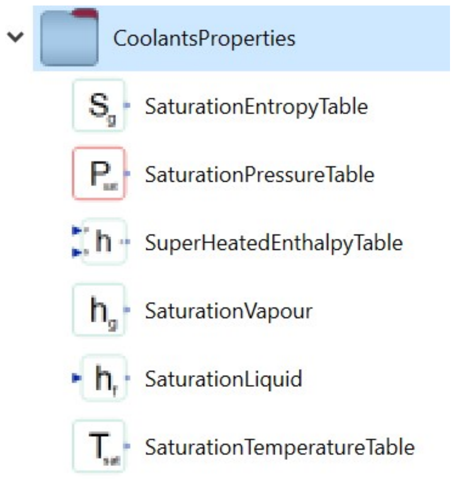


Figure 3.37: Coolant properties models

the inlet of the evaporator is the same to the one of the saturated liquid at the condenser outlet. This value is sent to the evaporator model and the complete model starts working. A constant value of the coolant mass flow rate is introduced by using the `FixedMassFlow` component. All the others values of enthalpy needed for the compressor and the condenser are get thanks to the models contained in the aforementioned `CooantProperties` sub-package containing the tables both for the saturation properties and for the super-heated vapor, where there is also the possibility to change the tables with those of another refrigerant.

Thus, giving as input the values of mass flow rate, evaporation temperature,

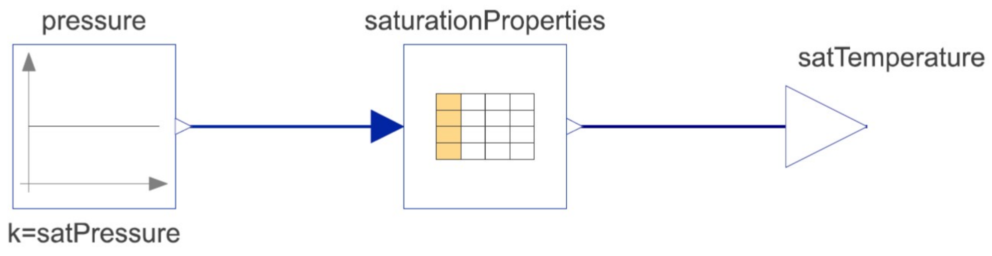


Figure 3.38: `satTemperature` model diagram view

condensation pressure, and after setting the refrigerator in all the models of the `CooantProperties` sub-package, that provide us the right values of entalphy for all the components, the cycle is defined and the thermal power values related to the evaporator and condenser are provided as output.

Once the `RefrigerationCycle` model is ready, it can be drag and drop to another class in order to create the real `heatPump` model, graphically represented by its diagram view in figure 3.39.

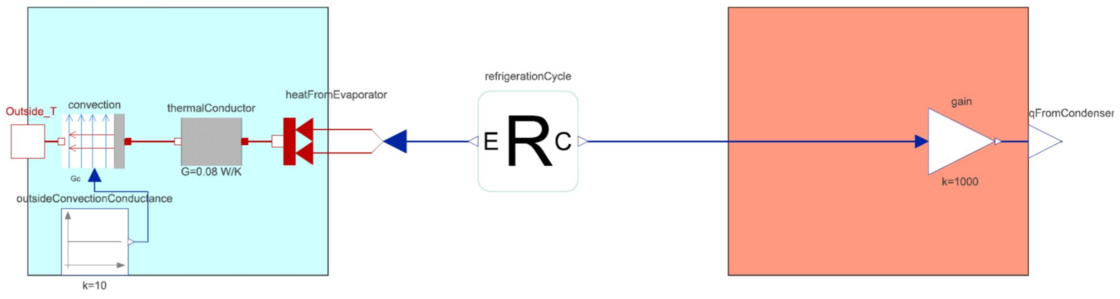


Figure 3.39: `heatPump` model diagram view

The heat produced in the condenser is sent to a gain simply to turn it in Watt, since in the outgoing by the `RefrigerationCycle` is in kWatt, and it is made available as output thanks to the presence of the white `qFromCondenser` connector. On the other side, the external temperature signal is sent to the heat pump model through the `Outside_T` thermal pin.

Finally, the heat pump model is ready to be used, as we will see soon in the creation of the top model.

3.7 Overall model

In the previous sections the subsystems needed to realizing the complete AC heating system model have been developed and largely discussed in their every aspect.

Therefore now, simply dragging and dropping all these component representing by their icon view and by connecting each in the right way, the top model of the whole considered scenario inherent to heating a building using a heat pump system has been implemented on SystemModeler. The diagram view of the entire system may be observed in the figure below:

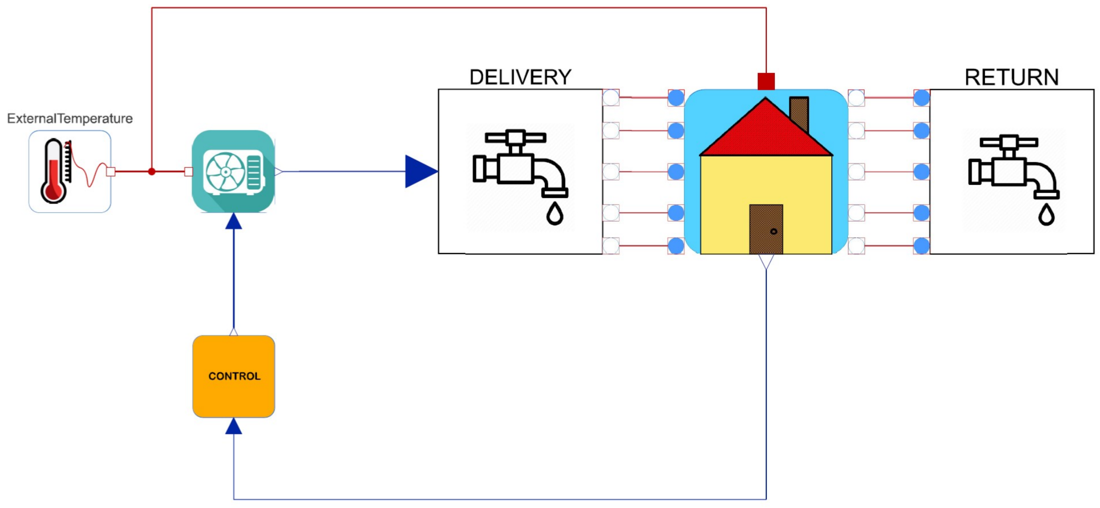


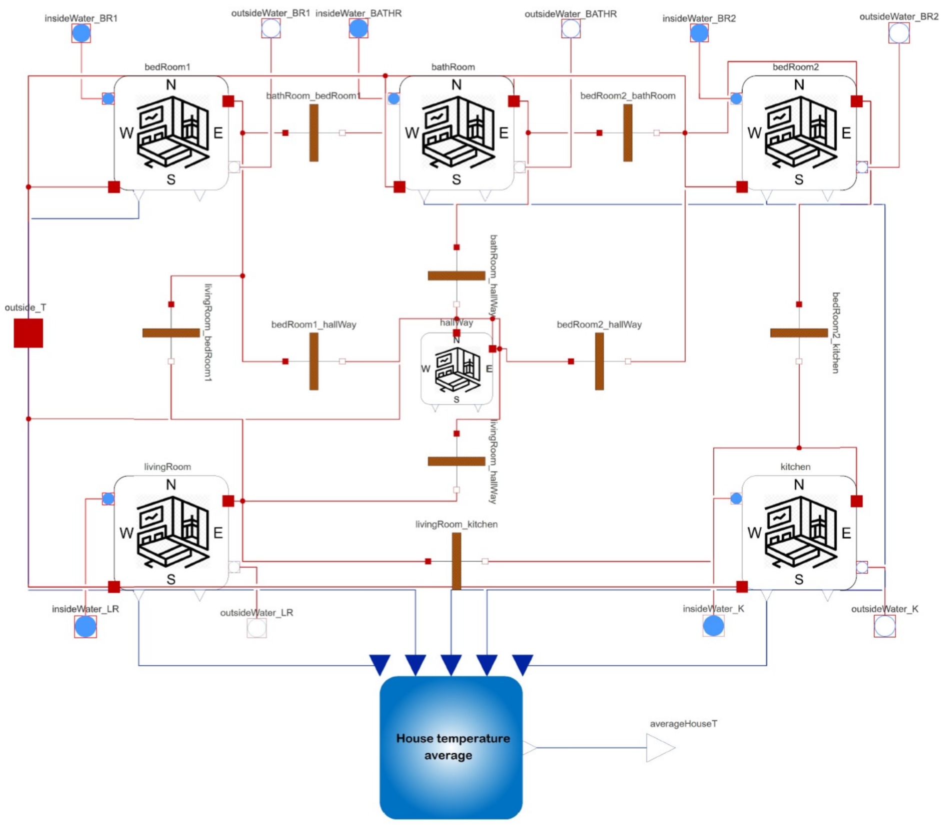
Figure 3.40: `ACHeatingSistem` model diagram view

Each of the visible element in the top model will be briefly discussed to explain their working when they have to operate all together discussing interactions occurring among each other as well.

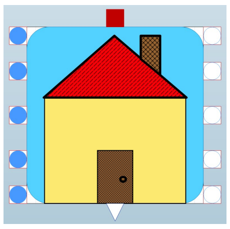
3.7.1 Apartment with fan coils

As mentioned in fan coils modeling section, developing `GeneralRoomWithFanCoil` model has been possible containing the fan coil model that allows to make available the thermal power required by the room to get and keep the desired comfort. As result, a new model named `ApartmantWithFanCoil` has been created that differs from the one presented simply in the Apartment section 3.3 only for the presence of that new room model, instead of the `GeneralRoom` model which doesn't present any thermal device. Thus, `ApartmantWithFanCoil` is an extension of the previous one. Doing so, each room of the apartment presents in their icon view two flow ports, as can be seen in the figure 3.41, allowing the injection on them the required hot water to make possible the fan coil's operating and ensure the heat transfer.

Furthermore, it presents a component called `house temperature average` which provides the average room temperature of in the house. This value is



(a) Diagram view



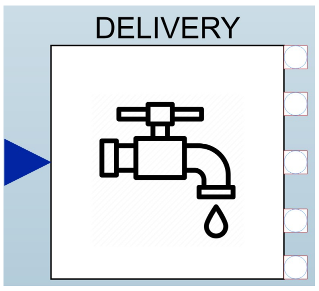
(b) Icon view

Figure 3.41: Graphical representation of `ApartmentWithFanCoil` model

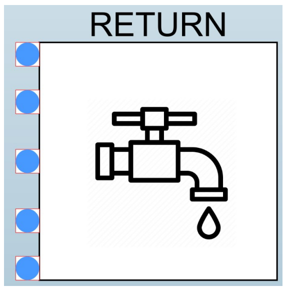
sent as output thanks to the triangular connector `averageHouseT` visible in the icon view 3.41b on the bottom, while on the top we can find a red thermal port to connect the external temperature signal to every single room. Always by observing the icon view, representing the whole `ApartmantWithFanCoil` model, on the left there are five blue flow ports that allow the connection of each pipe of the delivery network to their respective fan coils, whereas similarly on the right the white ones connect them to the return network.

3.7.2 Delivery and return networks

Let's explain now the operating of the water distribution circuit when it has to work together with the other components of the system.



(a) `deliveryNetworkIcon` view



(b) `returnNetworkIcon` view

Figure 3.42: Icon views of the distribution networks

In the figure 3.42 are presented the icon views of both delivery and return model, while the diagram views are not here presented since they are exactly the same of those already presented in their dedicated section 3.5. Substantially, the water is heated by the heat produced in the heat pump which is sent from its related model to the delivery network as input value thanks to the `thermalPower` connector visible both on the left of the icon view 3.42a and in the already seen diagram view 3.31. The warmed water crosses every pipe and reaches all the fan coils by connecting the `deliveryNetwork` flow ports with the already mentioned apartment flow ports.

Doing so, the water may enter in each fan coil making available the heat it posses and then the outgoing water returns to the return network, which is connected to the apartment model exactly how the delivery one does. These connections may be graphically appreciate in the figure 3.40

3.7.3 Heat pump

Once the apartment model has been connected with the water circuit distribution, the water needs to be heated before entering into every fan coil; the required heat is given by the heat pump model. As can be seen by its icon view 3.43, it presents three different connector; the white thermal port on the left allows to send the outside temperature signal to the evaporator in order to realize the refrigeration cycle, while the triangular one on the right side is that correlated with the heat produced by condenser that allows its injection to the delivery network in order to the water heating, how it is appreciable by observing again the top model scheme.

