Practical course Smart Energy Systems Report

Sandro Speth

Markus Zilch

Dominik Wagner

Wintersemester 18/19

1 Introduction

The traditional power grid is changing more and more over time. Due to increasing sensitization for the use of renewable and reliable sources of energy instead of nuclear power sources or fossil energy sources, there is an increasing accommodation of renewable energy. To fulfill our daily energy need only with such energy sources is quite difficult and needs lot of planning and simulation. In this work we build a smart energy system to simulate a smart grid.

A smart grid is an energy efficient system with information and communication technology, automation and awareness of energy consumption. There are many different actors and technologies which are connected to each other and interoperate to optimize the grid.

A smart energy system creates the bridge between a power grid and a resilient and reliable smart grid. Users can simulate reliable energy sources, as well as different kinds of energy consumers, e.g. homes or offices. Simulation of distributed energy sources and automation of processes build an energy management system with interaction on different levels of the power grid and the consumer grids. In a smart grid a new aspect is becoming important. Contrary to currently used normal power grids the exchange of information becomes more and more important. Especially the consumer side (demand side) of these grids are now being integrated into the power systems in order to ensure optimal power usage and a better understanding of customer needs. Those new smart grids use various methods to control the flow of energy in order to optimize efficiency and grid usage. For a fully functional smart grid, the grid has to be made more resistant to any attacks and have mechanisms to self-diagnose problems and take appropriate recovery actions. The new smart grids offer better ways to integrate renewable but unstable power suppliers like windturbines and solarpanels through better controllable interactions on a much smaller scale than conventional power grids. Through these microgrids we can possibly rely completely on renewable energy sources in the future. This can be checked with our smart energy system.

The report is structured as follows. In section 2 we present our system design. First, we give some basic foundations which are relevant for our smart energy system, such as the difference between kWh and kW. Afterwards we present our functional requirements for the smart energy systems as user stories. From these functional requirements we created our architecture which will be described in this section as well. The next section contains the theoretical background to understand the implementation.

2 System Design

2.1 Difference between kW and kWh

W is a measuring scale for energy applied per time instance. There are different possibilities to describe W in common terms. A pretty graphic one is the movement of mass. $1W$ equals $1kg$ of mass moved by 1 meter in one second: $1\frac{kg*m^2}{s^3}$. Or in electrical terms: $1W$ equals 1 Ampere of electrical power with a voltage of 1 Volt. Both of those formulas are equal to a much simpler Term for Watt: $1W = 1J/s$. In simple terms, 1 Watt is the same as one Joule of energy applied over 1 second. For completeness, $1kW = 1000W$ [12, 3, 14].

Wh are the common term for measuring energy consumption and production. $1Wh$ is $1W$ applied continuously over 1 hour. $1Wh = 1W * 1h = 1J/s * 3600s = 3600J$. For a scientific context the Wh therefore is simply not used, instead the common SI standard J is used. In comparison, Wh is the total amount of energy used. W is how much energy is used in a specified timeslot (mostly 1 second) [8, 4].

2.2 Difference between consumption and demand

Electricity consumption and electricity demand are two different properties and measured with different measurement units. The following section contains a description of both and an example at the end.

2.2.1 Demand

The demand is the rate of consumption of electricity or mathematical speaking the demand is the derivation of the consumption [15]. Most of the time the demand is measured in Watt. If you turn on a 100W light bulb, it will demand 100W while it is turned on. At the same time the grid must provide electricity at a rate of 100W. In most cases it is possible to calculate the demand with the following formula [10].

$$Demand = Voltage * Current$$

Some customers also have to pay for the demand or peak demand they have because if you have a higher (peak) demand the grid has to support this [15, 6].

2.2.2 Consumption

It is easier to understand electricity consumption because we are more used to this concept [15]. Many people deal with electricity consumption while paying their electricity bill because most German electricity meter measure only the consumption. The consumption is the amount of

electricity used per time unit [15][6]. Most of the time the consumption is measured in kilowatt per hour. The formula to calculate the consumption is the following [15].

$$Consumption = Demand * Time$$

For example, 5 W LED bulbs turned on for 1h have the consumption of 5 Wh.

2.2.3 Difference

Figure 1: Example for demand and consumption

The demand is the rate of which we use energy and the consumption is the total energy used for a given time frame [15]. The formulas also show this relation between the consumption and the demand. The consumption is the demand multiplied with the time. If someone turns on 1 heating unit with a demand of 1kW for 2 hours than the demand during this hours is 1kW, but the consumption is 2kWh. The consumption is the same if two heating units are used for half an hour, but the demand is doubled (2kW). Figure 1 contains another but similar example. To put it in simple terms the demand is comparable with the speed of a car and the consumption is the distance you drive. The faster you drive the more distance is accumulated over time.

2.3 Userstories

1. As a user, I need to create one or more wind turbines in the simulation so that I can calculate the potential energy output.

