2019

MODELING OF ELECTRIC WATER HEATER AND ELECTRIC SPACE HEATER

RESEARCH ASSIGNMENT

JAN DUBIŃSKI

SUPERVISOR

Roberto Villafáfila Robles

TABLE OF CONTENT

1. Introduction

The purpose of this paper is to develop an electric water heater model and an electric space heater model. The main parameter of interest is the energy consumption of those devices. In order to achieve this goal two physical models based on temperature equation are created. Those models are then represented using Object Oriented Programming in C++ language.

In the long run the purpose of the models created in this paper is to be integrated into a larger demand side simulation or serve as a starting point for creating more advanced models that will be used in said simulation.

Ultimately an even larger simulation for the purpose of testing and implementing flexibility management in the electric grid will be created. [1]

The code for the models of Electric Water Heater and Space Water Heater was written in C++ in order to make it useful as part of an embedded simulation for microgrids. Such an emulator is described in [2].

The red square on the image below marks the area in which the models described in this paper could find application in relation to the whole scheme of the grid flexibility simulation project.

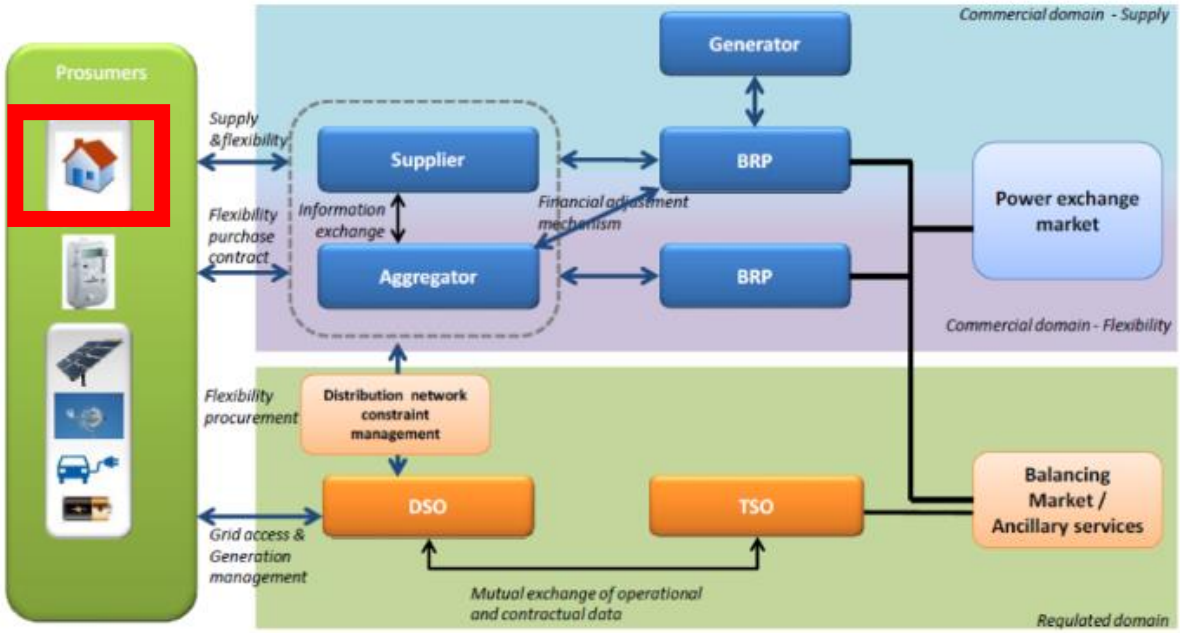

Figure 1 Area of application for the developed models vs whole scheme of grid flexibility simulation project

The Electric Water Heater model is based on a model described in [3] which is widely used in the literature and findings about hot water consumption presented in [4].The Electric Space Heater model is also based on a model described in [3]. However, in calculation of heat losses the methodology presented in polish norm PN-EN 12831 [5] is used.

The development of the class objects in C++ was done using QtCreator IDE.

2. Physical models

2.1 Electric Water Heater

The hearth of this model is a temperature equation which calculates current water temperature in the tank of the heater based on the water consumption, energy input rate and heat losses. Once the temperature of the water is known it is possible to check the power demand of the heater in a given moment. If the temperature is below a given deadband and the heater is off, the heater will turn itself on. If the temperature is above a given deadband and the heater is on, the heater will turn itself off. The power demand coming from the heater is accessed every minute. The power demand can be integrated within a global timestep (default size of this timestep is 15 minutes) to calculate the energy consumption.

2.1.1 Temperature equation

The electric water heater (EWH) model described in this paper is a physically-based single-element EWH model that is presented in [3] . This model is based on energy flow analysis, and the temperature of hot water in the EWH tank can be obtained as a function of time (t) according to the equation:

$$T(t) = T0 \times \mathrm{e}^{-\frac{1}{\mathrm{R'C}}t} + (GR'Tout + BR'Tin + Q \times \eta)\left(1 - \mathrm{e}^{-\frac{1}{\mathrm{R'C}}t}\right)$$

Where:

T0: initial water temperature (°C)
Tin : incoming cold water temperature (°C)
Tout : ambient environment temperature (°C)
T(t): the water temperature inside EWH at time t (°C)
Q: the electric energy input rate (W)
η: efficiency of the electric heater
P: heating element power (W)
R: tank insulation thermal resistance (m^2*°C/W)
SA: tank surface area (m^2)
Volume: the capacity of the tank (m^3)
G: SA/R
F: hot water flow rate (m^3/s)
B=rho*F*Cp
rho: density of water (kg/m3)
Cp: the specific heat of water (J/kg);
C=volume*rho*Cp
R'=1/(G + B)

In the original paper [3] the equation was using imperial units not SI units. An important part of the work on the model consisted of the transforming the units used in the equation to SI.

2.1.2 Hot water consumption profile

Hot water consumption during the day has the most important impact on the power demand of the heater. As described in paragraph 3.1.3 of this work two methods of specifying the hot water consumption profile are provided in this model. First method allows to read the water consumption

profile from a csv file. Second method uses a default minute-by-minute hot water consumption profile that was created for the purpose of this work. This profile is based on the findings presented in [4], according to witch an average Spanish person uses 30 liters of hot water (60°C). The profile is semi-randomized which mean that each time it is created there are some differences between the old and the new profile. This represent the fact that the water usage routine is similar every day, but not identical. The water usage profile is different for weekdays and for weekends. It is also depended on the number of people using the heater and the installed number of showers. The average daily hot water consumption is however always equal to 30 liters of hot water per day per person to match the findings presented in [4]. The purpose of this paragraph is only to describe default water profile. Instructions on using all 2 methods of specifying hot water consumption are described in detail in point 3.1.3 of this work.

Weekday profile

EVENT	START TIME	DURATION	USAGE	QUANTITY
shower	7:00 – 7:30 (+7 min for next shower user)	5 min	4 l/min	1 per user
Tap/sink	During 1 hour after first shower	1 min	0.5l/min -1.5l/min	5 per user
Tap/sink	18:00 -22:00	1 min	0.5l/min – 1.5l/min	5 per user

Table 1Weekend hot water consumption profile

Weekday profile

EVENT	START TIME	DURATION	USAGE	QUANTITY
shower	7:00 – 8:30 (+7 min for next shower user)	5 min	4 l/min	1 per user
Tap/sink	During 1 hour after first shower	1 min	0.5l/min -1.5l/min	5 per user
Tap/sink	14:00 -20:00	1 min	0.5l/min – 1.5l/min	5 per user

Table 2 Weekday hot water consumption profile

2.2 Electric space heater physical model

The electric space heater (ESH) model described in this paper is a modified physically-based single-element model developed for electric water heater that is presented in [3] and was describe in point 2.1. of this paper. This is due to the fact that ESH and EWH work are similar devices. The methodology for obtaining temperature, power demand and energy consumption is the same as the methodology for EWH. First the temperature is calculated, then the control logic checks if the heater is on or off. Finally energy consumption for a global 15-minutes timestep is calculated.