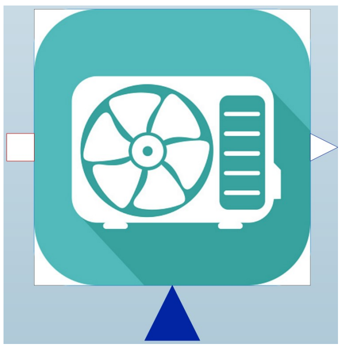


Figure 3.43: `HeatPump` model icon view

The third on the bottom is related to the control signal, that will allow to modulate the heat pump's operating to follow and keep the temperature reference and obtain the desired comfort inside the building; let's see how.

3.7.4 Control logic

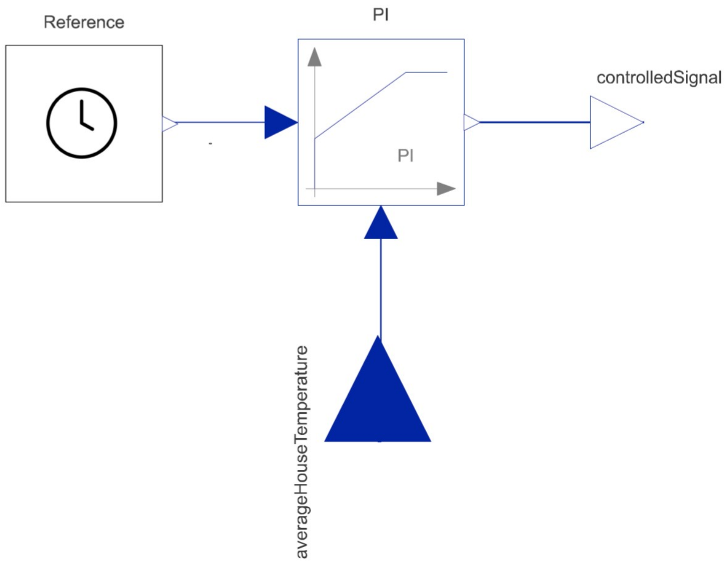
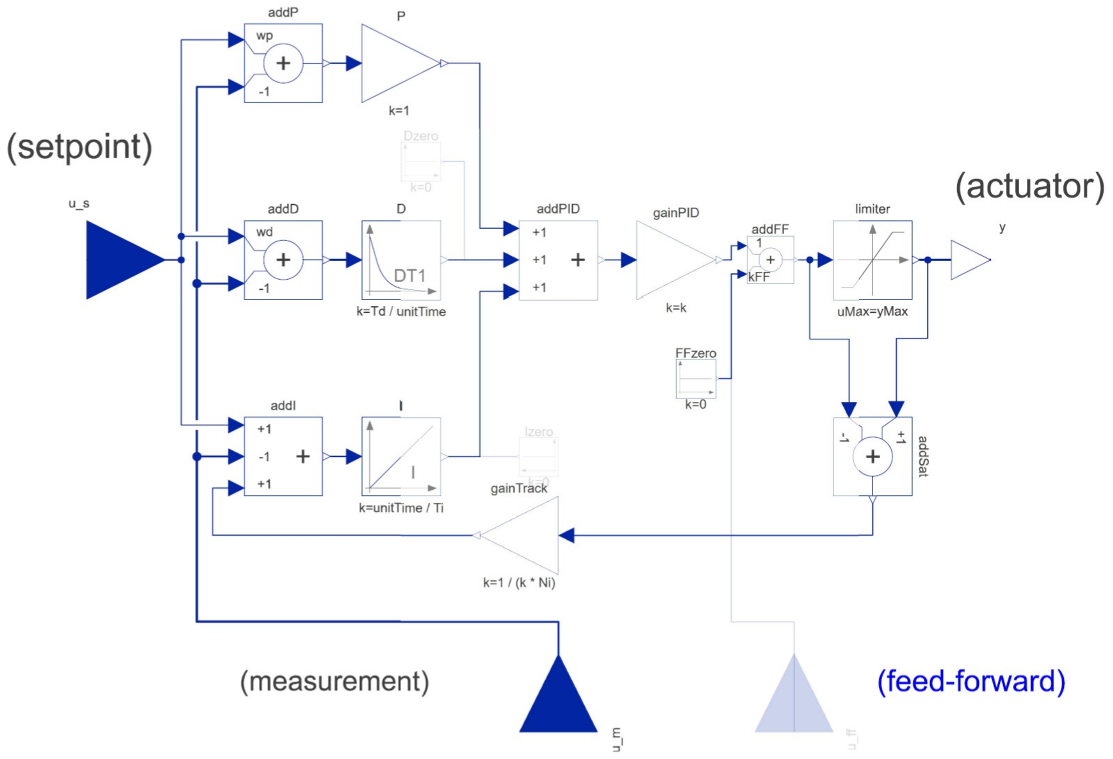
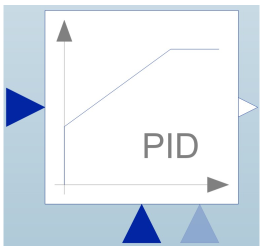


Figure 3.44: `Control` model diagram view

The control part is crucial for the right operating of the whole system in order to obtain the desired useful effect, or rather the thermal comfort achievement. To do this, a control component already present in the *Modelica* library has been used, called `LimPID`; basically it is a controller but with a limited output. That means



(a) Diagram view



(b) Icon view

Figure 3.45: `LimPID` component

there is the possibility to select the minimum and the maximum value for the output accordingly with our issue specifications. Furthermore, it is possible to set it in three different kind: P-controller, PI-controller or PID-controller. For instance by selecting the PI mode, that is the one used in this activity, the derivative part components are removed and not considered. The **k** gain of the proportional part and the **Ti** time constant of the integrator can be manually tuned in order to obtain the best response by the system. Its graphical representation in presented in the figure 3.45: as can be seen both by the icon and diagram view, the component

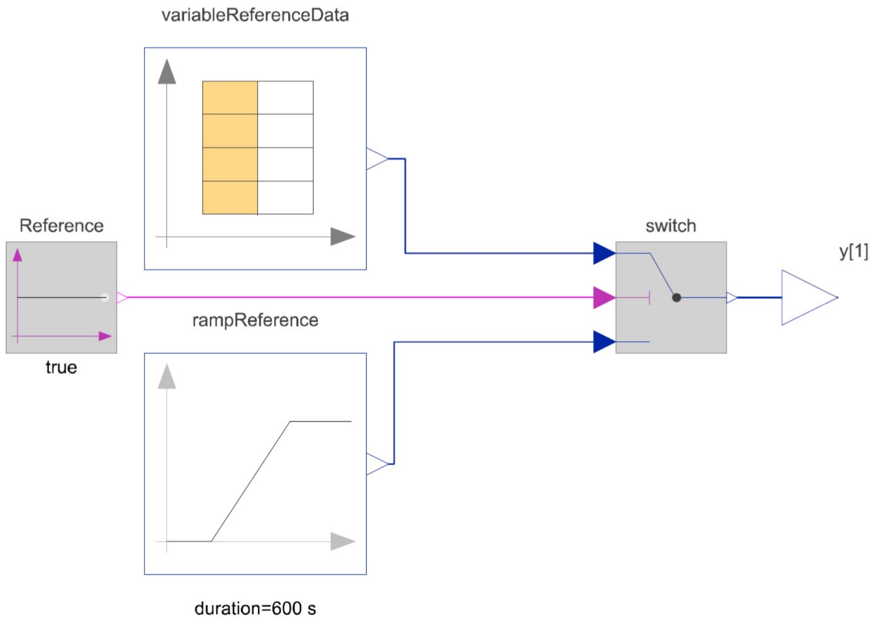


Figure 3.46: `Reference` model diagram view

receives the measurement of the parameter must be controlled comparing it with the set-point of the same quantity producing in output a coherent signal that allows the system to continuously follow the reference value. In the case of the considered system, the control component receives as measurement the house average temperature signal and as set-point the temperature reference that must be ensured inside the house. The signal produced by the `LimPID` modulates the mass flow rate of refrigerant in the heat pump, turning it to its maximum value when the house average temperature is lower then the reference, or to its minimum vice-versa. Thus, by setting as maximum value for the `LimPID` component the needed refrigerant mass flow rate and as minimum a null value, the control component sends the checking signal depending if the heat pump has to work or not based on the difference between the measurement and the set point. The control component is obtained by dragging and dropping the `LimPID` model into a new class by creating the `Control` component visible in its diagram view in the figure 3.44. The set point is generated by the `Reference` thanks to which it is possible to change the reference kind; it basically consists of a *switch* that allows to select either a *ramp* reference, or a *variable* reference generated with a `CombiTimeTable` model to study the situation in which the reference is not constant, but changes during the time passing. This table is easily developed thanks to ***Mathematica***, of which we will talk soon. As can be seen in 3.46 by turning the boolean constant `Reference` value true or false, the choice between respectively *variable* or *ramp* is possible. Lastly, the `controlledSignal` is sent to the heat pump model, and since the signal is time dependent the `FixedMassFlow` of the initial `RefrigerationCycle` model 3.36 is substituted with a new that is variable, as can bee seen in the figure 3.47. Doing so, the system's response follows accurately the reference. To better explain how the control strategy practically operates, an example chart has been plotted consisting of two different plots visible in the figure 3.48. The one on the top represents the reference temperature value, set in the ramp mode, the average house temperature and that of the external environment. The one on the bottom represents the controlled signal that fluctuates between the minimum and maximum mass flow rate value accordingly to the necessity to make the heat pump working. It is appreciable that when the reference is active the mass flow rate is in its maximum value if the house average temperature is lower than it in order to increase the temperature in the house, whereas in its null value when the average is higher than the reference. In this way, as can be observed, the reference is accurately followed.

3.7.5 External temperature data

The last component visible in the top model diagram view 3.40 is the one named `ExternalTemperature` allowing to send the external environment temperature data to each component that needs them. Its way to operate is similar to the `Reference` model, as can be seen by the figure 3.49, since using a switch and a boolean constant value that can be true or false, it is possible changing from data