2. As a user, I need to create one or more photovoltaic panels in the simulation so that I can calculate the potential energy output.

3. As a user, I need to create one or more batteries in the simulation so to save unused energy of in my simulation.

4. As a user, I need to remove one or more suppler modules like wind turbines form the simulation to get an optimal mix of all kinds of suppliers.

5. As a user, I need to remove one or more consumer modules like commercial buildings form the simulation to get an optimal mix of all kinds of consumers.

6. As a user, I need to remove one or more batteries form the simulation to remove batteries which were added by mistake.

7. As a user, I need to get the charging state of my batteries to know the impact of the energy storage on the grid.

8. As a user, I need to create one or more homes on the demand side in the simulation so that I can simulate some energy consumer.

9. As a user, I need to create one or more commercial buildings on the demand side in the simulation to simulate some high energy consumer.

10. As a user, I need to get dynamic energy prices calculated from the simulation to determine if I want to sell my produced energy or store it for later use.

11. As a user, I need to use weather data in the simulation to simulate the smart energy system more precise and realistic.

12. As a user, I need to use already saved weather data in the simulation to not be dependent on the availability of the weather service.

13. As a user, I need to generate a forecast for energy generation and demand using the simulation in order to make informed decisions.

14. As a developer, I want to add more supplier modules than just wind turbines and photovoltaic panels to the simulation to improve the smart energy system in the future with further technology due to adding more kinds of suppliers.

15. As a developer, I want to add more consumer modules to the simulation to be able to add more kinds of consumers to the simulation.

16. As a user, I want to be able to create a smart energy system which is independent to a main power grid to simulate a reliable smart grid.

17. As a user, I want to get a visual notification if the supply of energy is smaller than the demand of energy to know when more energy supplier are needed.

18. As a user, I want to model a rechargeable battery so I can store the energy for later usage, if the supply is greater than the demand.

19. As a user, I need the battery to be able to discharge energy if the supply is lower than the demand in order to make my stored energy usable and keep the demand satisfied.

20. As a user, I need the smart grid to be able to manage peaks in the demand in order to smooth the impact on the grid and reduce the likelihood of power outages.

21. As a user, I need that the demand side consumers feature different load scenarios like home users and commercial users (constant load, occasionally peak loads) in order to make the simulation accurate for real life applications.

22. As a user, I want be able to use the system with my webbrowser so that I can use different platforms to view it and have easy access to the simulated data.

Smart Energy Systems   *Sandro Speth*
Wintersemester 18/19   *Markus Zilch*
Report   *Dominik Wagner*

23. As a user, I want to be able to see the energy supply of each individual supply component in order to be able to assess the efficiency of the supplier.

24. As a user, I want to be able to see the energy demand of individual components for efficiency assessment and informed decision making.

25. As a user, I want to see a summary of energy supply and demand for all components in order to easily assess the current situation.

26. As a user, I want to adjust the demand by postponing the use of devices during peak hours in order to prevent a complete outage of the grid or to react to one-time-only scenarios.

27. As a user, I want the simulation to adapt the demand based on the price per kWh in order to minimize costs of my energy demands and maximize profits of my energy supply.

2.4 Architecture

This section describes our system architecture. An overview over the architecture is shown in figure 2.
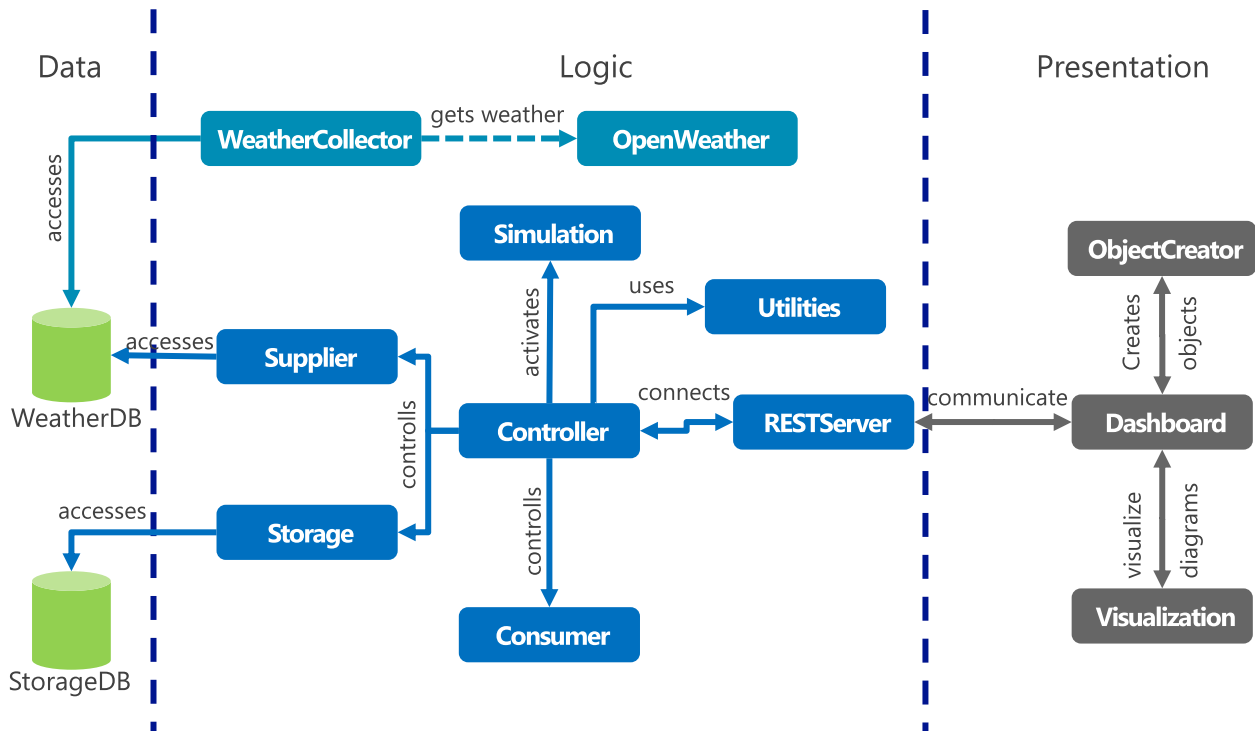