2.2.1 Temperature equation

The hearth of ESH model is an temperature equation similar to EWH temperature equation. Instead of water density and specific heat, density and specific heat of air is used. The volume of the tank is substituted by volume of the heated space. Water consumption is substituted by air circulation due to ventilation needs. Thermal resistance of the tank is substituted by thermal resistance of the heated space. It is however important to notice that in order to know the air circulation additional calculations are required. Thermal resistance of the heated space also requires additional calculations. The model of an ESH is based on energy flow analysis, and the temperature of air in the heated space can be obtained as a function of time (t) according to the equation:

$$T(t) = T0 \times \mathrm{e}^{-\frac{1}{\mathrm{R'C}}t} + (GR'Text + BR'Text + Q \times \eta)\left(1 - \mathrm{e}^{-\frac{1}{\mathrm{R'C}}t}\right)$$

Where:

T0: initial air temperature inside the heated space (°C)
Text = ambient environment temperature (°C)
Th(t): the air temperature inside the heated space (°C)
Q: the electric energy input rate (W)
P: heating element power (W)
Volume: the volume of the heated space (m^3)
G: SAwall/Rwall + SAwindow/RAwindow + SAfloor/Rfloor
R: thermal resistance of areas of heat loss (m^2*°C/W)
SA: surface of areas of heat loss (m^2)
F: air circulation rate (m^3/s)
B=rho*F*Cp
rho: density of air (kg/m3)
Cp: the specific heat of air (J/kg);
C=volume*rho*(Cp + inertia)
R'=1/(G + B)

2.2.2 Building heat inertia

In the process of heating not only the air inside the heated space but also the building is absorbing heat. When there is no heating not only the air, but also the building loses heat. In other words the building acts as a heat magazine. This fact is represented by the *inertia* term that is added to the specific heat of air. This values is calculated based on a heating reduction concept presented in [5] in which it is assumed that 22W per meter square of area are absorbed by the building.

$$Inertia = 22\ \left[\frac{W}{m^2}\right] \times floor\ area\ of\ the\ heated\ space\ x\ 60\ seconds$$

2.2.3 Temperature reduction coefficients

As stated before the calculations of the heat losses are bases on PN-EN 12381 norm. According to these regulations all heat loses are calculated using a temperature difference between the internal and external temperature. Then those heat loses are multiply by the corresponding temperature reduction coefficient [6, 5]. This coefficient represent the fact that the actual temperature difference is in many cases lower than the difference between the internal and external temperature. In this work 2 reduction coefficients are used:

1. **reduction coefficient for the heat losses from the floor surface = 0.5** [7]
   This coefficient represents the assumption that the temperature difference between the air in the heated space and ground is only 50% of the temperature difference between the air in the heated space and external air.

2. **reduction coefficient for the losses coming from air circulation = 0.35** [7]
   This coefficient represents the assumption that the temperature difference between the air in the heated space and air incoming from the ventilation system is only 35% of the temperature difference between the air in the heated space and external air. This difference is lower due to assumed heat recuperation in the ventilation system.

2.2.4 Air circulation rate

Energy required to heat up a space due to air circulation is a substantial part of total energy required for heating. In this work the following approach was chosen: it is assumed that the heated space is a functional building with ventilation. It is also assumed that the ventilation is done properly and provides adequate air circulation. Therefore the air circulation in the heated space is equal to required air circulation corresponding to the described heated space and the needs of its inhabitants. [5]

The air circulation rate is calculated using the following equation:

$$V_{out} = \left(V_{air\ change} + V_{comfort} + V_{CO2} + V_{kitchen} + V_{bathroom}\right) \times reduction\ coefficent$$

Where:

$V_{air\ change}$ - ventilation needs for air change. Air change is a parameter that informs how many times per hour all air in a given space is replaced with fresh air. One air change per hour is assumed in this work for simplicity.

$V_{air\ change} = 1 \times space\ volume$

$V_{comfort}$ - ventilation needed to provide comfort for inhabitants of a given space. [7]

$$V_{comfort} = 0.4 \times S_{floor} + 4 \times number\ of\ inhabitants$$

V_{CO2} - ventilation needed to remove CO2 produced by the inhabitants of a given space. [7]

$$V_{CO2} = 2.8 \times number\ of\ inhabitants$$

$V_{kitchen}$ - extra ventilation needed to remove humidity caused by cooking in the kitchen.

$$V_{kitchen} = 47\ [\frac{m^3}{h}]$$

This value was obtained in the following way:

1. For external air density $\rho = 1{,}2\ \frac{kg}{m^3}$ and humidity $x_n = 3.5\ \frac{g}{kg}$ are assumed.
2. For air in the kitchen in witch cooking is happening a steam flux W=300g/h [7] and humidity x_u=8,8 g/kg (from Moliere diagram for t=20°C and φ=60%) are assumed.
3. Ventilation requirements are calculated using the equation $V = \frac{W}{\rho(x_u - x_n)}$

$V_{bathroom}$ extra ventilation needed to remove humidity caused by laundry

$$V_{bathroom} = 78.6\ [\frac{m^3}{h}]$$

This value was obtained in the following way:

1. For external air density $\rho = 1{,}2\,\frac{kg}{m^3}$ and humidity $x_n = 3.5\,\frac{g}{kg}$ are assumed.
2. For air in the bathroom with a washing machine a steam flux W=500g/h [7] and humidity x_u=14,6 g/kg (from Moliere diagram for t=24°C and φ=80%) are assumed.
3. Ventilation requirements are calculated using the equation $V = \frac{W}{\rho(x_u - x_n)}$

After the ventilation needs are calculated V_{out} is multiplied by reduction coefficient equal to 0.35. [7] This coefficient represents the use of heat recuperation in the ventilation system, which makes the temperature difference between fresh and old air equal to only 35% of the difference between the air outside of and inside of the heated space.

3. Object Oriented Programming model representation

The physical models of EWH and ESH described in part 2 of this work were represented as C++ classes using object oriented programming approach. This part of this work gives information about the usage of each class and provides diagrams that describe their behavior.

3.1 Electric Water Heater class usage

3.1.1 Electric Water heater object creation

Class constructor arguments description

NAME	DESCRIPITON	UNIT	DEFAULT
Power	Electric power of the water heater	Watt	1500
Tset	Set temperature of water in the tank	◦C	60
V	Volume of the water tank	Liter	60
number_of_users	Number of people that are using the water heater	-	2
S	Surface of the water tank	m2	0.9
Tin	Temperature of feed water	◦C	15
Text	Ambient temperature outside of the tank	◦C	22
R	Thermal resistance of the tank	m2*K/W	0.9 [4]
efficiency	Efficiency of electric heating	- (0-1)	0.98
deadband	Band of temperature in which the heater does not change its current behavior. (on/off)	◦C	2.25

Table 3 EWH constructor arguments description

To create a heater object use the class constructor

```
ElectricWaterHeater(float power, float Tset = 60, float V = 60,
                    int number_of_users = 2, float S = 0.9 float Tin = 15,
                    float Text = 22, float R = 0.9, float efficiency = 0.98,
                    float deadband = 2.25)
```

Examples:

```
ElectricWaterHeater ExampleHeater(3000, 60, 100, 3, 14, 23, 0.9, 0.99, 2.25);
```

An electrical water heater with power 3000W, volume of 100 liters, resistance of 0.9 (m2 ∗ (◦C) / W) and heating efficiency of 99% is created. The EWH is used by 2 people, the set temperature is 60◦C and the deadband is equal to 2.25◦C. The inlet water temperature is equal to 15◦C and the external temperature is equal to 22

`power` is the only argument that needs to be specified when creating a heater object. Other arguments take default values if they are not specified. It is possible to specify only a number of arguments but the omitted arguments must come after the specified arguments.

Examples:

```
ElectricWaterHeater ExampleHeater (1500);
```

An electrical water heater with power 1500W and default other properties is created.

```
ElectricWaterHeater ExampleHeater (1500, 50, 100);
```

An electrical water heater with power 1500W, set temperature of water 50 degrees Celsius, volume equal to 100 liters and default other properties is created.

Another possibility to create a heater object is to use the default class constructor which takes no arguments

```
ElectricWaterHeater()
```

Examples:

```
ElectricWaterHeater ExampleHeater;
```
An electrical water heater with all default properties is created.

3.1.2 Editing properties of the heater object

To edit properties of a created heater object use set methods. Those methods are described in the table below.