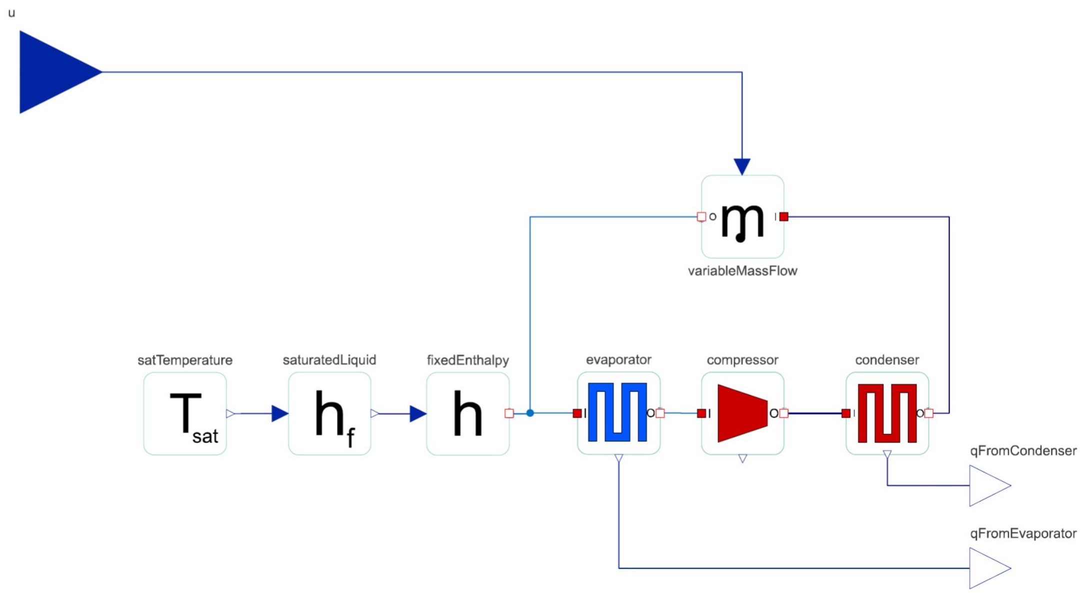


Figure 3.47: Refrigeration cycle model with variable refrigerant mass flow rate

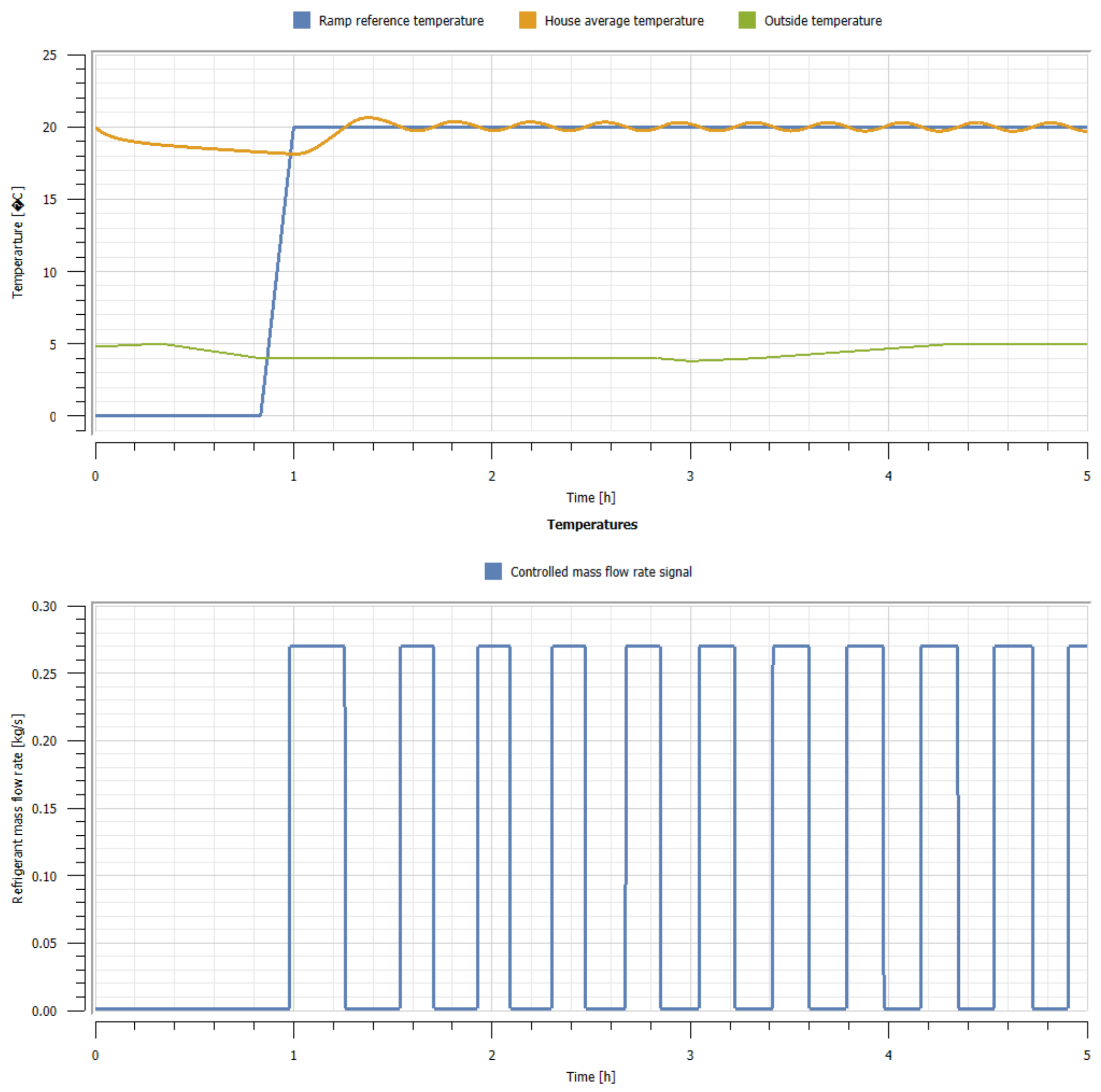


Figure 3.48: Control logic plots

in Milan to that in Rome. Such data has been obtained with Mathematica and may be daily or seasonal and will be presented in the results chapter.

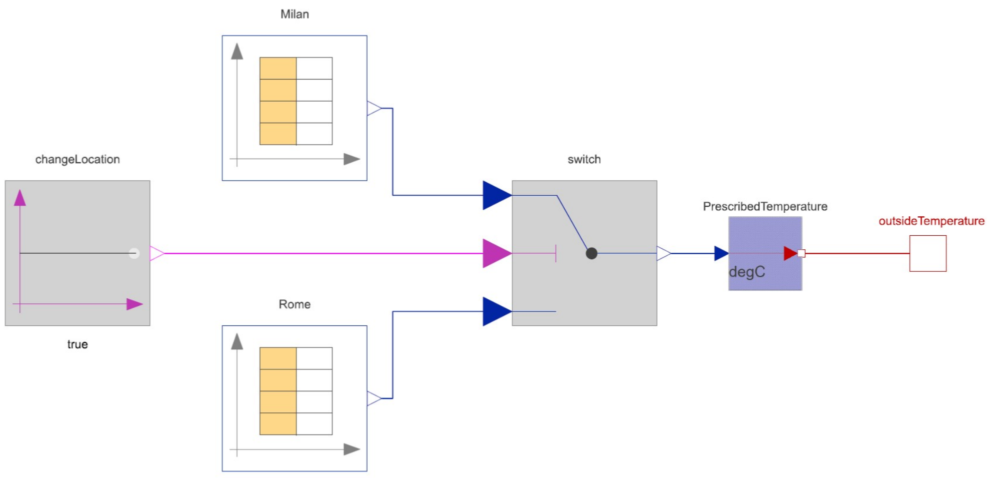


Figure 3.49: `ExternalTemperature` model diagram view

Chapter 4

Model simulation and results

Once the complete system has been modeled on SystemModeler, several simulations and analysis may be carried out in order to demonstrate its operating when different scenarios are considered. That is where ***Mathematica*** comes into play. Mathematica is a computing symbolic and numeric environment that uses a powerful interpreted programming language called *Wolfram Language* that is unique thanks to its *knowledge based* feature: it allows you to do basically everything. As Stephan Wolfram says in his book [20]: «I view the Wolfram Language as an optimized tool for turning ideas into reality. You can make things that are visual, textual, interactive or whatever. But the very knowledge and automation that makes the Wolfram language so powerful also makes it accessible to anyone; you don't have to know about the workings of computers, or about technical or mathematical ideas; that's the job of the Wolfram Language». Regarding systems modeling and simulating, Mathematica may be used to visualize and analyze the results of the simulation of a model developed on SystemModeler, thanks to the countless ready-to-use functions together with all the strong tools Mathematica offers. Thus, combining Mathematica and SystemModeler by using them together, it is possible visualize the AC heating system operating by plotting several charts showing its working and moreover carry out several analysis to appreciate how the main parameters affect the system's behaviour. Furthermore, its versatility and ease of use make possible a energy analysis useful to estimate the energy consumption of the system when it is submitted to different scenarios, allowing to make thermal-economic consideration about how this consumption may be reduced, as we will see soon in the following paragraphs.

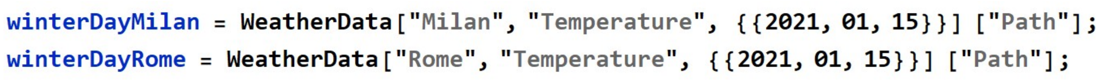
4.1 Simulation of coupled heat pump and building

Purpose of this section is showing how the developed system reacts when it is turned in operating in a real scenario to test its right operating, by carring out a consistency analisys. As said in the complete model's discussion, it has been developed and sized considering Milan as place of destination but the possibility tho test its operating in Rome as well has been implemented, especially to verify its adapting to a situation characterized by different external climate condition, allowing comparative analysis. Thus simply by selecting Milan or Rome in the `ExternalTemperature` model 3.49, the two scenarios may be submitted to our model in order to observe its response.

For doing this, the model has been simulated considering a typical winter day both for Milan and Rome, testing therefore the daily operating of the system. The daily climate data needed by the system have been obtained thanks to Mathematica with extremely ease; once data are available, the desired simulations may be carried out. Let's first get the data using Mathematica.

4.1.1 Climate data

Among the countless functions Mathematica offers, **`WeatherData`** is that allowing to get real data about the climate condition of a specific place. The aim is to obtain the two tables shown in the `ExternalTemperature` model 3.49 that consist basically of two columns; one contains the time from zero to 24 hours, with a temporal step of around 30 minutes, to which correspond the relative temperature data in the other column. In this way a relationship between the time and the temperature values is obtained for a typical winter day of the considered place. The figure below shown the two code lines needed to obtain the data:



```
winterDayMilan = WeatherData["Milan", "Temperature", {{2021, 01, 15}}] ["Path"];
winterDayRome = WeatherData["Rome", "Temperature", {{2021, 01, 15}}] ["Path"];
```

Figure 4.1: `WeatherData` function on Mathematica

As it can be seen, by entering sequentially the city, the type of information needed, temperature in our case, and the chosen day, **`WeatherData`** provides as output the list of these data; subsequently, by manipulating a bit the data and by simply using a function called **`CreateDataSystemModel`**, these lists are converted into the two `CombiTimeTable` models that may be directly used on SystemModeler. Thus, the two timetable both for Rome and Milan are ready containing the real temperature values of the chosen day. The climate data, both daily and seasonal, will be the same for all the simulations.

4.1.2 Milan

Once the daily data for Milan are obtained, the top model may be simulated to demonstrate its right operating.

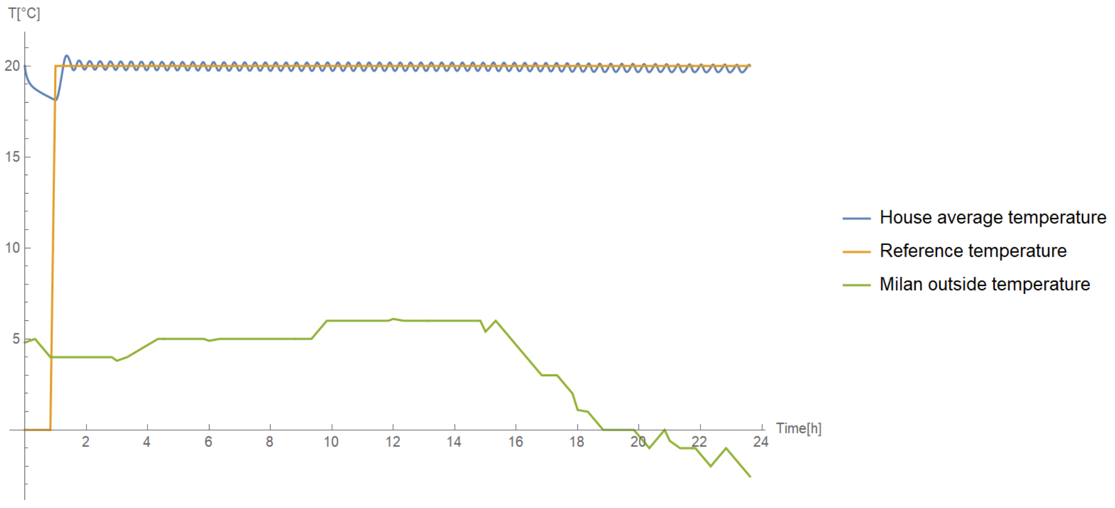


Figure 4.2: Daily simulation for a typical winter day in Milan

The picture 4.2, obtained using `SystemModelPlot` function on Mathematica, represents the system's daily simulation for Milan; as can be seen, until the reference is equal to 0, the system does not work and the average house temperature decreases due to the temperature difference between the internal environment and external one which temperature is represented by the green curve. After one hour, the ramp reference is turned on and the system starts working; indeed, the house average temperature increases trying to continuously follow the imposed reference, showing major difficulty when the external temperature arrives during the day to its lowest values, as predictable. Let's now simulate and plot together Rome's and Milan's scenarios visually comparing the system's different responses.

4.1.3 Milan vs. Rome

As said before, changing location is possible simply setting it in the `ExternalTemperature` model; in this way Rome's temperature data may be loaded as well. To compare the two different scenarios, a simulation with a lower time range than a whole day has been done, to better see the different system's responses about the two considered scenarios in order to demonstrate its correct operating.