Figure 2: Smart energy system architecture

The `WeatherCollector` component is responsible for gathering actual weather data from *OpenWeather* API. This weather data is stored into a database for further use of the supply and demand components of the system.

In order for the whole system to be reliable and responsive we separated the `WeatherCollector` component from the rest of the system. It speaks only to the weather database where it get the locations for which it should collect weather data. The collected data gets written into the

database and stored. Even in the case of a failure of the weather component or OpenWeather the already collected data is still accessible and therefore the system can work with it, making the system independent of the weather component. In the case of a new registration of a supplier component during a weather component failure, the database may return default values for the new component in a way the running system is not being halted by missing data. These measures should make the system reliable and and responsive in that context.

The three-layered architecture was already planned for in the first conceptions of the project, therefore no extra steps had to be taken. We divided the project in subcomponents and put them in the respective part of the architecture.

The first layer is the data layer which only consists of data-providing hardware and databases. Since the current project does not include data-providing hardware (as far as the current conception goes) only databases are left in this layer.

The second layer consists of the functional components of the project. All suppliers, storages, consumers and utility components are part of this layer, since they mostly take data from the databases and compute their respective power in- and output based on those values and give them to the simulation component. The simulation component is part of this layer, and takes the data the other component in order to model the different interactions of the smart grid. The workflow is controlled by the controller component and connects all components together. The last component in this layer is the REST component which makes the computed data from the simulation component available to the frontend layer.

The frontend layer is the third and last layer in our project. It handles the interaction with the user and makes it possible for the user to add and remove components to the smart grid simulation as he sees fit.

3 Implementation

The current implementation consists of two independent parts. The weather collector component which is described in section 3.1 is developed with NodeJS express. The main component which handles all the simulation and calculations is written in Java and uses the Spring Boot framework. The theoretical foundation of this component is described in section 3.2 and the following. Both components are linked through a shared database.

3.1 Gather weather design

3.1.1 Weather service

For our smart energy system, we need a weather service which can provide all needed data on an hourly basis. There is one weather service which fulfills our needs more than appropriate. The weather service `weatherbit.io` has different kinds of endpoints on its API. In our case we use two of them, the current weather data API and the 48 hours weather forecast API.

Before we take a look at the endpoints provided to the user and their responses, we want to present the data collectable on weatherbit.io. An example response of the current weather API is shown in listing 1. The property `data` of the JSON object contains an array of weather data points. For current weather API there is only one object in the array. But for the forecast there are up to 48 objects, one for every hour of the forecast.

Listing 1: Example response of weatherbit.io for $lat = 31.23$ and $lon = 121.47$

```
1   {
2       "data": [
3           {
4               "wind_cdir": "SSE",
5               "rh": 100,
6               "pod": "n",
7               "lon": 121.47,
8               "pres": 1012.2,
9               "timezone": "Asia/Shanghai",
10              "ob_time": "2018-11-07 20:35",
11              "country_code": "CN",
12              "clouds": 0,
13              "vis": 10,
14              "solar_rad": 0,
15              "state_code": "23",
16              "wind_spd": 0.89,
17              "lat": 31.23,
18              "wind_cdir_full": "south-southeast",
19              "slp": 1013.2,
20              "datetime": "2018-11-07:20",
21              "ts": 1541622900,
22              "station": "E7205",
23              "h_angle": -90,
24              "dewpt": 13.9,
25              "uv": 0,
26              "dni": 0,
27              "wind_dir": 156,
28              "elev_angle": -29.2797,
29              "ghi": 0,
30              "dhi": 0,
31              "precip": null,
32              "city_name": "Shanghai",
33              "weather": {
34                  "icon": "c01n",
35                  "code": "800",
36                  "description": "Clear sky"
37              },
38              "sunset": "09:00",
39              "temp": 13.9,
40              "sunrise": "22:14",
41              "app_temp": 13.9
42          }
43      ],
44      "count": 1
45  }
```

From all those data we chose only to store `lat` for latitute, `lon` for longitute, `pres` for air pressure, `rh` for relative humidity, `ghi` for global horizontal solar irradiance, `wind_spd` for wind speed, `temp` for the temperature, and the timestamp in UTC format.