METHOD	ARGUMENTS / UNIT	DESCRIBTION
set_timestep	Int [minutes]	Edits the global timestep of the simulation. Default timestep is equal to 15 min.
set_set_temperature	float [◦C]	Edits the desired (set) temperature of water in the tank
set_power	float [Watt]	Edits electric power of the heater
set_deadband	float [◦C]	Edits temperature deadband
set_volume	float [liters]	Edits volume of the tank
set_surface	float [m2]	Edits surface of the tank
set_resistance	float [m2*K/W]	Edits thermal resistance of the tank
set_efficiency	float [from 0 to 1]	Edits efficiency of the electric heating
set_external_temperature	float [◦C]	Edits the ambient temperature outside of the tank
set_inlet_temperature	float [◦C]	Edits the temperature of feed water
set_temperature	float [◦C]	Edits the current temperature of the water in the tank
set_start_time	Int [hour] 0-23 Int [quarter] 1-4 *See below for more detail*	!IMPORTANT METHOD! Edits starting time of time simulation. Should be used at the beginning of the simulation Described below in detail.
set_number_of_users	int [-]	Edits number of people using the water tank. When this method is used water usage profile is rebuild for new number of users.
set_number_of_showers	int [-]	Edits number of showers available for users of the tank. When this method is used water usage profile is rebuild for new number of showers.
set_is_it_weekend	bool [false or true]	Decides if it is weekend of weekday. Important for water usage profile. When this method is used water usage profile is rebuild.
set_water_usage_from_file	String [path to file[	Allows to use custom hot water usage profile from a .csv file. When this method is used water usage profile is rebuild.

set_water_usage_default	None	Changes the use of hot water usage profile from a .csv to use of default hot water usage profile

Table 4 EWH class setter methods

Examples:

```
ExampleHeater.set_is_it_weekend (true);
```
Changes the water consumption profile from weekday to weekend

Setting the starting time of the simulation

The behavior of the water heater depends heavily on hot water consumption. Hot water consumption is represented by a semi-randomized water usage profile, different for weekend and for weekdays that stores data about hot water consumption for each minute of the day. Because of that fact it is important to set the starting time of the simulation. The default starting time is midnight. To set starting time of the simulation use the following method:

```
set_start_time(int hour, int quarter)
```

`hour` should be an integer value between 0 and 23 and gives information about the starting hour of the day.
`quarter` should be an integer value between 1 and 4 and give information about the quarter of the selected hour.

Examples:

```
ExampleHeater.set_start_time (6,1);
```
6:00 is selected as a starting time of the simulation

```
ExampleHeater.set_start_time (16,3);
```
16:30 is selected as a starting time of the simulation

3.1.3 Hot water consumption profile

By default the hot water consumption profile described in paragraph 2.1.2 is used. This semi-randomized profile is affected by the following methods: `set_is_it_weeked`, `set_number_of_users`, `set_number_of_showers`. Those methods affect respectively whether the profile for a weekend or for a weekday should be created, the number of people that are using the water and the number of showers that can be taken at the same time.

It Is also possible to load a hot water consumption profile from a csv file. In order to do this `set_water_usage_profile_from_file` method should be used. The file should have the following format:

Number of total water draws events;
Minute of 1. water draw; water consumed;
Minute of 2 water draw; water consumed;
Minute of 3 water draw; water consumed;

For example:

```
1;
359;10;
```

Corresponds to a water consumption profile where water is used only one minute per day – at 6:00 ten liters of water are consumed. A more detailed example of water consumption profile file can be found in Annex C. In order to use the default hot water consumption profile `set_water_usage_default` method should be used.

3.1.4 Calculating next timestep of the simulation

The timestep of the simulation is equal to 15 minutes. To calculate physical properties of the heater after 15 minutes use the following method

```
simulate_timestep(iteration_number)
```

When this method is used energy consumed by the heater, maximum power used, and consumed hot water during a 15 minutes timestep are calculated. Temperature of hot water after the end of the 15-minutes timestep is also calculated. Time of the simulation is also updated, which is important for water consumption profile.

`iteration_number` should be 1 for first timestep, 2 for second etc. This argument is used to keep track of time which is important for the water consumption which changes during the day.

Examples:

```
for (int timestep = 0; timestep < 10; timestep += 1) {
      ExampleHeater.simulate_timestep(timestep); }
```

3.1.5 Accessing values calculated after a time step

To access values of physical parameters calculated by using simulate_timestep method use get methods. Available get methods are described in the table below.

METHOD NAME	RETURNS	UNIT
get_max_power()	Maximal electric power required by the heater during current 15min timestep	Watt
get_temperature()	Current temperature of water	◦C
get_water_flow()	Total water consumed during current timestep	Liter
get_energy_kWh()	Energy consumed during current timestep	kWh
get_energy_joule()	Energy consumed during current timestep	Joule
get_current_minute_of_the_day()	Current minute of the day in the simulation	Minute
get_minute_with_max_power()	Minute of the timestep with maximal power demand	Minute

Table 5 EWH class getter methods

Examples:

```
float T = ExampleHeater.get_temperature();
```

A variable T is created that is equal to current temperature of the water in the tank

3.1.6 Full usage example

```
int main() {
  ElectricWaterHeater ExampleHeater(1500, 50);
// An electric water heater with power 1500W, set temperature and default other
values is created

  ExampleHeater.set_external_temperature(20);
// External temperature is changed to 20 degrees Celsius from default 22

  ExampleHeater.set_start_time(7, 1);
// Starting time of the simulation is changed to 7:00 from default 0:00

  ExampleHeater.set_number_of_users(1);
// Number of users is changed to 1 from default value

  for (int timestep = 0; timestep < 10; timestep += 1) {
// 10 timesteps of 15 minutes are solved. For each timestep:

      ExampleHeater.simulate_timestep(iter);
// all parameters after a 15-minuts timestep are calculated

      cout << ExampleHeater.get_energy_kWh() << ” ; “
// energy in kWh consumed by the heater during the timestep is printed

      << ExampleHeater.get_max_power() << ” ; “
// maximal power in Watts used by the heater during the timestep is printed

      << ExampleHeater.get_temperature() << endl;
// temperature of water after 15-minutes timestep is printed
  }

  return 0;
}
```

3.2 Electric Water Heater class behavior diagram

All the properties of the EWH are specified using the ElectricWaterHeater constructor or by using SETTERS. When a EWH is first created a private method build_water-consumption_profile is called which creates a 96 float vector with water consumption for each minute of the day. This function is also called whenever a SETTER modifying water consumption profile is used to rebuild the vector. To calculate a timestep of the simulation and calculate the energy consumption the simulate_timestep method should be used. This method calls the calculate_power_method which takes the volume of water that is used for current minute from the water consumption profile vector and checks if the heater should be on or off. Then calculate_power calls calculate_temperature that upadtes the temperature of water inside the heater for current Q updated by calculate_power, water usage taken from water_consumption profie for current time and other properties. This loop is repeated for every minute of the global timestep and after each global timestep the energy consumption is calculated. The values of energy consumption, temperature and max power after a timestep can be accessed using GETTERS.

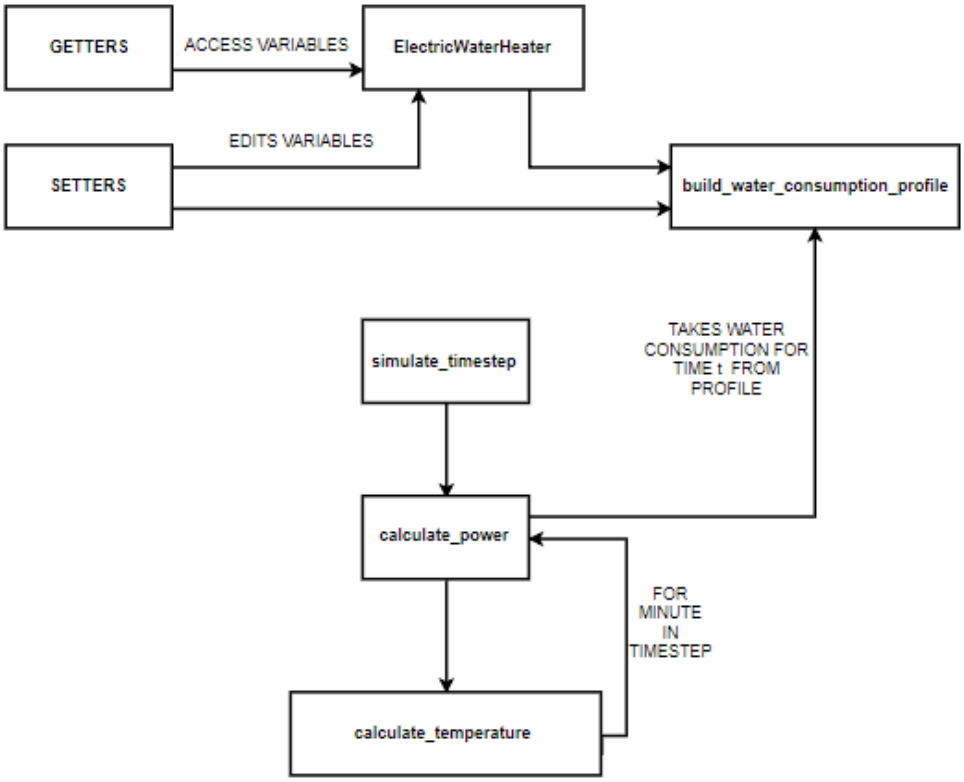

Figure 2 EWH class behavior diagram

3.3 Electric Space Heater class usage

3.3.1 Electric Space Heater object creation

Class constructor arguments description

NAME	DESCRIPITON	UNIT	DEFAULT
Power	Electric power of the space heater	Watt	1500
Tset	Set temperature of air in the heated space	◦C	21
Volume	Volume of the heated space	m^3	5
number_of_inhabitants	Number of people that are in the heated space	-	2
Text	Ambient temperature outside of the heated space	◦C	10
Efficiency	Efficiency of electric heating	- (0-1)	0.98
deadband	Band of temperature in which the heater does not change its current behavior. (on/off)	◦C	2.25
has_thermostat_control	If false the heater is always ON. If true the heater keeps the temperature within the deadband and has a control loop.	-	true