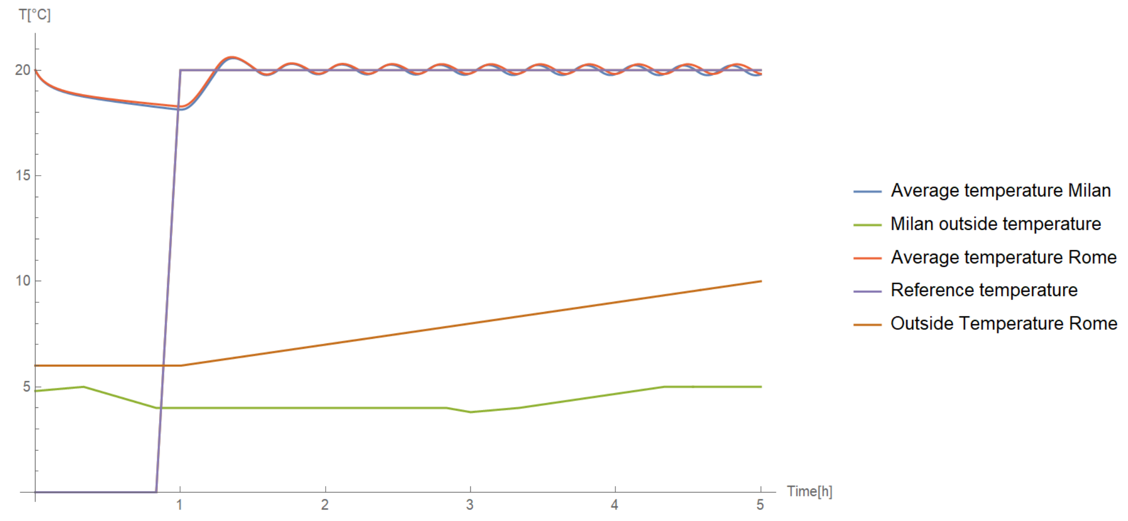


Figure 4.3: Milan vs. Rome scenario

By observing the picture 4.3 indeed, only the first five simulation hours have been considered; it can be appreciate that the two response curves present a small but clearly visible difference; the Milan one decreases faster compared with that of Rome when the signal reference is absent. This is due to, obviously, the different climate conditions for the same day in the two cities clearly shown by their related curves. That's means the system has to work less when in Rome, requiring a lower energy expenditure as it will be demonstrate in the relative energy analysis section.

4.2 Parameter comparison

In this section parametric comparison have been done in order to test the response of the system when some main parameters are varied. As first analysis, the influence of the refrigerant's mass flow rate has been considered in order to demonstrate the importance of a correct heat pump system design, especially about the refrigeration cycle that represents the core of its operating and also to avoid an unnecessary and excessive waste of energy. For doing this, only the daily operating of the system has been considered and just for Milan, the design place used for sizing all the system. Secondly, the system's response has been tested to varying the principal parameters of the PI controller in order to observe how these can perturb the system's behavior.

```
coolantSim = SystemModelSimulate["Luigi_Model.Test.ACHeatingSystem",
   {0, 1 * 24 * 60 * 60}, <|"ParameterValues" → {"coolantFlowRateControl" → 0.27 * {0.7, 1, 1.3}}|>];
```

Figure 4.4: Simulating code for the refrigerant parameter comparison

4.2.1 Refrigerant mass flow rate variation influence

The `SystemModelSimulate` function has been here used similarly how already did before with a small but significant difference; by observing the figure 4.4 it has been possible to store in only one variable, `coolantSim`, a multiple simulation where the parameter `CoolantFlowRateControl` assumes three different values obtained by multiplying the design one for 0.7, 1 and 1.3 to see how changes the response of the system when the refrigerant mass flow rate is varied of 30 percent more or less than the design value.

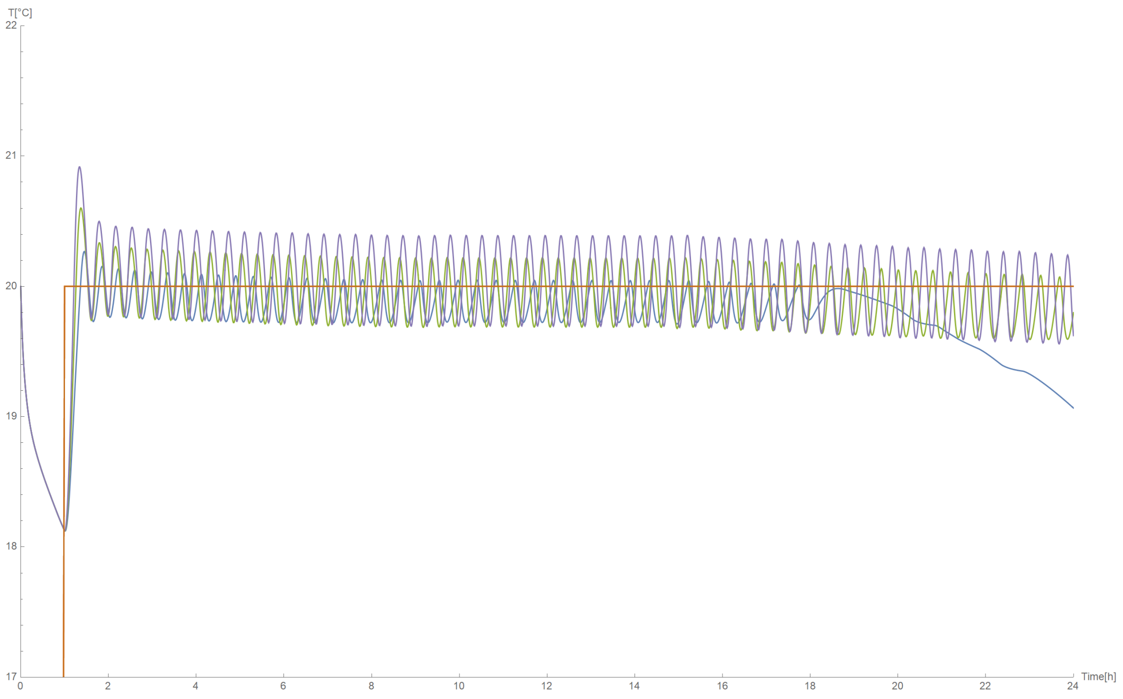
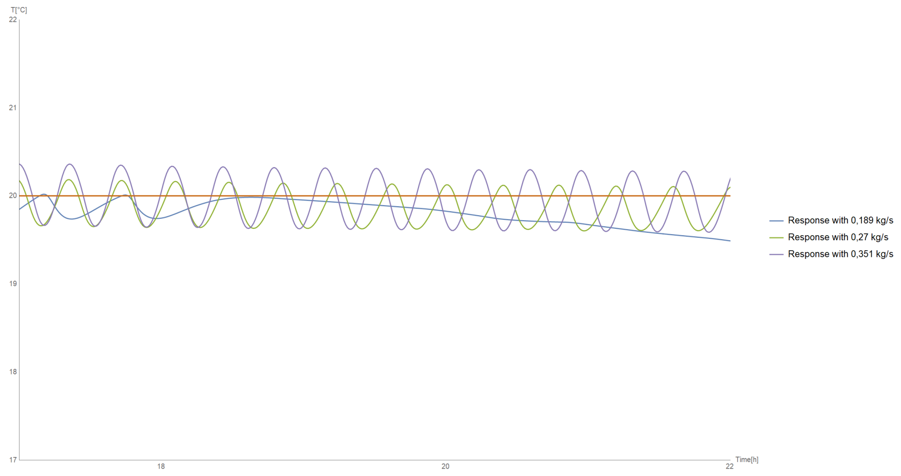
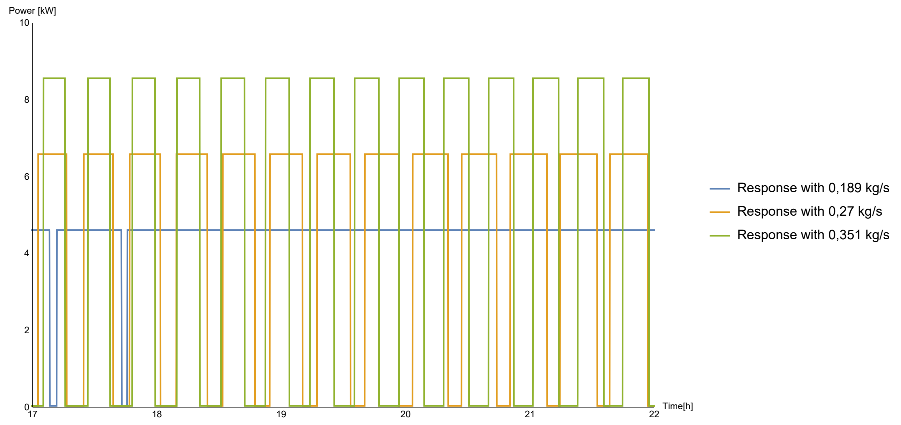


Figure 4.5: System's response to different refrigerant mass flow rate values

The picture above shows simulation results for the whole day; the green curve represents the design condition, while the blue and the purple respectively those with a minor and major value of refrigerant mass flow rate. As it can be seen, the reference is accurately followed by the green one, whereas that blue can't provides the required demand properly and that purple is characterized by an excessive energy expenditure. To better graphically shows the difference a five hours range



(a) Temperature response



(b) Absorbed power

Figure 4.6: Influence of refrigerant mass flow rate

time between 17 and 22 has been considered, where the blue one starts to decay. The simulation's plot of this scenario is presented in the figure 4.6. It is now clearly visible how the in the blue condition the house average temperature starts to decay not following the reference being its refrigerant mass flow rate value lower than the minimum and suitable one needed for the system to work properly.

4.2.2 PI controller

Another interesting parameter comparison analysis might be the one related to the PI controller; indeed, by varying its main parameters such as the **k** gain of the proportional part or the **Ti** time constant of the integrator part, different system's behavior can be obtained. Thus, an evaluation of how this variation may perturb

```
controllerkSim = SystemModelSimulate["Luigi_Model.Test.ACHeatingSystem",
   {0, 5 * 60 * 60}, <|"ParameterValues" → {"proportionalGain" → {0.1, 1, 10, 100, 1000}}|>];
```

Figure 4.7: Simulating code

the considered system has been carried out by simulating first the top model, presuming that the Ti time constant is constant, while the k gain changes assuming five different values, one for each order of magnitude from 0.1 to 1000, in order to establish which provides among them the better system response. Secondly, the situation turns upside down since k is kept constant whereas Ti assumes four different values from 0.01 to 10 s. The error between the feedback and the reference is plotted as well for each situation in order to appreciate its trend. For all the simulations in this section a 5 hours range time has been considered and Milan as location.