The API for the current weather data returns current weather data of over 45.000 stations. Every API request will return the nearest, and most recent observation. There are several endpoints available in the API. For example, a user can get current weather observation by lat/lon as we do or by city name. All endpoints differ only on their required query parameters. They define whether to get the observation by lat/lon, city name or any other possible way. The basepath for all endpoints is `https://api.weatherbit.io/v2.0/current` and the supported method is `GET`. To get the data in the metric format you have to add an additional query parameter `unit=m`. Every request must provide an API key in query parameters.[16] This API key can be requested by creating an account on weatherbit.io. After creating an account a user has to choose a pricing plan. For our system the free plan is totally sufficient.

The API for the weather forecast data returns the weather forecast for up to 48 hours on an hourly basis. Nevertheless, we only request it for 24 hours. While the basepath is now `https://api.weatherbit.io/v2.0/forecast/hourly`, the provided query parameter are the same with an additional query parameter `hours` which must get a value between 0 and 48 to specify the size of the forecast.

3.1.2 Our WeatherCollector service

To gather weather data for different locations and store it into a database we build a service, which is under MIT licence available on GitHub[1]. The service runs a NodeJS express server with two endpoints to create or delete `WeatherCollector` objects. A `WeatherCollector` object gathers every hour current weather data for a given location as well as 24 hours weather forecast on an hourly basis and stores the data in a database. While gathering new forecast data, the old deprecated data is getting replaced, so there is always the latest 24 hour forecast data. Multiple WeatherCollector objects are possible to collect weather data for different locations at the same time.

There are several dependencies you have to install before running the server. Since the code runs on NodeJS version 8 or higher with a current npm, you need to have this installed first. First, run `npm install` to install needed dependency packages. You can look up the installed libraries and their versions in `package.json` file. After installing the packages, you need to create the database. Therefore, run `node DBCreator.js` to create the database file and required SQL tables. Now you are nearly ready to run the server.

You can test the functionality without running the complete server. You just need to run `WeatherCollectorTest.js` to play around with some locations. On the first run of the module you need to provide it with your weatherbit.io API key, latitute and longitute of the location: `node WeatherCollectorTest.js "<API Key>" <lat> <lon> "<mode>"`. Keep in mind that you already need the API key requested from weatherbit.io. There are 5 modes available to test:

- `collect`: Collects a current weather data of the given location and stores it into the database

- `get`: Gets the latest stored current weather data of the given location

---

[1]`https://github.com/smart-energy-system/WeatherCollector`

- `collect forecast`: Collects a 24 hour weather forecast on hourly basis for a given location and stores it into the database

- `get forecast`: Gets the latest version of the forecast weather data of the given location

- `start`: Starts a WeatherCollector object on hourly picking new weather data (current and forecast) and stores the data into the database.

If you did run the complete run command once, a test config file was created. So unless you want to change your API key or the location you can shorten the run command a bit, using only `node WeatherCollectorTest.js "<mode>"`.

You can simply run the server with `npm start <API key>`. This will create a config file with your API key, so later you can start the server just running `npm start` command.

Since the service is a REST service, there are two HTTP REST endpoints available on the server, one for creating WeatherCollector objects and one for deleting them. To create a new WeatherCollector object, you can send an `POST` request against `<basepath>/weathercollectors` providing the body shown in listing 2 as content-type `application/json`.

Listing 2: Body of `POST` request endpoint

```
1  {
2      "lat":<lat>,
3      "lon": <lon>
4  }
```

If `lat` and `lon` are correct and the object is created successfully you will get status 201 and a JSON object as shown in listing 3 in return.

Listing 3: Body of the response of the `POST` request

```
1  {
2      "lat": <lat>,
3      "lon":<lon>,
4      "id":<id>
5  }
```

The `id` is the ID of the object created. You will need it to delete the object later, so better save it anywhere. After creating the object, the object is requesting the weather data (current and forecast) every hour. If one of the request body properties is not defined you will get a status 400 in return.

To delete a WeatherCollector object of given ID, you can send an `DELETE` request against `<basepath>/weathercollectors` providing the in body the data shown in listing 4 as content-type `application/json`.

Listing 4: Body of `DELETE` request endpoint

```
1  {
2      "id":<id>
3  }
```

If `id` is set and there is a WeatherCollector object with such an ID, it will be deleted and therefore stops gathering data. The return will be status 200. If id is not set or there is no WeatherCollector object with such an ID, the return will be status 400.