Table 6 ESH constructor arguments description

To create a heater object use the class constructor

```
ElectricSpaceHeater(float power, float Tset = 21, float volume = 200,
                    int number_of_inhabitants = 2, float Text = 10,
                    float efficiency = 0.98, float deadband = 2.25,
                    bool has_thermostat_control = false)
```

Examples:

```
ElectricSpaceHeater ExampleHeater(3000, 21, 75, 4, 10, 0.97, 2.25, true);
```

An electric space heater with power 3000W with efficiency of 98% is created. The set temperature of air is 21◦C with deadband 2.25 ◦C. The heater has a control loop. The heated space has volume of 200 m3 and is inhabited by 4 people. The outside temperature is equal to 10◦C

`power` is the only argument that needs to be specified when creating a heater object. Other arguments take default values if they are not specified. It is possible to specify only a number of arguments but the omitted arguments must come after the specified arguments.

Examples:

```
ElectricSpaceHeater ExampleHeater (1500);
```

An electrical space heater with power 1500W and default other properties is created.

```
ElectricSpacHeater ExampleHeater (1500, 22);
```

An electrical space heater with power 1500W, set temperature of air 22 degrees Celsius and default other properties is created.

Another possibility to create a heater object is to use the default class constructor which takes no arguments

```
ElectricSpaceHeater()
```

Examples:

```
ElectricSpaceHeater ExampleHeater;
```

An electric space heater with all default properties is created.

3.3.2 Specifying areas of heat loss

Unlike a water heater object, space heater object can NOT be used right after creation. It is necessary to specify areas of heat loss. Know dimensions of heated space do not give information about which walls are adjected to external conditions (external temperature). In order to understand the problem examine the following case: a room with dimensions L:4m x W:4m x H:3m can have 1, 2, 3 or 4 walls exposed to external conditions. Some of the walls may be adjected to other rooms/ flats or building. That is why 4 separate methods are implemented to specify areas of heat loss that are used in the simulation: create_wall_surface, create_floor_surface, create roof surface and create_window_surface.

Setting the wall surface

```
void create_wall_surface(float surface, float resistance = 1. / 0.3)
```

`surface` must be equal to total surface (m2) of walls exposed to external conditions **minus the area of windows.**
`resistance` must be equal to thermal resistance of the wall material (m2 * K) / W. The default value is equal to 1/0.3 (m2*K)/W. [7] This value is proposed for solid, normally isolated walls.

Setting the window surface

```
void create_window_surface(float surface, float resistance = 1. / 1.3)
```

`surface` must be equal to total surface (m2) of glass windows or glass doors exposed to external conditions.
`resistance` must be equal to thermal resistance of the windows (m2 * K) / W. The default value is equal to 1/1.3 (m2*K)/W. [7]

Setting the roof surface

```
void create_roof_surface(float surface, float resistance = 1. / 0.3)
```

`surface` must be equal to total surface (m2) of roof exposed to external conditions.
`resistance` must be equal to thermal resistance of the roof (m2 * K) / W. The default value is equal to 1/0.3 (m2*K)/W. [7]

Setting the floor surface

```
void create_floor_surface(float surface, float resistance = 1. / 0.5)
```

`surface` must be equal to total surface (m2) the floor.
`resistance` must be equal to thermal resistance of the floor (m2 * K) / W. The default value is equal to 1/0.5 (m2*K)/W. [7] The actual resistance of the floor surface that is set using this function is equal to 2* `resistance`. The value 2 is temperature difference reduction coefficient. It represents the

assumption that the temperature difference room ground is only 50% of the temperature difference room- outside.

When this method is used air circulation is recalculated for new floor area. This is due to the fact that surface area affects the ventilation needs of the heated space.

When this method is used Inertia is calculated. It is equal to 60 * 22 / m2 of surface and represents the heat that is absorbed by the building not by the air in the heated space and is later released when the building acts as a heat magazine.

It is important to remember that after creating the heater object all 4 previously described methods MUST be used or the model will not work properly.

Examples:

```
ExampleHeater.create_floor_surface(100);
ExampleHeater.create_roof_surface(100);
ExmapleHeater.create_wall_surface(100);
ExampleHeater.create_window_surface(10, 1/1.5);
```

A wall surface 100m2. R =1/0.3, a roof surface 100m2. R =1/0.3, a window surface 10m2, R=1/1.5 and floor surface 100m2, R=0.5 are created.

3.3.3 Editing properties of the heater object

To edit properties of a created heater object use set methods. Those methods are described in the table below.

METHOD	ARGUMENT / UNIT	DESCRIBTION
set_timestep	Int [minutes]	Edits the global timestep of the simulation. Default timestep is equal to 15 min.
set_set_temperature	float [◦C]	Edits the desired (set) temperature of air in the heated space
set_power	float [Watt]	Edits electric power of the heater
set_deadband	float [◦C]	Edits temperature deadband
set_volume	float [m3]	Edits volume of the heated space. When this method is used air circulation is recalculated for new volume. This is due to the fact that volume of the space affects the ventilation needs of the heated space.
set_efficiency	float [from 0 to 1]	Edits efficiency of the electric heating
set_external_temperature	float [◦C]	Edits the ambient temperature outside of the heated space
set_temperature	float [◦C]	Edits the current temperature of the air in the heated space
set_start_time	int [hour] 0-23 int [quarter] 1-4 *see below for more details*	!IMPORTANT METHOD! Edits starting time of time simulation. Should be used at the beginning of the simulation Described below in detail.
set_number_of_inhabitants	Int[-]	Edits number of people in the heated space. When this method is used air circulation

		is recalculated for new number of inhabitants. This is due to the fact that ventilation needs depend on the number of people in the space.
set_has_kitchen	bool [false or true]	Decides if there is a kitchen in the heated space. When this method is used air circulation is recalculated. This is due to the fact that ventilation needs are higher when the humidity from cooking needs to be removed.
set_has_bathroom	bool [false or true]	Decides if there is a bathroom in the heated space. When this method is used air circulation is recalculated. This is due to the fact that ventilation needs are higher when the humidity from laundry needs to be removed.
ext_temp_from_file	String [path to file]	Uses a hour-by-hour external temperature profile loaded from a file
ext_temp_default	float [◦C]	Uses constant external temperature
set_termostat_control	bool [false or true]	Decides if the space heater has a control loop or not. If false the heater is always ON. If true the heater turns itself ON and OFF to keep the temperature in the heated space within the deadband. **!THIS IS AN IMPORTANT METHOD AS IT CHANGES THE BEHAVIOUR OF THE HEATER!** The two modes of heater operation are described in the later part of the documentation.

Table 7ESH setter methods

Examples:

```
ExampleHeater.set_has_bathroom(true);
```
the air circulation to take into account increased ventilation needs because of the laundry humidity.

Setting the starting time of the simulation

If an external temperature profile is loaded from a csv file the behavior of the space heater changes because of the changes in the external temperature. Because of that fact it is important to set the starting time of the simulation. The default starting time is midnight. To set starting time of the simulation use the following method:

```
set_start_time(int hour, int quarter)
```

`hour` should be an integer value between 0 and 23 and gives information about the starting hour of the day.
`quarter` should be an integer value between 1 and 4 and give information about the quarter of the selected hour.

3.3.4 External temperature profile.

By default the external temperature is constant. It can be changed manually by using `set_external_temperature` method. It is also possible to use an external temperature profile witch will make the external temperature change over time. This profile has to be uploaded from a csv file using `ext_temp_from_file` method .This file should consist of 24 values corresponding to the temperature for a given hour of the day, separated by ";". An exemplary file is presented in Annex C of this work. To switch back to using constant values of external temperature ext_`temp_default` method should be used.