Proportional part

Let's consider the **k** gain variation of the controller's proportional part. Always using the `SystemModelSimulate` function on Mathematica, it is possible to recall the top model and simulate it directly with five different values of k, controlled by the `proportionalGain` parameter, in just one time and with one code line, presented in the figure 4.7

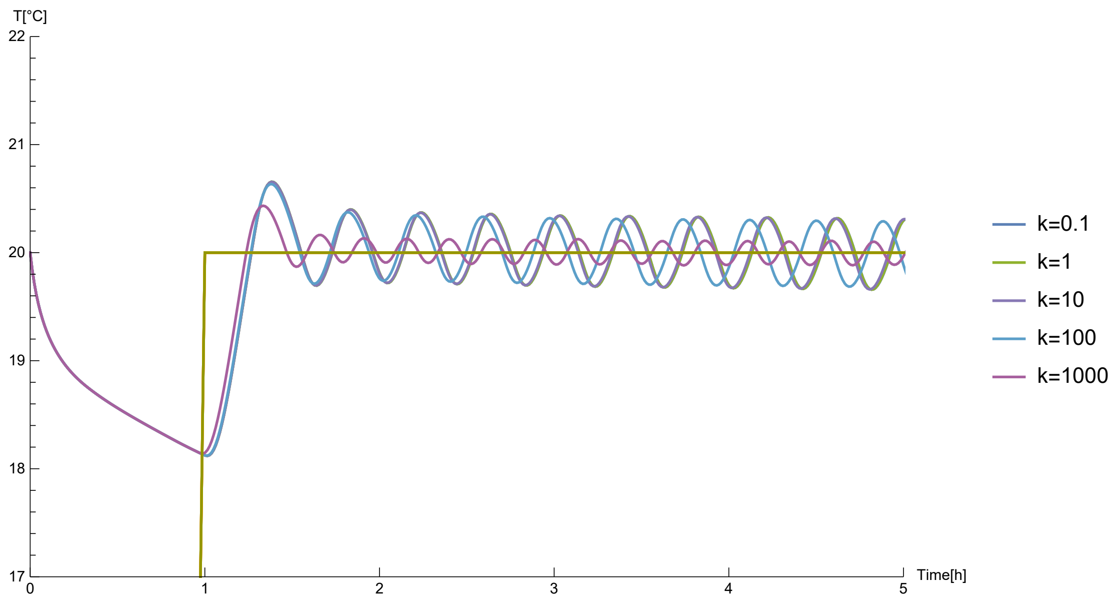


Figure 4.8: Different response of the system when **k** is varying

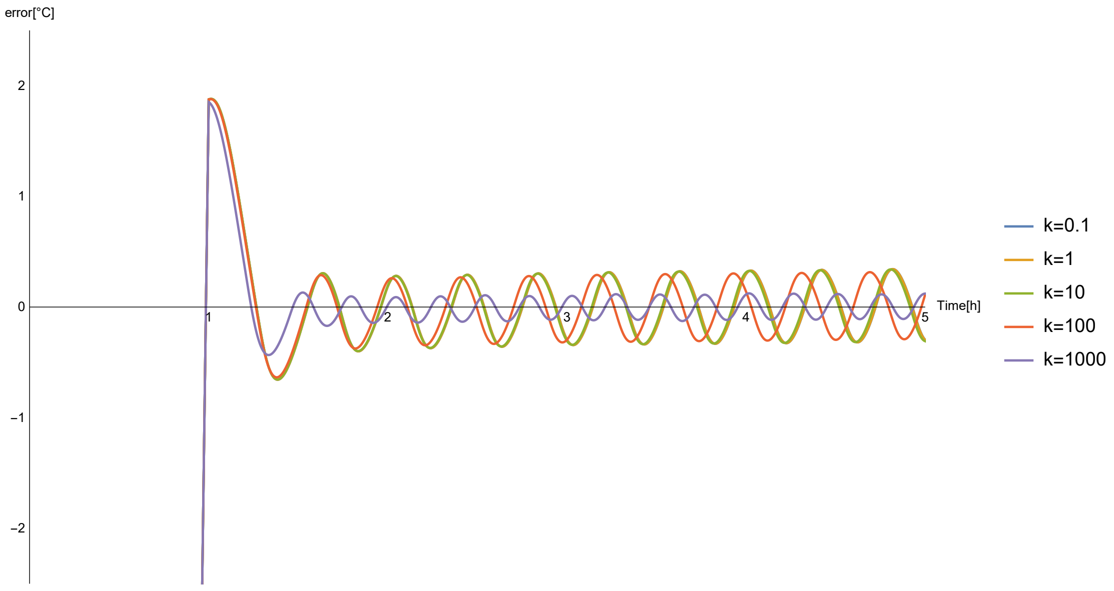


Figure 4.9: Control error when **k** is varying

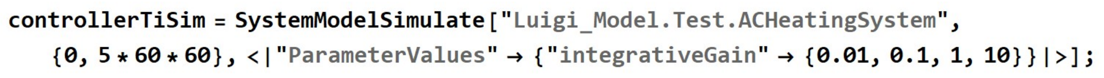
The simulation's results are shown in the figure 4.8. As can be noted, those with a value of k equal to 0.1, 1 and 10 do not display much differences, the curves basically overlap each other; wheres those with k equal to 100 or 1000 begin to show substantial difference relating to the system's response: the condition with k=1000 shows the optimal trend speaking only in terms of following the reference. However, a so high value for k produces a really good response but the system becomes more *nervous* with the possibility to lead it unstable; for this reason a compromise is needed among the instability and a good response by choosing for k a value equal to 500.

By the figure 4.9, it can be observed how, indeed, the error related to the situation with k=1000 is the smallest. Furthermore it can be seen from here that the system does not work until the ramp reference is turned on, or rather after 50 minutes from the initial time instant and with a duration of 10 minutes.

Integrative part

Let's now variate the time constant **Ti** of the integrator block by keeping that of proportional at a constant value equal to 500 previously chosen as the optimal one. Again, the top model has been recalled by the `SystemModelSimulate` function where this time the fluctuating variable is **Ti** controlled by `integrativeGain` parameter, as can be seen in the code presented in the figure 4.10

By observing the chart in the figure 4.11, representing the simulation's results, it is evidently that with a time constant higher than 1 second the reference is not



```
controllerTiSim = SystemModelSimulate["Luigi_Model.Test.ACHeatingSystem",
   {0, 5 * 60 * 60}, <|"ParameterValues" → {"integrativeGain" → {0.01, 0.1, 1, 10}}|>];
```

Figure 4.10: Simulating code

properly followed, while when it assumes as value 0.01 or 0.1 seconds the system can follow the reference correctly. Among them, that characterized by a time constant equal to 0.1 seconds presents a lower overshoot and a more regular trend compared with the one with a $Ti = 0.01$ seconds; hence, a value with the same order of magnitude has been chosen for the system. The error's plot regarding this situation is presented in the figure 4.11b where it is easy verifying how the orange curve, $Ti = 0.1$ seconds, follows the reference better than the others.

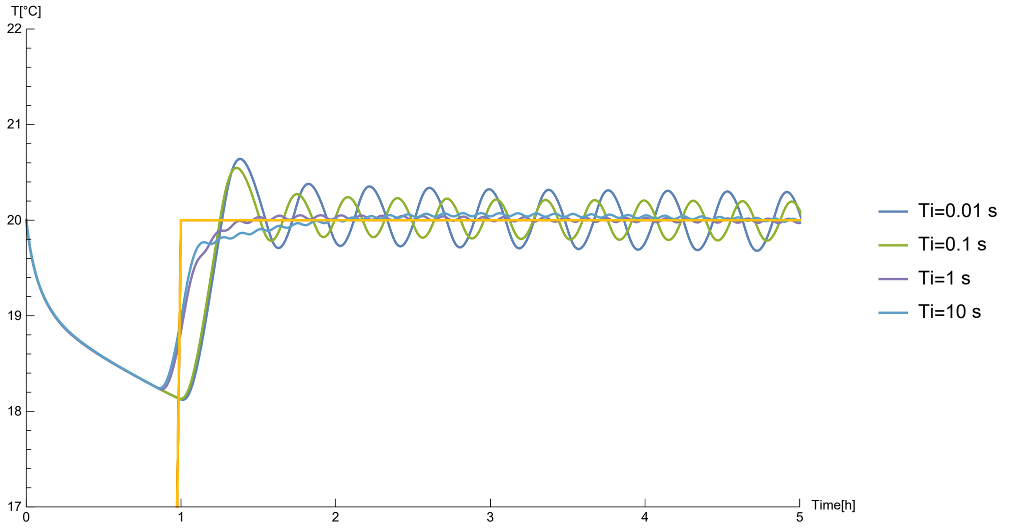
4.3 Energy analysis

The second part of the system's analysis activity is focused on the energy consumption estimation by carrying out an energy analysis in order to demonstrate that the developed model may be used as a tool to quantify the system's expenditure when it is submitted to different scenarios and to make thermal-economics considerations in order to compute the energy saving that can be achieved by using for instance a thermal coat. Thus, the simulations made in this section concern the whole winter season, guessed from 2020/11/15 to 2021/02/15; the temperature data related to this range are obtained again on Mathematica identically how done for the daily situation by using `WheaterData` function, both for Milan and Rome, that provide the real temperature values for the considered cities and in the specified period.

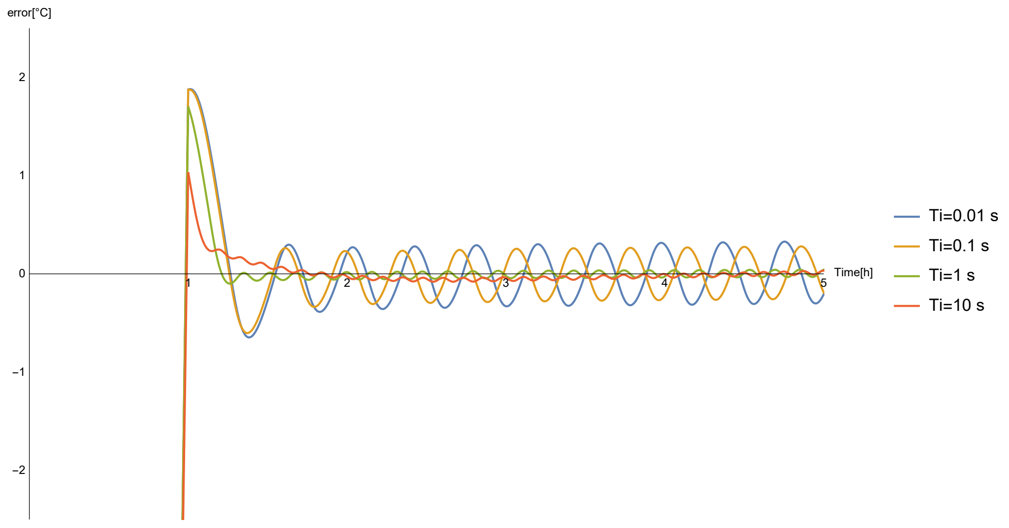
Hence firstly Milan's scenario is considered, creating a simulation of the system's operating for the entire time range considered in order to compute its energy expenditure and it is compared with that one obtained for Rome. Secondly, two different situation are implemented just for Milan in order to demonstrate how, by using a thermal coat or a variable reference, this consumption may be reduced.

4.3.1 Seasonal consumption Milan vs. Rome

Let's hence simulate the model for the whole winter season in Milan and subsequently plot the chart containing the simulation results. The plotted parameters, visible in figure 4.12, are: the reference temperature signal, always constant along the season and equal to 20° C; the house average temperature representing the system's response; the effective Milan outside temperature trend along the considered months. Exactly the same has been done for Rome, of which simulations results are presented in figure 4.13



(a) Temperature response



(b) Error plot

Figure 4.11: Milan scenario

In order to quantify the energy consumption spent by the system when it is installed in the two different cities and to carry out comparison among their energy expenditure, the compressor's operating must be considered. In fact, to compute the energy consumption, it is possible to store in a variable the corresponding absorbed power by the compressor; indeed, since we want estimate the kWh consumed by the system, we should multiply the considered time for the absorbed power and this means compute the time integral of it, in order to turn the kW in kWh.