3.2 Windturbine

3.2.1 Swept area function

The swept area of a wind turbine is dependent on the radius of the rotary blades. Combined with the constant Pi the area swept can be computed by the equation[2]: $A = r^2 * \Pi$

3.2.2 Vapor pressure

To compute the actual vapor pressure in the air the relative humidity has to be given. The relative humidity is computable with the equation $H = P_{av}/P_s$ with P_{av} being the actual vapor pressure and P_s being the saturated vapor pressure at a given temperature. This equation can be restructured to get the actual vapor pressure: $P_{av} = H * P_s$. The relative humidity is a parameter for the function, which is filled with data from the before mentioned weather data collection.

To be able to now compute the actual vapor pressure at a given temperature we still need the saturated vapor pressure. For this we can use the Herman Wobus equation (E being the vapor pressure):
$E = e/p^8$ with
e = 6.1078 and
$p = c_0 + T * (c_1 + T * (c_2 + T * (c_3 + T * (c_4 + T * (c_5 + T * (c_6 + T * (c_7 + T * c_8 + T * c_9)))))))$
with c_0 to c_9 being constants and T being the temperature in degrees Celsius.[1, 9]

3.2.3 Density of moist air

In order to compute the density of moist air, we have to have a look at how it is compounded. Moist air density is a mixture of dry air and water vapor. The physical equation for this is:
$D_m = (P_d/(R_d * T_k)) + (P_v/(R_v * T_k))$
D_m is the density of moist air, P_d is the pressure of dry air at the specified temperature, R_d is the gas constant for dry air, P_v is the pressure of water vapor at the specified temperature, R_v is the gas constant for water vapor and T_k is the given temperature in degrees Kelvin. With the previously implemented methods this equation system is able to compute the air density with only the temperature and relative humidity given.[1, 13]

3.2.4 Wind turbine model

The wind turbine model contains some variables which can be set by the user. The function for computing the generated energy is derived from the power coefficient of the turbine (basically the efficiency), the size of the turbine (shows in area swept), the density of the air in the area and of course the weather conditions which apply at the moment given. The general equation for this setup is: $P_{avail} = (1/2) * p * A * v^3 * C$ where p is the air pressure, A the area swept, v the windspeed and C the power coefficient. [7]

3.3 Photovoltaic

3.3.1 Temperature loss

The function for the temperature loss is giving a linear function for the percentage loss of energy depending on the degrees over 25 degree Celsius. We made the assumption that the efficiency does not go over 100 % even for temperatures below 25 degree Celsius.The resulting equation is as follows: $L_t = max((T - 25) * 0.005, 0)$ with the result being the percentage of energy lost due to temperature as a point number.

3.3.2 Performance ratio

The performance ratio is sum of all losses. With the previous computed losses for temperature and the constant loss L_0 of 0.14 we get the equation $R = (L_0 + L_t) = 1.0 - (0.14 + L_t)$. To use the result in our calculation we have to subtract the result form 1.

3.3.3 Solar irradiance

The computation of the solar radiation incident and therefore the effective solar radiation on the surface of the solar panel can be computed with the equation $(SP_{horizontal} * sin(\alpha + \beta))/sin(\alpha)$. α is the effective tilt of solar radiation on a specified latitude and set as $90 - latitude$ of the position from the solar panel $+ \delta$. δ is the tilt of the solar radiation in regard of the day of the year d and can be computed with the following equation: $23.45 * sin((360/365) * (284 + d))$. ($SP_{horizontal}$ is the horizontal solar radiation. Our weather service provides us with this information. The tilt of the solar panel itself is used as β. [11]

3.3.4 Photovoltaic model

The formula for computing the generated energy as given in the slides is: $E = A * r * PR * S$ A is the area of the solar panel and is given by the user in m^2. Same goes for the solar panel yield r which is given by the user via the maximum power yield as well. The performance ratio PR is computed in on of the previous steps and is a percentage. S is the solar radiation incident and is computed in regards of solar radiation and other factors taken from the database. The measurement unit for S is W/m^2. The resulting unit for produced energy is therefore W.

3.3.5 Battery model

The first function for returning the current state of the battery is a simple return of the currently stored energy. The formula for charging and discharging is more complex and takes several parameters. The parameters are charging and discharging rate as well as charging efficiency. The function returns the actual change of energy stored in the battery. The returned value is positive if the battery is charged and negative if the battery is discharged.