3.3.5 Thermostat control loop vs. always on

Unlike an electric water heater, there are 2 modes of operation of an electric space heater. In the first mode the heater is always ON. This represents the behavior of a simple electric heater that is connected to a power source and works until it is disconnected. In the other mode the space heater has a control loop and keeps the air temperature within a given deadband. This represents the behavior of an electric heating system with a thermostat that controls the heating.

The Boolean variable `has_thermostat_control` is responsible for defying in which mode the heater object operates. Its default values is false, which means that by default the heater is always on. This variable can be changed by using `set_has_thermostat_control` method.

3.3.6 Sizing the power of the heater

The ElectricSpaceHeater class has `a find_correct_power` method which allows to find the correct electric power of the heater for specified size of the heated space, heat losses and other conditions. This method was implemented because in many cases only the properties of the building/room/flat, desired internal temperature and external temperatures are known and the right heater power is not known. The user may create a heater object specifying a random or guessed power and then use `find_correct_power` method that will iteratively find the power of the heater that matches the conditions specified during the creation of the heater and specifying areas of heat loss

```
void find_correct_power(int power_search_step = 50)
```

`power_search_step` defines what values of electric power will be checked. For example for `power_search_step = 50` heaters with electrical power n*50 W will be checked (50, 100, 150 … etc). This parameter was created because actual heater power available on the market is not a continues value. In other words - it is possible to buy 1000W or 1200W space heater, but not 1038W space heater.

Examples:

```
//A heater used to heat a room with dimensions 10x10x2.5 and keep a
temperature equal to 20 degrees is created. The power of this heater is
equal to 0W.
ElectricSpaceHeater Asia(0, 20, 10, 10, 2.5);

//The number of inhabitants is set to 3
ExampleHeater.set_number_of_inhabitants(3);

//Floor heat loss surface is set to 100m2
```

```
ExampleHeater.create_floor_surface(100);

//Wall heat loss surface is set to 100m2
ExampleHeater.create_wall_surface(100);

//Window heat loss surface is set to 10m2
ExampleHeater.create_window_surface(10);

//External temperature is set to 10 degrees
ExampleHeater.set_external_temperature(10);

//After this method is called the power of the heater is adjusted from 0W
to a power that will keep the temperature in the heater space at Tset=20
level. The best electric power for this application will be found from
n*100W(100W, 200W … etc.)
Example.find_correct_power(100);

cout << ExampleHeater.get_power()<<endl; //The adjusted power is printed
```

3.3.7 Calculating next timestep of the simulation

The timestep of the simulation is equal to 15 minutes. To calculate physical properties of the heater after 15 minutes use the following method

```
simulate_timestep(iteration_number)
```

When this method is used energy consumed by the heater, maximum power used, and air circulation during a 15 minutes timestep are calculated. Temperature of air inside the heated space after the end of the 15-minutes timestep is also calculated. Time of the simulation is also updated.

`iteration_number` should be 1 for first timestep, 2 for second etc. This argument is used to keep track of time which is important for time tracking purposes.

Examples:

```
for (int timestep = 0; timestep < 10; timestep += 1) {
      ExampleHeater.simulate_timestep(timestep); }
```

3.3.8 Accessing values calculated after a time step

To access values of physical parameters calculated by using simulate_timestep method use get methods. Available get methods are described in the table below.

METHOD NAME	RETURNS	UNIT
get_power()	Electric power of the space heater	Watt
get_max_power()	Maximal electric power required by the heater during current 15min timestep	Watt
get_temperature()	Current temperature of water	◦C
get_air_flow()	Total air circulation during current timestep	m3
get_energy_kWh()	Energy consumed during current timestep	kWh
get_energy_joule()	Energy consumed during current timestep	Joule
get_current_minute_of_the_day()	Current minute of the day in the simulation	Minute
get_minute_with_max_power()	Minute of the timestep with maximal power demand	Minute

Table 8 ESH getter methods

Examples:

```
float T = ExampleHeater.get_temperature();
```

a variable T is created that is equal to current temperature of air in the heated space

3.3.9 Full usage example

```
int main() {
  ElectricSpaceHeater ExampleHeater(0, 20, 10, 10, 2.5);
//A heater used to heat a room with dimensions 10x10x2.5 and keep a temperature
equal to 20 degress is created. The power of this heater is equal to 0W.

  ExampleHeater.set_number_of_inhabitants(3);
//The number of inhabitants is set to 3

  ExampleHeater.create_floor_surface(100);
//Floor heat loss surface is set to 100m2

  ExampleHeater.create_wall_surface(100);
//Wall heat loss surface is set to 100m2

  ExampleHeater.create_window_surface(10);
//Window heat loss surface is set to 100m2

  ExampleHeater.set_external_temperature(10);
//External temperature is set to degrees 10 Celsius.

  ExampleHeater.find_correct_power();
//After this method is called the power of the heater is adjusted from 0W to a
power that will keep the temperature in the heater space at Tset=20 level. The
best electric power for this application will be found from n*50W(50W, 100W …
etc.)

   << ExampleHeater.get_power() << ” ; “
// power of the heater in Watts is printed

  for (int timestep = 0; timestep < 10; timestep += 1) {
// 10 timesteps of 15 minutes are solved. For each timestep:

      ExampleHeater.simulate_timestep(iter);
// all parameters after a 15-minuts timestep are calculated

      cout << ExampleHeater.get_energy_kWh() << ” ; “
// energy in kWh consumed by the heater during the timestep is printed

      << ExampleHeater.get_max_power() << ” ; “
// maximal power in Watts used by the heater during the timestep is printed

      << ExampleHeater.get_temperature() << endl;
// temperature of air after 15-minutes timestep is printed
  }

  return 0;
}
```

3.4 Electric Space Heater class behavior diagram

All the properties of the ESH are specified using the ElectricSpaceHeater constructor or by using SETTERS. When a EWH is first created a private method calculate_air_circulation is called which calculates the air flow rate in the heated space. This function is also called whenever a SETTER modifying ventilation needs is used to rebuild the vector. To calculate a timestep of the simulation and calculate the energy consumption simulate_timestep method should be used. This method calls the calculate_power_method which takes the external temperature that is used for current minute from the external temperature profile vector (if it is upload from a file, if not uses constant value) and checks if the heater should be on or off. Then calculate_power calls calculate_temperature that upadtes the temperature of air inside the heated space for current Q updated by calculate_power and other properties. This loop is repeated for every minute of the global timestep and after each global timestep the energy consumption is calculated. The values of energy consumption, temperature and max power after a timestep can be accessed using GETTERS.