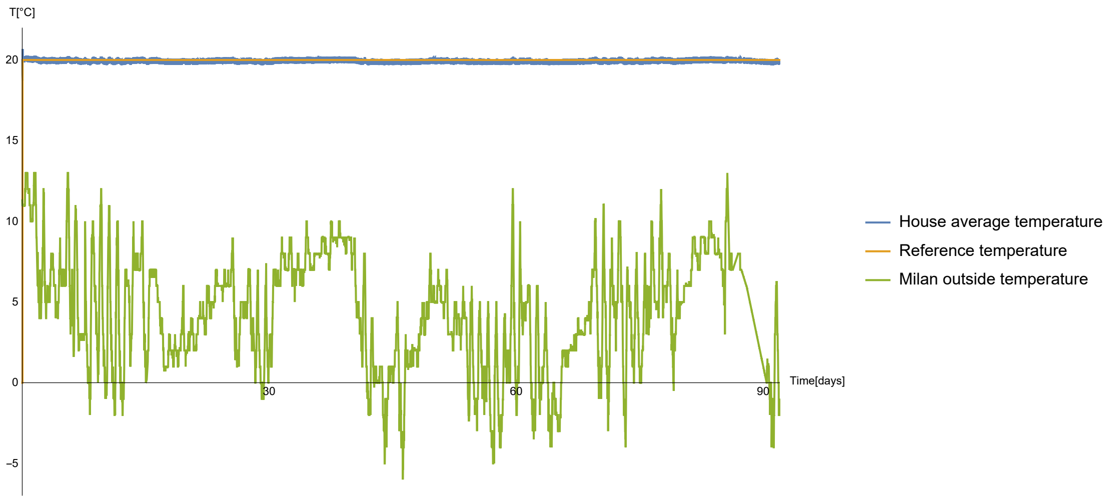


Figure 4.12: Season simulation results for Milan's scenario

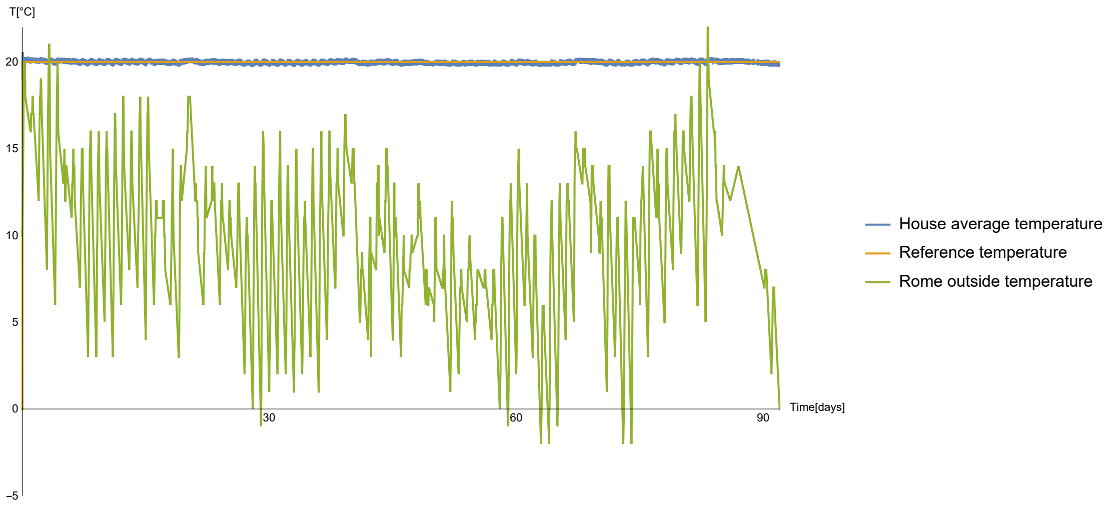
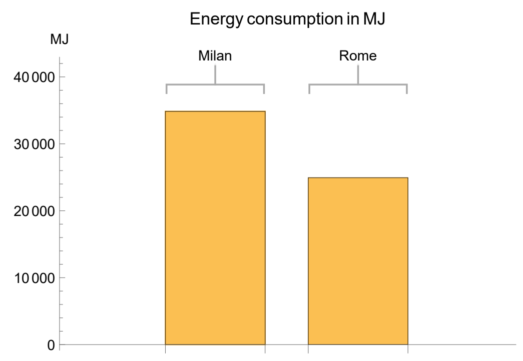


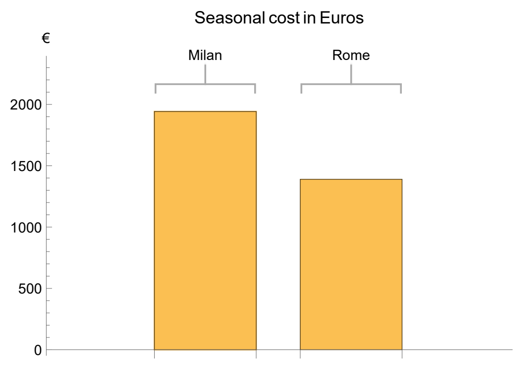
Figure 4.13: Season simulation results for Rome's scenario

Once the kWh are obtained, they may be turned in MJ to realize the energy comparison. Furthermore, by knowing the actual electricity cost per kWh, a simple seasonal price needed estimation may be occurred. Let's explain the energy expenditure estimation process:

First, the information about the absorbed power [kW] by the compressor is extracted by the simulation and then stored in two variable both for Rome and Milan under the name `CompressorPowerMilan` and `CompressorPowerRome` respectively. This allows later to use the function `NIntegrate` to compute the time integral by considering the whole season period, thus providing the kWh consumed by the



(a) Energy expenditure



(b) Operating cost

Figure 4.14: Milan vs Rome seasonal comparison

system in the scenarios reported by the following expressions:

$$\texttt{EnergyExpenditureMilan} = 9680.58\ kWh = 34850.1\ MJ \tag{4.1}$$

$$\texttt{EnergyExpenditureRome} = 6926.98\ kWh = 24937.1\ MJ \tag{4.2}$$

These values got through the integrals resolution represent the energy expenditure expressed in kWh; it is a practical parameter allowing the system's operating total price evaluation for the entire winter season. The values expressed in MJ are obtained simply by multiplying those in kWh for 3,6.

In order to express these energy expenditure in terms of spent euros, the actual electricity price per kWh has been assumed by consulting data provided by ARERA website [21], where the one related to the first quarter of 2021 has been chosen equal to **0,2006 €/kWh**. Then, supposing compressor's electrical efficiency equal to $\eta_{el} = 0.99$ that means the actually absorbed power is basically the same to the one computed thanks to the integration, the working's cost of the heating system for the winter season both for Milan and Rome is easily obtained by multiplying the electricity cost in Italy for the energy expenditure expressed in kWh:

$$Cost\ in\ Milan = 1941.92\ EUR \tag{4.3}$$

$$Cost\ in\ Rome = 1389.5\ EUR \tag{4.4}$$

As predictable, the heat pump system requires a lower amount of electric energy when works in Rome compared with Milan. This is self-evident since the outside temperature in Rome is higher than in Milan for the considered period that means a minor heat transfer from the inner of the house to the external environment under the same inner temperature; thus the apartment in Rome disperses minor heat allowing a heat pump operating more bland, which translates in a lower energy expenditure. It is also important observe that, since the system has been designed considering Milan as final destination, it results oversized when is located Rome.

4.3.2 Thermal building envelope

After having shown the difference in operating and how the energy expenditure changes by submitting the heating system to two different city with significant variation of external environment temperature, by considering winter season, such as Milan and Rome, let's consider another one scenario where a thermal coat is applied to the apartment in order to reduce its transmittance and consequently increase its capacity to retain heat. Also for this situation a comparison between the situation without the coat and that with has been carried out by comparing the total MJ consumed by the heat pump to ensure the right comfort inside the house. Furthermore, an economic evaluation has been done to quantify the number of years needed to recover the initial coat investment, since it brings a benefit in terms of energy saving every year compared with the situation in which the coat is absent. The analysis here considers always the seasonal situation in Milan.

Energy expenditure comparison

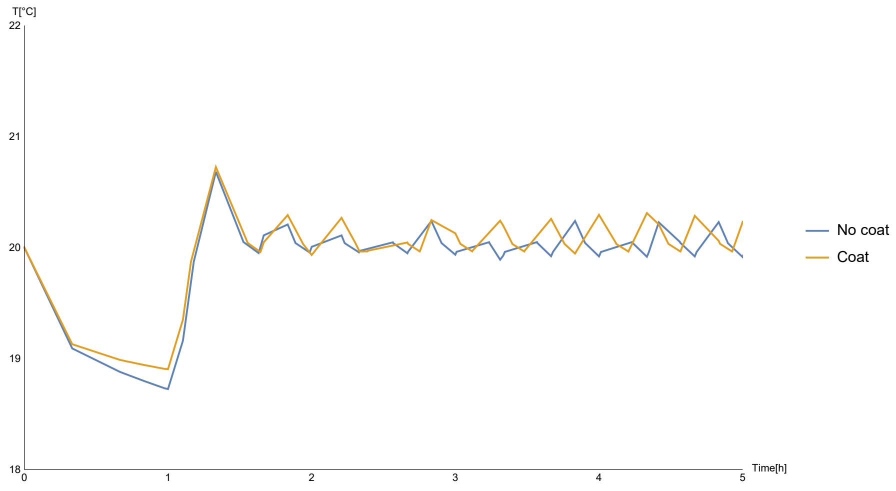
Such thermal coat has been guessed to be constituted of an additional insulation layer on the external surface of the building. **Rock wool** has been chosen as insulation material with a thermal conductivity $\lambda = 0.037\ \frac{W}{mK}$ and a thickness of 5 cm. Thus let's simulate the new scenario in the whole winter season and compute the energy expenditure; it then will be compared with the condition without coat.

In order to graphically demonstrate the difference about the system's response between the two scenarios, a chart showing the simulating results both for the condition with coat and without has been plotted and reported in figure 4.15, by considering a small 5 hours range time useful to better show the curves difference.

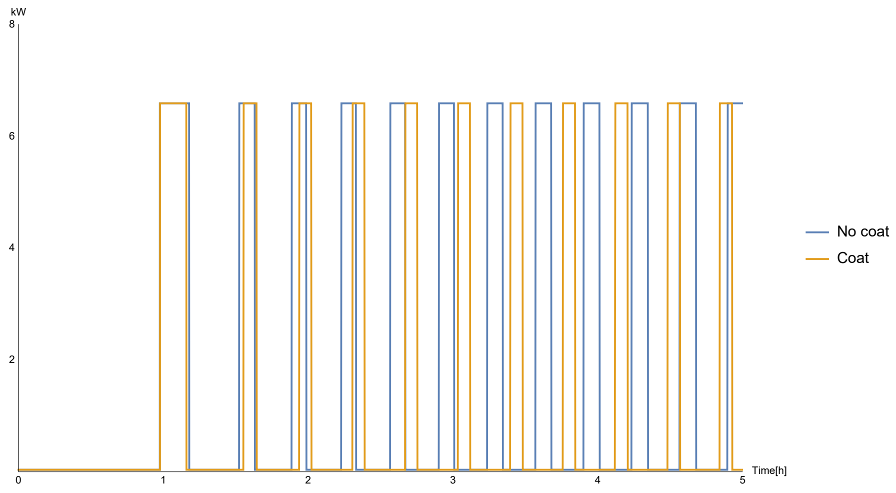
The energy expenditure for the two considered scenarios has been estimated exactly how done in the Milan vs Rome section. Thus, the kWh are obtained by integrating the absorbed power by the compressor and then the values in MJ may be get:

$$\texttt{EnergyExpenditureMilanNoCoat} = 9680.58\ kWh = 34850.1\ MJ \tag{4.5}$$

$$\texttt{EnergyExpenditureMilanCoat} = 5346.44\ kWh = 19247.2\ MJ \tag{4.6}$$



(a) House average temperatures



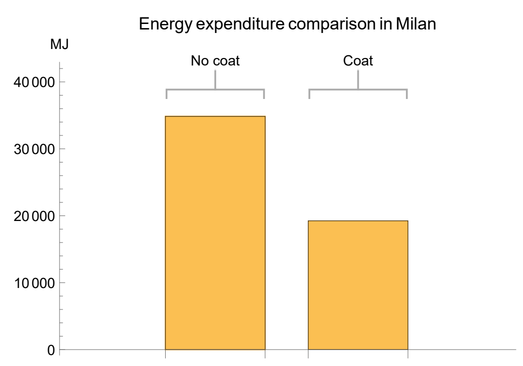
(b) Absorbed power by compressor

Figure 4.15: Simulating results thermal coat scenario

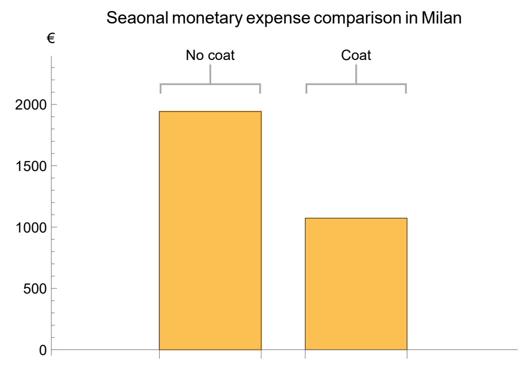
Regarding the conversion of the energy expenditure in money by knowing the kWh consumed and by multiplying it for the already introduced electricity cost, quantifying monetary expense and comparing the two situations have been possible:

$$Monetary\ expense\ without\ coat = 1941.92\ EUR \tag{4.7}$$

$$Monetary\ expense\ with\ coat = 1072.5\ EUR \tag{4.8}$$



(a) Energy expenditure



(b) Operating cost

Figure 4.16: Seasonal comparison thermal coat scenario

Return of investment

Let's now carry out an economic evaluation by computing the number of year needed to recover the initial investment for buying the thermal coat. In order to do this, the Rock wool's cost has been considered, guessed to be equal to 60 Euros per square meters, where the material's cost, transport costs and labour have been considered. The total suface that needs to be covered is equal to 252 m^2, hence computing the initial investment is easy amounted to 15120 EUR.