3.3.6 Demand profile

Our demand profile is based on data from `https://open-power-system-data.org/`. We use the „Household Data" package from 2017-11-10. The calculations are based on the 60 minutes interval data set. We use two different profiles, one is based on the feed $DE\_KN\_residential2$ $\_grid\_import$ and the other is based on $DE\_KN\_industrial3\_grid\_import$. The raw data

from Open Power System Data always report the total amount of energy used so far in kWh. Based on this we needed to calculate the difference between the data points to the get change of the total value per hour. After that, we calculated the average value for each hour of the day for all data points. The consumer in profile 1 used in total about 4500 kWh in the time between the 15. April 2015 and the 1. February 2017. This equals roughly a total of 2500 kWh per year. The second consumer used in total 695000 kWh in the time between 11. February 2016 and 09. February 2017. Our electricity provider estimates a small one or two people household for the first consumer based on the total usage per year [5]. Based on this we amused a floor area size of 50 m^2. Figure 3 and figure 4 visualize the demand profiles. The user does not only provide an area but also an average daily occupancy. We assumed that the latter value is the fraction of the time in which the building is in use. Based on this we calculate the final energy demand D for each hour for a building with this formula $D = profileDemandPerSquareMeter_{hourOfTheDay} * floorAreaSize * averageDailyOccupancy$.
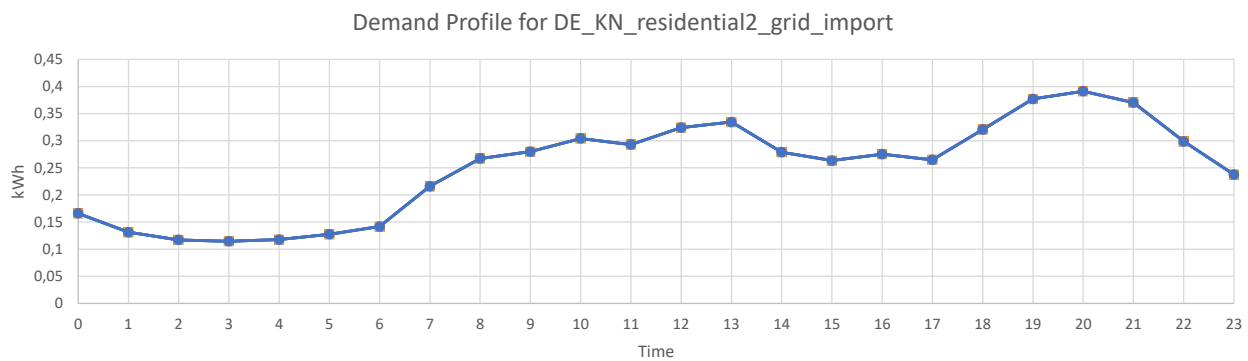

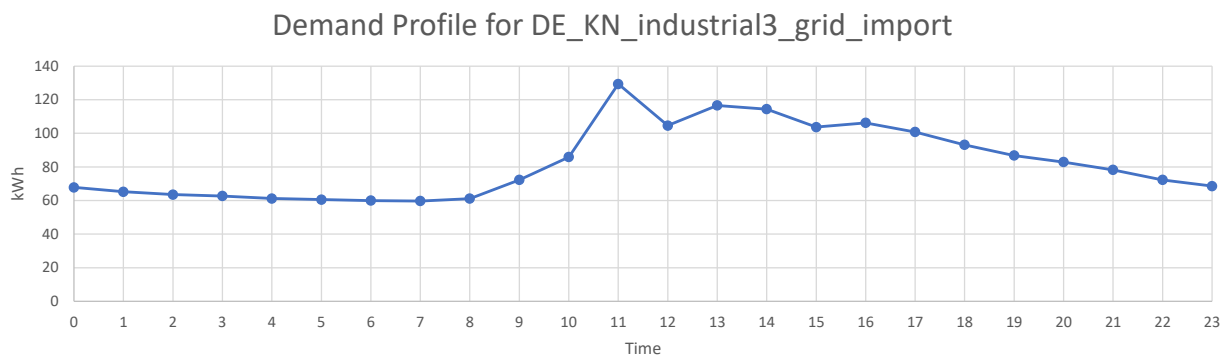Figure 3: Demand profile 1: Home building

Figure 4: Demand profile 2: Office building

4 Demand Response and coordination

4.1 Balance within the microgrid

To guarantee the balance in the microgrid as defined in the slides we have to formulate a mathematical equation to use as a constraint. We came up with the following equation:

$$\forall t \in T : \sum_{s \in S} P_s(t) + \sum_{b \in B} P_b(t) + P_g(t) = \sum_{h \in H} (P_h(t) + PposShift_h(t) - PnegShift_h(t))$$
$$+ \sum_{c \in C} (P_c(t) + PposShift_c(t) - PnegShift_c(t)) - \sum_{b \in B} Pd_b(t) \tag{1}$$

As one can see, for all timesteps t in the set of all timesteps T the equation has to hold true. This is the constraint for the whole microgrid. The left side of the equation represents the power supplied by the supplier ($P_s(t)$), the batteries ($P_b(t)$) and the import or export from the main grid ($P_g(t)$), the right side represents the consumers, which contains homes ($P_h(t)$) and commercial buildings ($P_c(t)$), the batteries ($Pd_b(t)$) while charging. On the right side, there are also the demand shift values for homes and commercial buildings.