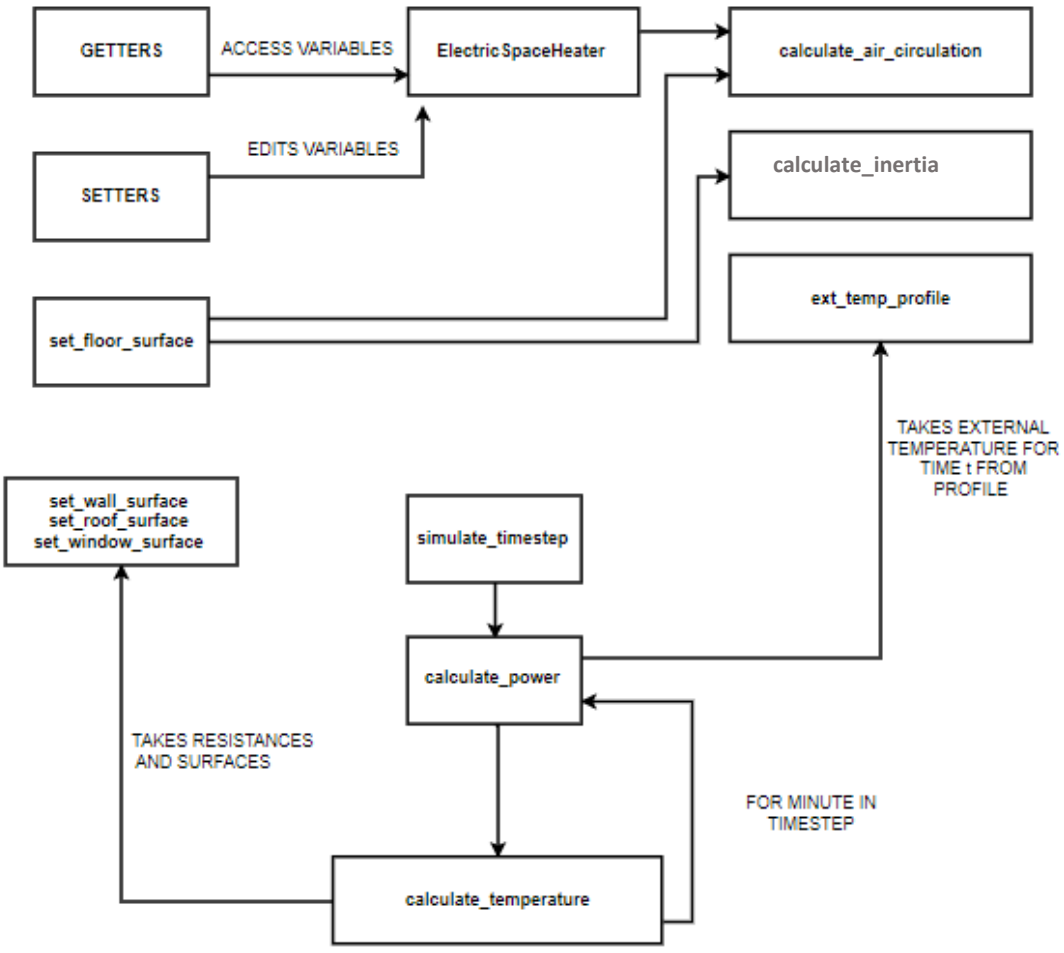

Figure 3 ESH class behavior diagram

4. Exemplary results

This paragraph presents the results of a 24h simulation using an exemplary EWH and an exemplary ESH. The results are presented in form of graphs.

4.1 Results for an Electric Water Heater

The simulated water heater had the following properties:

Power: 2kW
Volume: 60l
Set temperature 60◦C
External temperature 22◦C
Feed water temperature: 15◦C
R: 0.9 (m2*K)/W
S: 0.9 m2
Efficiency: 98%
Number of users: 2
Number of showers: 1
Default hot water usage profile
Deadband 2.25◦C