In order to quantify number years required for the return of investment, this initial cost has been divided for the annual saving obtained by using the coat, computed as difference between the monetary expense in Milan without coat and the one with:

$$\texttt{Seasonal money saving} = 869.428\ EUR \tag{4.9}$$

Finally, the years needed to recover the investment are computed:

$$\textsc{Years} = \frac{\textsc{Initial investment}}{\textsc{Seasonal money saving}} = 17.4 \tag{4.10}$$

Hence, we can conclude the discussion saying that by covering the house with a rock wool layer 5cm-thick, the energy expenditure, considering Milan as system's operating city, may be reduced of a 45 % in the whole winter season. Furthermore, by carrying out a simple computation, the needed years for the return of investment has been found equal to around 18 years. This means that after 18 years of using, the total initial investment for the thermal coat will be recovered.

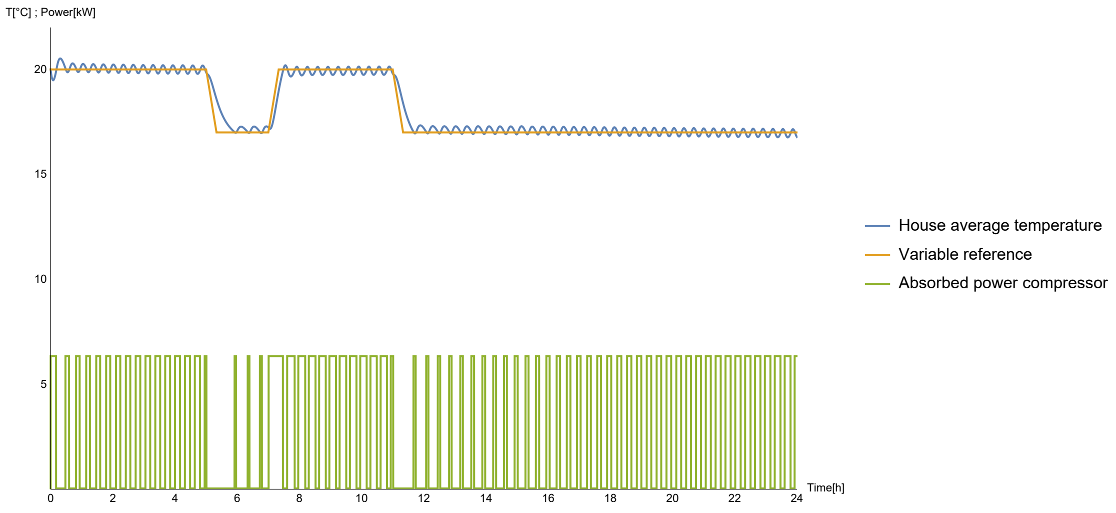


Figure 4.17: Variable reference scenario

4.3.3 Timer: variable reference

Another way to save energy is set a timer in order to make the temperature reference variable. Let's consider daily operating and suppose the building represents a work office in Milan, where the day starts at 8:00 and until 13:00 since people are working, a constant temperature of 20 °C must be ensure. From 13:00 to 15:00 all the workers go to lunch break and none is in the office, so the timer starts working by setting the temperature reference to 17 °C. At 15:00 all the employees back to work and the the reference is changed again to 20 °C, until 19:00 when the working day is ended and the system set the temperature to 17 °C again, in order to save electric energy since the heat pump works fewer. The simulation results both for the system's response and for the absorbed power by the compressor related to this new scenario are presented in figure 4.17: after 5 hours the reference changes its value and the system is automatically turned off to achieve the new reference value; in this juncture the compressor works at its minimum power, as can be appreciate observing the green curve; this value is kept for 2 hours until it is returned again to 20 °C in order to ensure the thermal comfort for all the workers. After 4 hours the working day is ended and the reference varies again to 17 °C. This analysis demonstrates in addition the capacity of the modeled system to adapt its operating to any conditions, always guaranteeing the set reference and following it properly.

Economic comparison

Lastly, in order to quantify the saving brought by the choice to have a variable reference along the day rather than be constant, the daily money expense has been

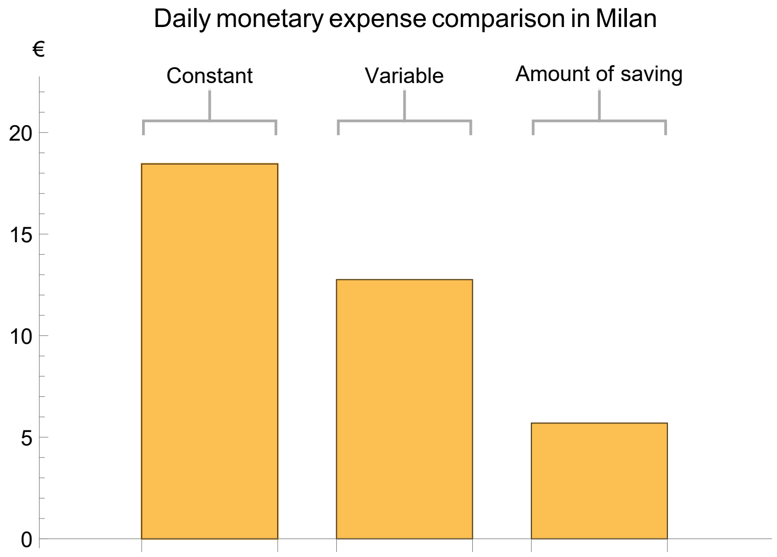


Figure 4.18: Economic comparison

computed and compared with the already studied condition of daily heating in Milan but with a constant reference, visible in 4.1.2. The procedure to pass from the consumed energy in terms of kW to spent Euros is exactly the same already used in the previous analysis. Hence, the daily money expense obtained are 18.45 Euros per day for constant reference scenario, wheres 12.75 per day for the one with the variable reference. The comparison is presented by the bar chart in figure 4.18.

Thus, by having care of using a timer to make the temperature reference variable, a daily saving of 5.69 euros may be achieved.

Chapter 5

Conclusions and future development

5.1 Conclusions

The principal purpose of this activity has been modeling and simulating an air conditioning heating system by using the innovative object-oriented modeling approach through Modelica on Wolfram SystemModeler and to demonstrate how the developed model may be useful in the teaching field. It indeed represents a powerful tool capable to make the physical phenomena behind the heat transfer and the practical aspects for such a plant's sizing in general more understandable .Moreover, it provides the possibility to easily and quickly appreciate how, by changing some key parameters, the thermal behaviour can vary through the creation of interactive charts and multiple simulations achievable thanks to Mathematica, other product developed by Wolfram, based on a powerful programming language, the Wolfram language. It may be used not merely for a didactic use, but as powerful tool to analyze a so complex system in order to consider all the critical issues which its sizing can present and to develop optimization strategies to, for instance, reduce its energy expenditure.

In the introduction a brief literature research has been carried out about how the OOM approach have already been used in the air conditioning field, pointing out how its utilisation allows the user focusing the attention more on the physical formulation of the considered system, rather than developing of a resolutive algorithm which might introduce errors or time wasting.

Subsequently, the first two chapters deal with all the main OOM approach aspects regarding on the logic behind its operating and on how it is used on SystemModeler, by presenting its applying ease and the advantages derived by its use, by focusing more on the thermal domain with an example developed by

Michael Tiller [14] which should be capable to give a whole overview of its potential, such as modularity, flexibility, hierarchical modeling etc. Thus, starting from some already present models in the `College Thermal` library developed in cooperation with Politecnico di Torino, an heat pump - fan coils heating system has been studied and modeled under all its aspects, from the computation of the required thermal load by the building to the set-up of the PI controller useful to rule the system's operating; studying the heat transfer mechanisms inherent to the considered system has been carried out in order to model all the major component thermally affecting the behavior of a such system: apartment, room, fan coil, heat pump, controller etc. Thanks to the ease using of SystemModeler, a dedicated library has been developed containing all the models needed to realize the complete plant, models that can be modified and rearranged each other in a countless number of different ways to study different situations and to adapt them to the considered issue; therefore its modularity and flexibility have allowed a smart and easy creation of the complete system. Once it has been got, it makes available a large number of analysis by combining the using of Mathematica and SystemModeler. That is what it has been done in the fourth chapter; firstly the model has been tested to verify its correctness and performance by simulating it considering different scenarios obtaining great results in terms of consistency with expected results. Secondly, it turns out to be an optimal tool to perform several energy analysis to quantify the system's energy expenditure. Indeed, by using some pre-built functions available on Mathematica, entirely dedicated to the interaction with models developed on SystemModeler, an approximate evaluation of the energy expense required by the system for its operating has been carried out. This also allows to take into consideration different scenarios in order to lead a performance comparison among them and make some critical economic judgments with a view to devise energy saving strategies. In this regard, it has allowed to quantify the energy saving when a thermal coat or a variable temperature reference are used, simply by modifying the model a bit and using the powerful Wolfram language on Mathematica which has allowed a huge number of different analysis with few lines of code

In conclusion, from this thesis activity it has been possible to deduce how such approach may be applied for modeling a so complex system such as the heating system of a building, allowing with extreme versatility and flexibility usage to dig deep under every aspect, thermal and practical, decisive for the operating of such a plant. The developed models on SystemModeler are completely reusable and editable in order to analyse a different situation by connecting in several manners all the components thanks to the drag and drop approach; simply by interacting with the apartment model, for instance, with few operations the realization of a completely different layout is possible, perhaps using different construction materials or changing the occupancy level. Or even, thanks above all to the interaction with Mathematica, the climate data of any place where we want locate the system in order to study its operating may be obtained simply by writing two lines of code.

All these features make it a valid tool usable for teaching aspect, since it allows to expose the complex theoretical aspects behind the heat transfer and how they can be applied for sizing a such plant through model simulating and plotting charts allowing to the student to develop a critical thought about how the thermal behavior of the system's components may change by varying its operating modalities, or on how act in order to get a energy expenditure reduction simply by interacting with the already existing models or creating new ones. The ease with which it can be used or edited makes it incredibly powerful since it can be used also by users not particularly skilled both in the modeling and programming field and in applied physics.

5.2 Future development

Regarding possible future development to optimize the model and make it more complete, some activities might be realized and are now mentioned, considering that they are just some of the countless which may be introduced:

- Since the particularity of a heat pump system is to operates both in heating or cooling mode, this possibility may be implemented by editing the heat pump model by including the utilisation of a reverse valve to invert the refrigerant's cycle.
- In this thesis activity the system has been controlled by implementing a global control logic, that consider the apartment's temperature as checking variable computed as the average of the temperature of all the rooms. This means that, if the reference temperature is set on 20 degrees, in some rooms the temperature will be higher and others lower than that value, even if only slightly. This may be improved by implementing a local control for each fan coil to establish how much should be its related mass flow rate ensuring to follow the reference.
- Machine learning techniques, field in which Wolfram leads, may be applied to develop advanced control logic based on data collection coming from thermal images, thermal sensors to predict the system's behavior allowing to realize some algorithms that make the system's operating automated which independently can change its operative modalities to ensure both a right thermal comfort by considering the effective occupants feelings, and a smart use of energy sources, in order to maximize its performance.
- The models of radiant panel may be developed in order to carry out a comparison between the fan coils solution and the radiant panel one, by producing

energy analysis that practically demonstrate which is the best solution and under which aspects.

- Implementation of more techniques capable to allow a energy saving with a look at the environmental protection and at the optimal handling of the available energy source.