If the balance between the consumers and the suppliers is disturbed, severe consequences can happen. Any imbalance results in a change in the frequency. This change may forces more suppliers to stop production because most suppliers can only operate in a narrow frequency range. As a result of this the imbalance ingresses even more. If the change in the frequency exceeds a certain threshold suppliers and consumers still connected to the grid could be damaged.

The supply side of the equation itself is compartmentalized into different factors which are explained in the following.

Wind turbines and solar panels: On the supply side we have the sum the power in timestep t of all suppliers of the system ($(P_s(t))$), regardless of wind turbines or solar panels. We get those values from our model given data for the timestep and weather conditions. Our software calculates demand forecast and supply forecast based on models of the supplier and consumers. Additionally, the location of the installations are provided by the user and a weather forecast is provided by a weather service. For example, for the wind turbines, the Herman Wobus equation is used to calculate the saturated vapor pressure [9]. This equation is based on empirical data and helps us to incorporate the weather forecast in the wind supply forecast. Because all these calculations happen before the optimization step, the solver does not need to know anything about the specifics of the supplier or consumer. As a result of this we don't need any constrains for the consumer or suppliers. This supports reuse-ability of the solver and the models for it. The forecast data is provided as an array of data which contains a value for each timestep. The values represent the supply or demand and are measured in $Watts$.

Battery supply: The second part is the power provided by the sum of all batteries at a given timestep t. Each battery has a fill level for each time step. It holds the information how much charge is stored in the battery at each step. This results in the following constraints. Additionally, there is a limit on the charge and the discharge rate. This results in the following constraint:

$$\forall b \in B, \forall t \in T : P_b(t) < maxDischargeRate_b \tag{2}$$

$$\forall b \in B, \forall t \in T : fillLevel_b(t+1) = fillLevel_b(t) - Pd_b(t) \tag{3}$$

Import and export to the main grid: The connection to the main grid is resolved in the third part of the supply side of the equation. If we have more demand than supply, the solver will either use the energy stored in the battery or import energy from the main grid. While the demand is satisfied by the suppliers and the battery, no further energy import is necessary. But if the batteries are empty and the demand exceeds the supply more energy has to be imported from the main grid. The import of energy form the main grid costs money. If the supply exceeds the demand it is possible to charge the batteries or to export the energy to the main grid. Exporting energy to the main grid provides revenue.

For the demand side, we chose a similar approach.

Demand from homes and commercial buildings: The first one is the sum over all homes of a a given timestep t with related demand shift of every single home to calculate the total demand of all homes. The second one is the sum over all commecial buildings of a given timestep t with related demand shift of every single commecial building to calculate the total demand off all commecial buildings. Comparable to the supply forecast, the demand forecast is generated before the optimization step. It is based on the demand profiles described in section 3.3.6. The demand of the homes is provided as an array of constants to the solver and measured in $Watts$. Because the supply and the demand are provided as constants to the optimization component, we did not formulate any constraints for the supplier or the consumers. For the demand shift parts of the demand entities we have a positive shift and a negativ shift. The negative demand shift tells us how much demand is shifted away on this timestep. The positive demand shift tells us the opposite, how much demand is shifted to this timestep. So the negative demand shift will decrease the demand while the positive demand shift will increase the demand. If the negative demand shift of a home or commecial building is greater than 0, the positive demand shift has to stay 0 and vice versa. Additionally, overall the demand has to stay the same. This results in the following constraints:

$$\forall l \in H \cup C, \forall t \in T : (PposShift_l(t) > 0 \veebar PnegShift_l(t) > 0)$$
$$\veebar(PposShift_l(t) = 0 \wedge PnegShift_l(t) = 0) \tag{4}$$

For all consumers this constraints has to hold true:

$$\forall t \in T : \sum_{t \in T} PposShift(t) = \sum_{t \in T} PnegShift(t) \tag{5}$$

We need to make an upper bound for the negative shifting which a user can provide for the house or commercial building demand, e.g. 30%. Additionally, shifts can only happen in blocks. It is only possible to shift the max value of the shift or noting. The constraint for the house shifting is shown in Eq. (6). The constraint for the commercial building is the same.

$$\forall l \in H \cup C, \forall t \in T : \left(PnegShift_l(t) = \frac{P_l(t) * n}{100} \right) \veebar (PnegShift_l(t) = 0) \tag{6}$$

For this constraint n is provided by the user.

There are two possibilities for shifting demand which can be chosen by a user. The first one is used if the user wants to save time or has limited RAM. This variant allows only one shift per

supplier. For example form 30% of the load form 9 o'clock to 10 o'clock. This variant uses the constraint form Eq. (7). The second variant allows to shift form every hour to every other hour. This results depending on the input and the number of time steps in a very long run-time and a high RAM consumption.

$$\forall l \in H \cup C : numberOfShifts_h \leq 1 \tag{7}$$

Demand of batteries: The third part calculates the demand of all batteries at a given timestep t. If we have more supply than demand, the solver can either export the unnecessarily produced energy to the main grid or store it in a battery for later use. The storage of energy in the battery has a small loss of energy.

$$\forall b \in B, \forall t \in T : Pd_b(t) < maxChargeRate_b \tag{8}$$

$$\forall b \in B, \forall t \in T : fillLevel_b(t+1) = fillLevel_b(t) - \left(\frac{Pd_b(t) * chargeEfficiency_b}{100} \right) \tag{9}$$

4.1.1 Additional Constraints

There are additional constrains which have to hold true in our simulation.