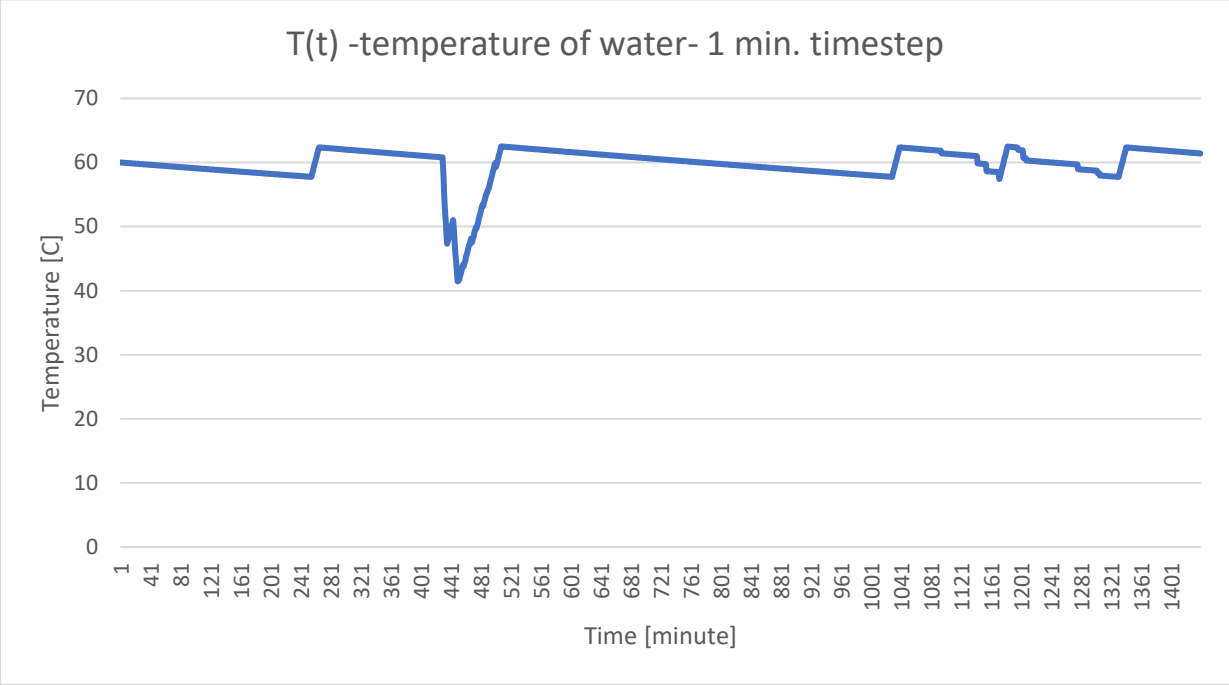

Figure 4 T(t) -temperature of water- 1 min. timestep

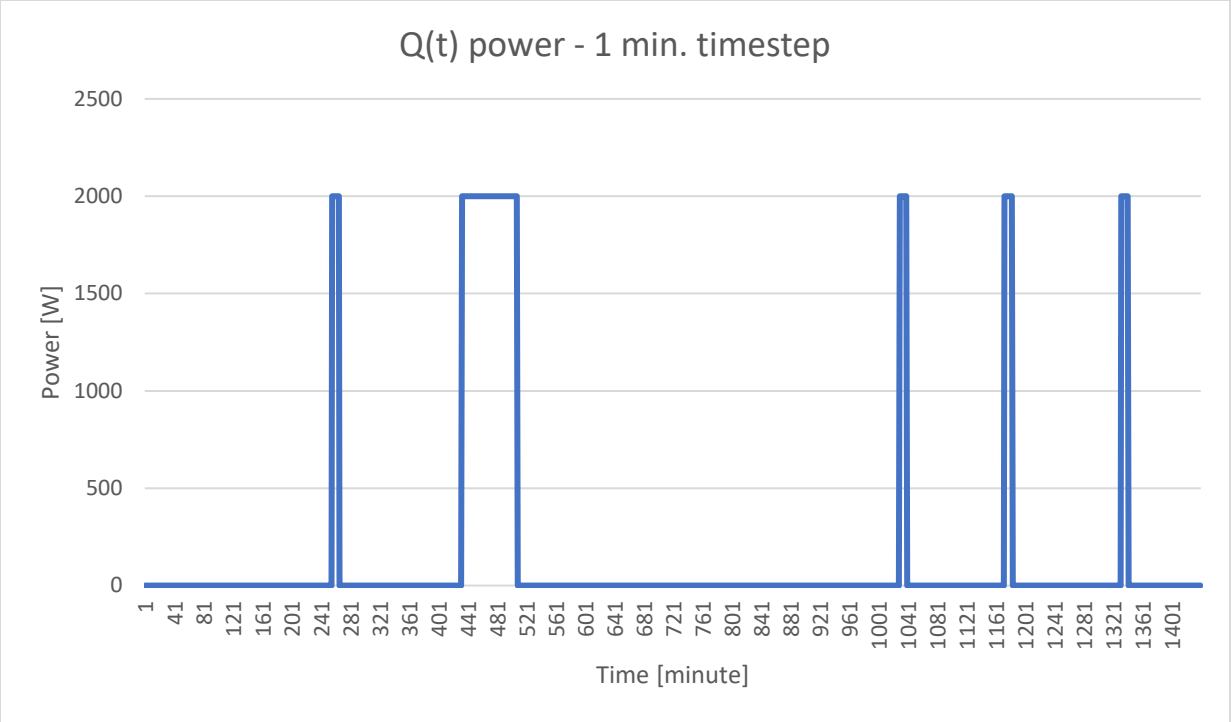

Figure 5 Q(w) -power- 1 min. timestep

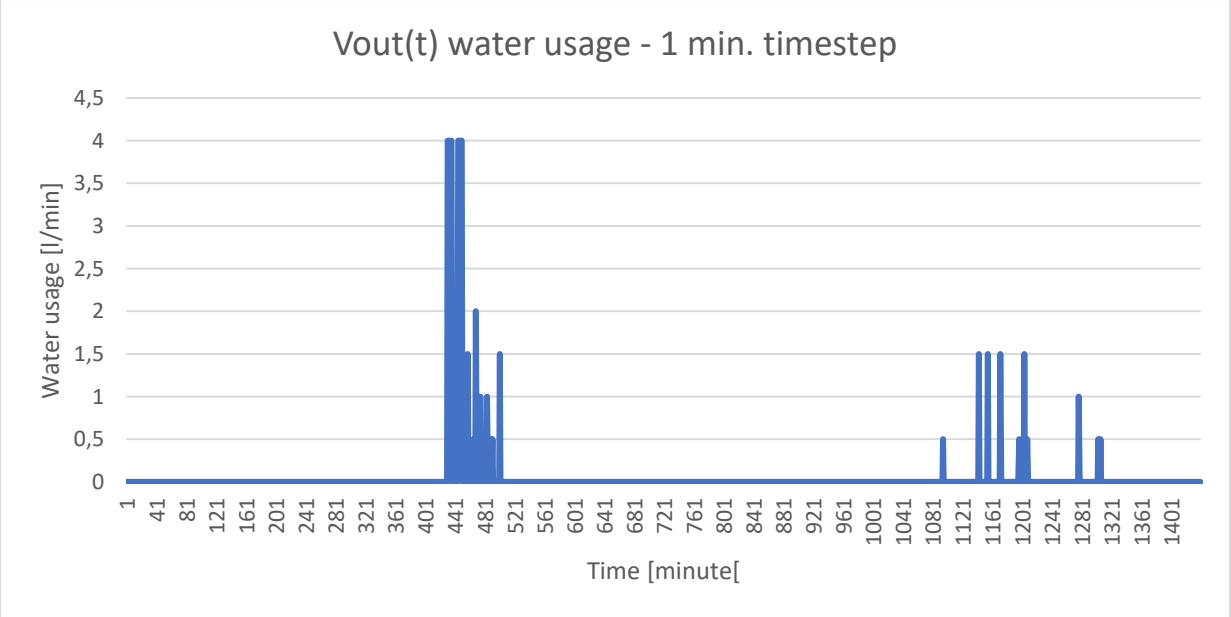

Figure 6 Vout(t) hot water usage - 1min. timestep

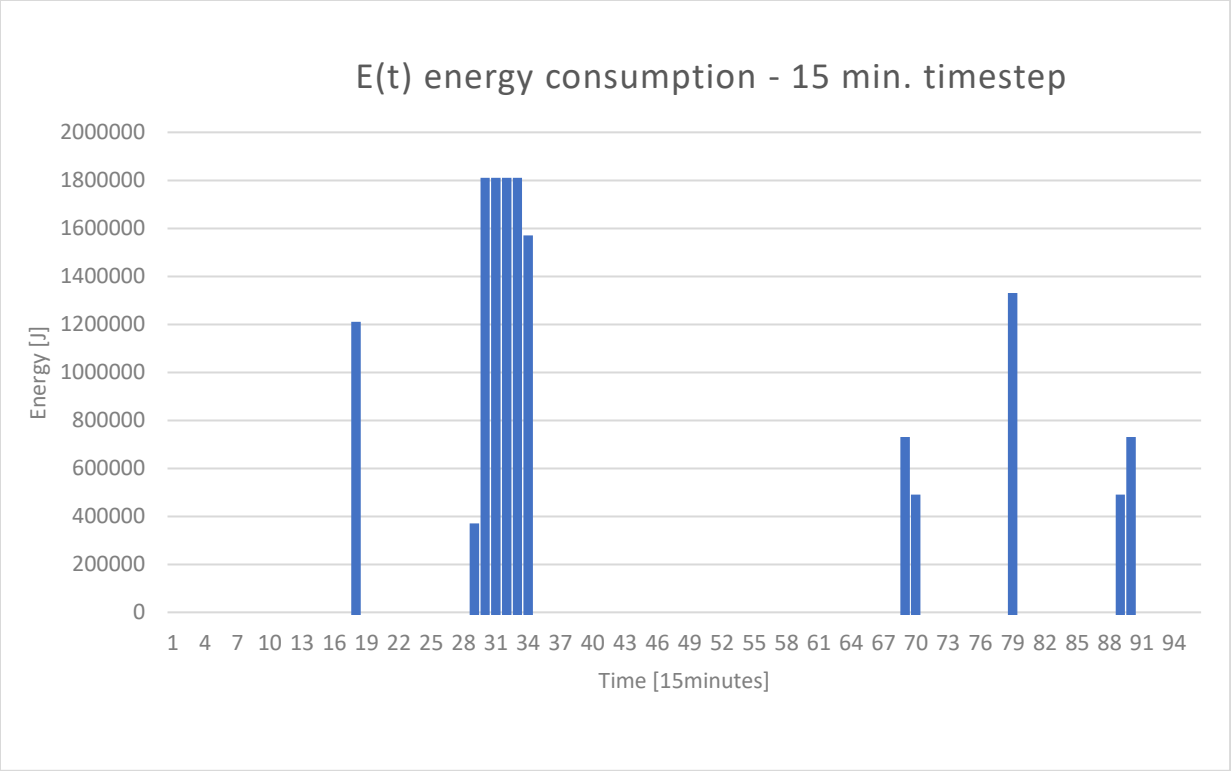

Figure 7 E(t) energy consumption - 15 min. timestep

4.1 Results for an Electric Space Heater

The simulated space heater had the following properties:

Power: 500W
Volume of the heated space: 30m3
Set temperature 21◦C
External temperature 0◦C
Areas of heat losses:
Rwall: 3.33 (m2*K)/W
Swall: w0 m2
Rdloor 1/.0.5 (m2*K)/W
Sfloor: 10 m2
Rwindow: 1/1.3 (m2*K)/W
Rwindow: 10 m2
Rroof: 3.33 (m2*K)/W
Swall: 0 m2
Efficiency: 98%
Number of users: 2
Deadband 2◦C