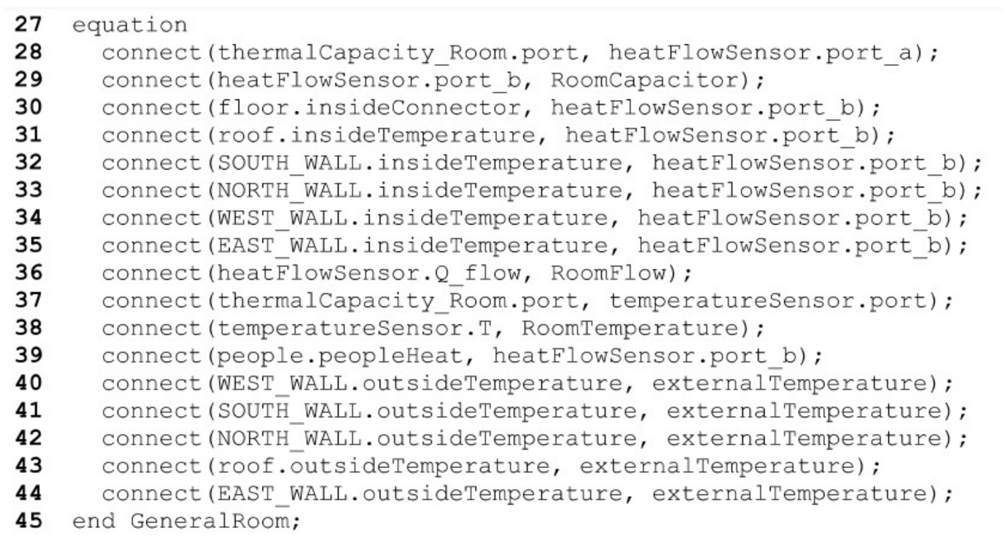
  For example, a thermal accumulator may be included that allows a significant energy saving: indeed, since the electricity cost is different if it is bought during the day or the night, we might think to make the system working along the night hours, when the electricity costs less, to produce hot water that is stored in the accumulator and make it available in the daily hours, when the electricity cost is higher, while the system does not working and thus by saving a lot of energy.

- In this activity a strong approximation regarding the radiation values has been done: the possibility to consider variable data for irradiation may be introduced, by building some `CombiTimeTable` on SystemModeler where a relationship between the time passing and the right values of irradiance is obtained in order to make the model more realistic under this aspect.

Appendix A

Codes

Text view of some developed models:



```
equation
  connect(thermalCapacity_Room.port, heatFlowSensor.port_a);
  connect(heatFlowSensor.port_b, RoomCapacitor);
  connect(floor.insideConnector, heatFlowSensor.port_b);
  connect(roof.insideTemperature, heatFlowSensor.port_b);
  connect(SOUTH_WALL.insideTemperature, heatFlowSensor.port_b);
  connect(NORTH_WALL.insideTemperature, heatFlowSensor.port_b);
  connect(WEST_WALL.insideTemperature, heatFlowSensor.port_b);
  connect(EAST_WALL.insideTemperature, heatFlowSensor.port_b);
  connect(heatFlowSensor.Q_flow, RoomFlow);
  connect(thermalCapacity_Room.port, temperatureSensor.port);
  connect(temperatureSensor.T, RoomTemperature);
  connect(people.peopleHeat, heatFlowSensor.port_b);
  connect(WEST_WALL.outsideTemperature, externalTemperature);
  connect(SOUTH_WALL.outsideTemperature, externalTemperature);
  connect(NORTH_WALL.outsideTemperature, externalTemperature);
  connect(roof.outsideTemperature, externalTemperature);
  connect(EAST_WALL.outsideTemperature, externalTemperature);
end GeneralRoom;
```

Figure A.1: `Wall model pins connection`

Luigi_Model.FanCoil.Components.insideConvection **1**

```
model insideConvection
  Modelica.Thermal.HeatTransfer.Components.Convection convection;
  Modelica.Blocks.Math.Gain computedGc(k = 1);
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_b fluid;
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a solid;
  parameter Modelica.Thermal.FluidHeatFlow.Media.Medium medium = Modelica.Thermal.FluidHeatFlow.Media.Water() "Fluido usado en el conducto";
  Real Pr "Prandtl Number";
  Real Re "Reynolds Number";
  Real Nu "Nusselt Number";
  parameter Modelica.SIunits.Distance innerDiameter = 0.02 "Duct inner diameter";
  Modelica.SIunits.CoefficientOfHeatTransfer h_inner "Inner heat transfer coefficient";
  Real PipeSurface(unit = "m2") "Pipe's surface";
  parameter Modelica.SIunits.Distance lengthPipe = 30 "Longitud del conducto";
  Modelica.Blocks.Interfaces.RealInput volumetricFlow(unit = "m3/s") "Volumetric Flow inside the pipe";
  Modelica.SIunits.Velocity v "Velocity inside the pipe";
equation
  ///
  v = volumetricFlow / ((innerDiameter / 2) ^ 2 * Modelica.Constants.pi);
  Pr = medium.nue / (medium.lamda / (medium.cp * medium.rho));
  Re = innerDiameter * v * (1 / medium.nue);
  //Laminar or turbulent flow
  if Re > 2100 then
    //Dittus-Boelter equation
    if solid.Q_flow > 0 then
      //Fluid being heated
      Nu = 0.0243 * Re ^ 0.8 * Pr ^ 0.4;
    else
      //Fluid being cooled
      Nu = 0.0243 * Re ^ 0.8 * Pr ^ 0.3;
    end if;
  else
    Nu = 3.65;
  end if;
  h_inner = Nu * medium.lamda / innerDiameter;
  PipeSurface = innerDiameter * Modelica.Constants.pi * lengthPipe;
  computedGc.u = PipeSurface * h_inner;
  ///
  connect(convection.Gc, computedGc.y);
  connect(fluid, convection.fluid);
  connect(convection.solid, solid);
end insideConvection;
```

Figure A.2: `insideConvection` code

Luigi_Model.HeatPump.Components.Evaporator **1**

```
model Evaporator "Thermal model of the evaporator component"
  extends HeatPump.Interfaces.Element1D;
  Modelica.Blocks.Interfaces.RealOutput qFromEvaporator;
  parameter Real tempEvaporator(unit = "degC") = -5 "Temperature of the evaporator";
  HeatPump.CoolantsProperties.SaturationVapour saturationVapour1(temp = tempEvaporator);
  parameter Real pressureCondenser(unit = "kPa") = 400 "Pressure in the condenser";
  HeatPump.CoolantsProperties.SaturationTemperatureTable saturationTemperatureTable1(satPressure = pressureCondenser);
  HeatPump.CoolantsProperties.SaturationLiquid saturationLiquid1;
equation
  h_diff = saturationVapour1.satVapourEnthalpy - saturationLiquid1.satLiquidEnthalpy;
  qFromEvaporator = -m * h_diff;
  connect(saturationTemperatureTable1.satTemperature, saturationLiquid1.tempInput);
end Evaporator;
```

Figure A.3: `evaporator` code

Luigi_Model.HeatPump.Components.Condenser **1**

```
model Condenser "Thermal model of the condenser component"
  extends HeatPump.Interfaces.Element1D;
  Modelica.Blocks.Interfaces.RealOutput qFromCondenser;
  HeatPump.CoolantsProperties.SaturationTemperatureTable
saturationTemperatureTable1(satPressure = pressureCondenser);
  HeatPump.CoolantsProperties.SaturationLiquid saturationLiquid2;
  parameter Real pressureCondenser(unit = "kPa") = 400 "Pressure of the condenser";
equation
  qFromCondenser = -m * h_diff;
  port_b.h = saturationLiquid2.satLiquidEnthalpy;
  connect(saturationTemperatureTable1.satTemperature, saturationLiquid2.tempInput);
end Condenser;
```

Figure A.4: `condenser` code

Luigi_Model.HeatPump.Components.Condenser **1**

```
model Condenser "Thermal model of the condenser component"
  extends HeatPump.Interfaces.Element1D;
  Modelica.Blocks.Interfaces.RealOutput qFromCondenser;
  HeatPump.CoolantsProperties.SaturationTemperatureTable
saturationTemperatureTable1(satPressure = pressureCondenser);
  HeatPump.CoolantsProperties.SaturationLiquid saturationLiquid2;
  parameter Real pressureCondenser(unit = "kPa") = 400 "Pressure of the condenser";
equation
  qFromCondenser = -m * h_diff;
  port_b.h = saturationLiquid2.satLiquidEnthalpy;
  connect(saturationTemperatureTable1.satTemperature, saturationLiquid2.tempInput);
end Condenser;
```

Figure A.5: `compressor` code

Bibliography

[1] Zakia Afroz, GM Shafiullah, Tania Urmee, and Gary Higgins. «Modeling techniques used in building HVAC control systems: A review». In: *Renewable and sustainable energy reviews* 83 (2018), pp. 64–84 (cit. on p. 1).

[2] Marija Trčka and Jan LM Hensen. «Overview of HVAC system simulation». In: *Automation in Construction* 19.2 (2010), pp. 93–99 (cit. on p. 2).

[3] Pengfei Li, Yaoyu Li, John E Seem, Hongtao Qiao, Xiao Li, and Jon Winkler. «Recent advances in dynamic modeling of HVAC equipment. Part 2: Modelica-based modeling». In: *HVAC&R Research* 20.1 (2014), pp. 150–161 (cit. on pp. 2, 5).

[4] Michael Wetter. «Modelica-based modelling and simulation to support research and development in building energy and control systems». In: *Journal of Building Performance Simulation* 2.2 (2009), pp. 143–161 (cit. on p. 2).

[5] Volodymyr B Kopei, R Onysko, and Vitalii G Panchuk. «Component-oriented acausal modeling of the dynamical systems in Python language on the example of the model of the sucker rod string». In: *PeerJ Computer Science* 5 (2019), e227 (cit. on pp. 2, 3).

[6] Peter Fritzson and Vadim Engelson. «Modelica—A unified object-oriented language for system modeling and simulation». In: *European Conference on Object-Oriented Programming*. Springer. 1998, pp. 67–90 (cit. on p. 3).

[7] Abed Al Waheed Hawila, Abdelatif Merabtine, Nadége Troussier, Salim Mokraoui, Abdelhamid Kheiri, and Amine Laaouatni. «Dynamic model validation of the radiant floor heating system based on the object oriented approach». In: *2016 International Renewable and Sustainable Energy Conference (IRSEC)*. IEEE. 2016, pp. 275–280 (cit. on pp. 3, 4).

[8] Hilding Elmqvist. «A Structured Model Language for Large Continuous Systems». eng. PhD thesis. 1978. URL: `https://lup.lub.lu.se/search/ws/files/4602422/8570492.pdf` (cit. on p. 3).

[9] R Franke. «Object-oriented modeling of solar heating systems». In: *Solar energy* 60.3-4 (1997), pp. 171–180 (cit. on pp. 3, 4).

[10] Felix Felgner, Rolf Merz, and Lothar Litz. «Modular modelling of thermal building behaviour using Modelica». In: *Mathematical and computer modelling of dynamical systems* 12.1 (2006), pp. 35–49 (cit. on pp. 4, 5).

[11] Kirill Rozhdestvensky, Vladimir Ryzhov, Tatiana Fedorova, Kirill Safronov, Nikita Tryaskin, Shaharin Anwar Sulaiman, Mark Ovinis, and Suhaimi Hassan. *Computer Modeling and Simulation of Dynamic Systems Using Wolfram SystemModeler*. Springer, 2020 (cit. on pp. 7, 8).

[12] Wikipedia contributors. *Modelica — Wikipedia, The Free Encyclopedia.* [Online; accessed 24-February-2021]. 2021. URL: `https://en.wikipedia.org/w/index.php?title=Modelica&oldid=998516587` (cit. on p. 7).

[13] Peter Fritzson. *Principles of object-oriented modeling and simulation with Modelica 3.3: a cyber-physical approach*. John Wiley & Sons, 2014 (cit. on pp. 8–13).

[14] Michael Tiller. «Modelica by example». In: *Xogency, Web* (2014) (cit. on pp. 13–19, 90).

[15] Ramesh K Shah and Dusan P Sekulic. *Fundamentals of heat exchanger design*. John Wiley & Sons, 2003 (cit. on pp. 39, 40, 43, 44).

[16] Mario Doninelli. «I circuiti ei terminali degli impianti di climatizzazione». In: *Quaderni Caleffi* () (cit. on pp. 40, 42, 53).

[17] *Eng-tips web-page*. URL: `https://www.eng-tips.com/viewthread.cfm?qid=248736` (cit. on p. 45).

[18] Jussi Saari. «Heat exchanger dimensioning». In: *Lappeenranta University of* (2010) (cit. on p. 46).

[19] *Caplor Energy*. URL: `https://www.caplor.co.uk/renewable-heat/heat-pumps-old-page/how-do-air-source-heat-pumps-work/` (cit. on p. 57).

[20] Stephen Wolfram. *An elementary introduction to the Wolfram langauge*. Wolfram Media, Incorporated, 2017 (cit. on p. 72).

[21] *ARERA*. URL: `https://www.arera.it/it/dati/eep35.htm#` (cit. on p. 83).