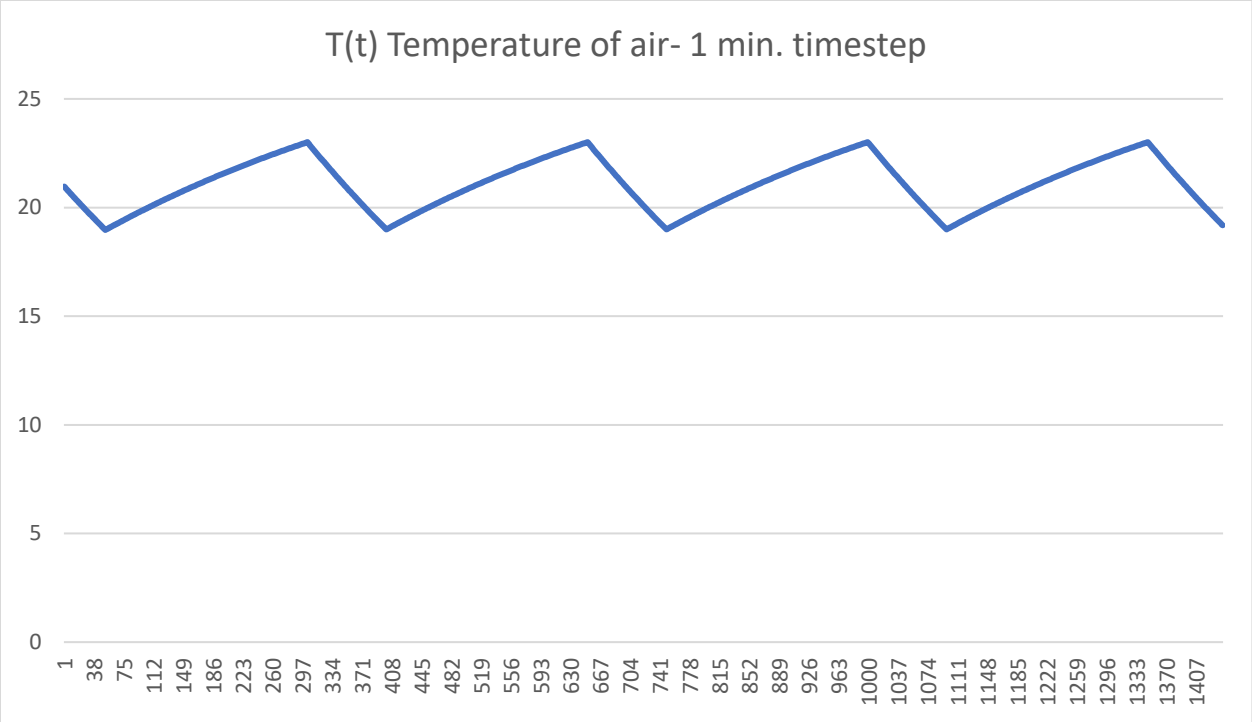

Figure 8 T(t) temperature of air 1 min timestep

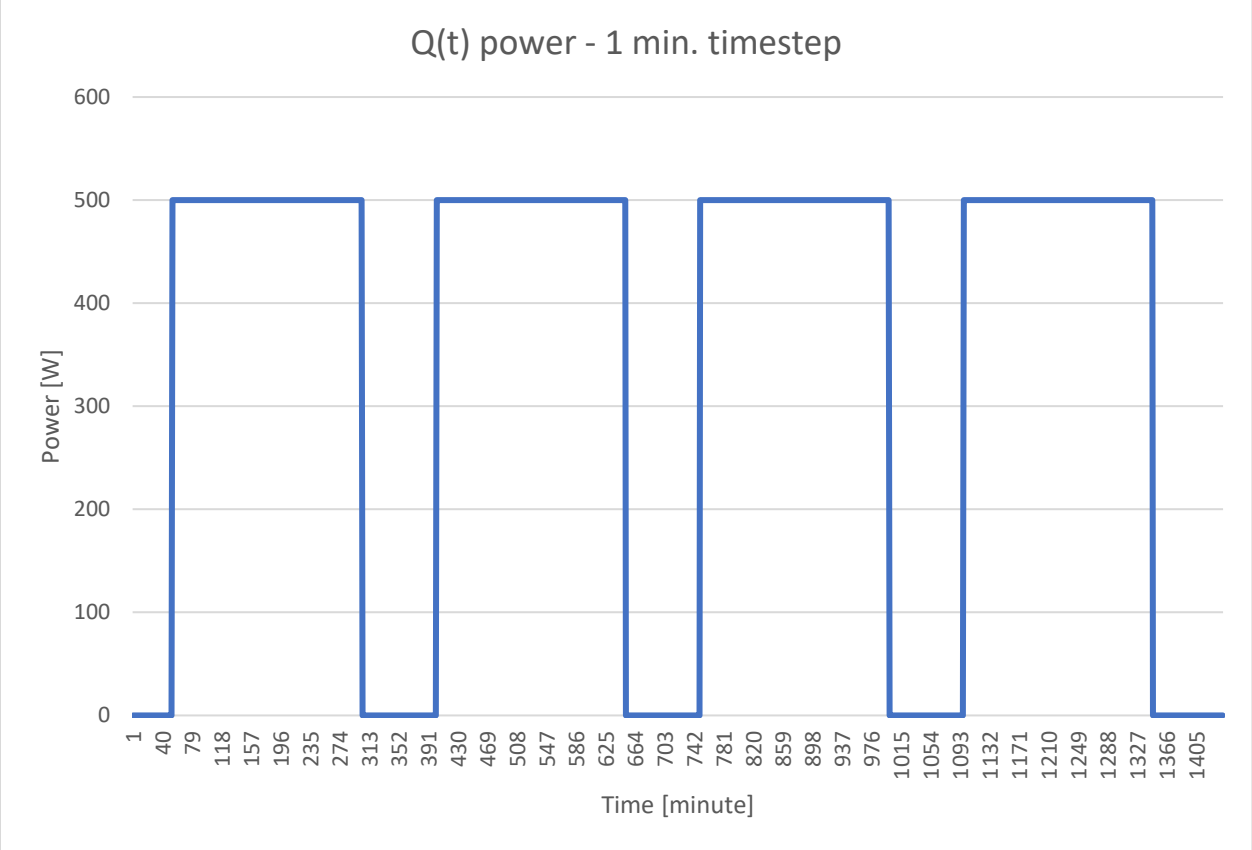

Figure 9Q(t) power - 1min. timestep

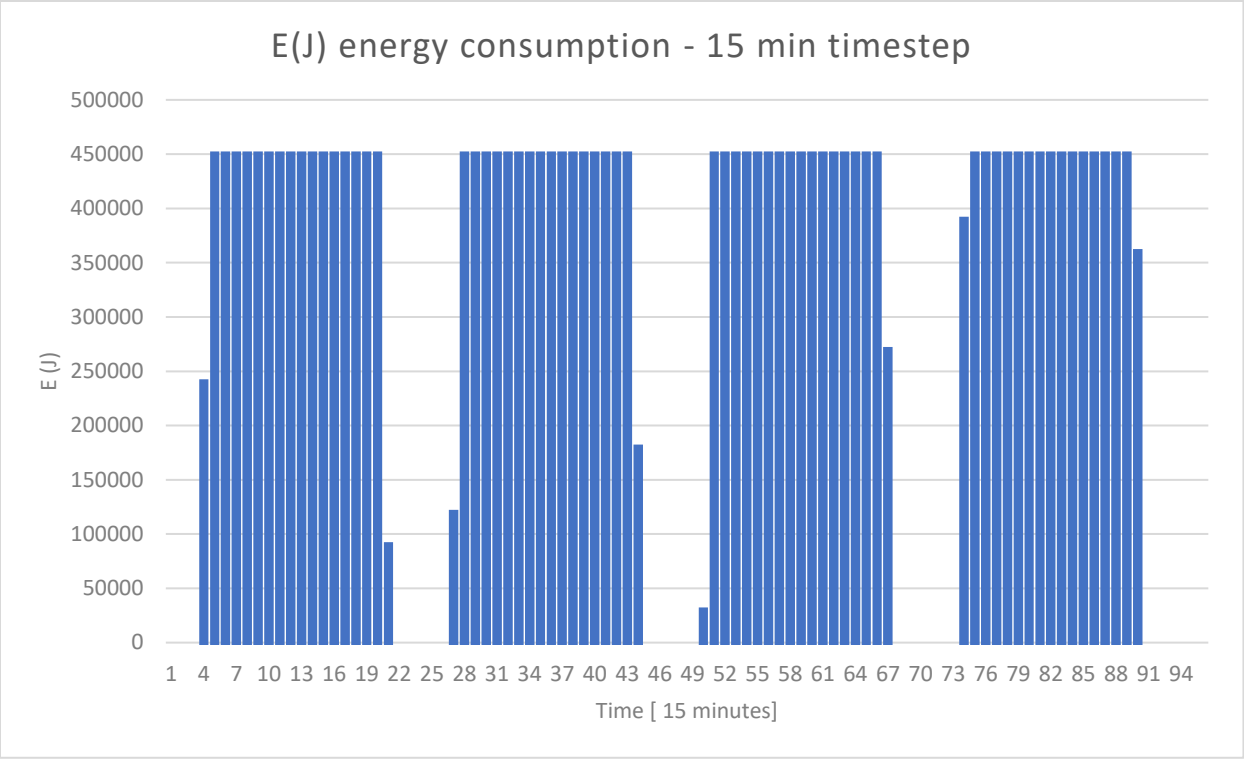

Figure 10 E(j) Energy consumption- 15 min timestep

5. Conclusions

The main goal of this assignment was achieved with satisfying results as functioning models for a EWH and ESH were created. The models were tested and the results of the conducted simulations are plausible. The code is well documented and with instructions presented in this paper it is possible to use those models and develop them even further.

The results of the simulations presented in point 4. of this work are not surprising and where expected from previous engineering knowledge about the behavior of heater devices.

Further improvements of the developed models that could be implemented in the future:

For the EWH:

1. A more detailed hot water usage profile could be implemented. Water consumption has the greatest impact on the behavior of the EWH so developing a more detailed and realistic daily water consumption profile would increase the quality of the model. The existing implementation of a method that allows to read the hot water consumption profile from a file makes potential new hot water consumption profile easy to integrate in the model.

For the ESH:

The heat losses have the greatest impact on the behavior of the ESH. Because of that:

1. A more detailed method of calculating air circulation could be implanted, possibly taking into account the external air infiltrating the building.
2. A better method of calculating heat losses from the surface of the building would further improve the quality of the model, as the method currently implemented does not take into account heat losses coming from other neighboring rooms / staircases etc.
3. Taking into account other sources of heat in the heated space including natural sources such as sun radiation.

As for both models it is possible that the implementation of the physical models using C++ language could be done in more efficient way. It has to be noted that the author of this work has not worked with Object Oriented Programming in C++ before. Although this work resulted in significant improvement on this field there is without a doubt a room for further improvements.

References

[1] P. O. –. U. G. T. B. –. e. s. J. T. P. R. –. E. Pau Lloret, Overall INVADE architecture, 2017.

[2] P. O.-R. M. C.-M. R. V.-R. O. G.-B. E. Prieto-Araujoa, Renewable energy emulation concepts for microgrids.

[3] R. J. D. A. P. D. J. H. M. Hashem Nehrir, Power Management of Aggregate Electric Water Heater Loads by Voltage Contro.

[4] L. A. J. S. E. Fuentes∗, A review of domestic hot water consumption profiles for application in systems and buildings energy performance analysis.

[5] M. Strzeszewski, Norma PN–EN 12831, Nowa metoda obliczania projektowego obciążenia cieplnego.

[6] d. i. M. Strzeszewski, „Współczynnik redukcji temperatury w metody obliczania obciążenia cieplnego wg PN-EN 12831”.*Warsaw University of Technology.*

[7] F. o. E. a. A. E. Warsaw University of Technology, Taken from materials from a course in Heating/Ventilation on WUT.

List of figures

List of tables