We don't want the batteries charge and discharge at the same time. So we need a constraint to prohibit this kind of behaviour as shown in Eq. (10)

$$\forall b \in B, \forall t \in T : (P_b(t) = 0 \wedge Pd_b(t) = 0) \veebar (P_b(t) > 0 \veebar Pd_b(t) > 0) \tag{10}$$

There is also such an constraint for the shifting part to prohibit positive and negative shifting at the same time.

4.2 Objective of the optimization problem

We import or export energy from the main grid with $P_g(t)$. If $P_g(t) < 0$ than energy will be exported since the overall supply gets less than only with the batteries and renewable supplier. If $P_g(t) > 0$ than energy will be imported since the overall supply gets more than only with the batteries and renewable supplier. We can use this to compute our profit or loss since we want the optimizer to maximize the profit. To compute profit and loss we declared two variables $isEnergyImport$ and $isEnergyExport$ as shown in Eq. (11).

$$\begin{aligned} P_g(t) < 0 &\Rightarrow isEnergyImport \\ P_g(t) > 0 &\Rightarrow isEnergyExport \end{aligned} \tag{11}$$

If $isEnergyImport$ is true, the optimizer will use the absolut value of P_g as import. In this case the export will be zero for timestep t. If $isEnergyExport$ is true, it's the other way round.

To maximize the profit in a specified interval of timesteps $t \in T$ between the timesteps $a \in T$ and $b \in T$ we propose the following function:

$$\max_{p \in P} \left(\sum_{t=a}^{b} ((E(t) * price(t)) - (I(t) * cost(t))) \right) \tag{12}$$

P is th set of all possible decision configurations. $E(t)$ contains the energy export (*isEnergyExport* is true) of timestep t and $I(t)$ (*isEnergyImport* is true) contains the energy import of timestep t. These variable are depending on the decisions made by the optimizer on the equation for the balance in the microgrid of the exercises before. All configurations from P have to satisfy all the constraints. The formula consists of two main parts. The minuend in the sum calculates the profit for exporting energy to the main grid for these timestep. For this calculation, the total energy export $E(t)$ for this timestep is multiplied by the price for the energy. The subtrahend in the sum expresses the cost for importing energy from the main grid. It consists of the total imported energy $I(t)$ multiplied by the cost for importing energy at this timestep. There are different possibilities to improve the total profit. For example, one could sell energy at a high price. Another possibility is to store energy if the price for selling is low. For our simulation we use timesteps with the width of one hour, so the values for power and energy are the same, but $E(t)$ and $I(t)$ are measured in Wh while the unit of all values of the previous equation is $Watt$. The price for exporting energy and the cost for importing energy are provided as an array of constants.

4.3 Implementation

To implement the constraints we use the library choco solver. [2].

4.4 Nomenclature Table

Symbol	Description	Unit	Equation
T	The set of all timesteps. The step width is 1h and usually 24 steps are used.	Hour	Eqs. (1) and (12)
t	A single step from the set T.	Hour	Eqs. (1) and (12)
S	The set of all suppliers (Different wind turbines, solar panels...).	Watt	Eq. (1)
s	A single supplier from the set S.		Eq. (1)
$P_s(t)$	The energy supply from a single supplier for the specific timestep t.	Watt	Eq. (1)
B	The set of all batteries.		Eq. (1)
b	A single battery.		Eq. (1)
$P_s(t)$	The energy supply from a single battery for the specific timestep t.	Watt	Eq. (1)
$P_g(t)$	The energy supply from the main grid for the specific timestep t.	Watt	Eq. (1)
D	The set of all consumers (Homes, commercial buildings)		Eq. (1)
d	A single consumer from the set D		Eq. (1)

---

[2]http://www.choco-solver.org/

$P_d(t)$	The energy demand for a single consumer at a specific timestep t.	Watt	Eq. (1)
$P_b(t)$	The energy demand for a single battery at a specific timestep t.	Watt	Eq. (1)
$Pd_g(t)$	The energy exported by the microgrid to the main grid at a specific timestep t.	Watt	Eq. (1)
P	The set of all possible configurations		Eq. (12)
p	A single configuration		Eq. (12)
a	A single step form the set T. It marks the first timestep of the optimization	Hour	Eq. (12)
b	A single step form the set T. It marks the last timestep of the optimization	Hour	Eq. (12)
E(t)	The total energy exported to the main grid for a specific timestep t.	Wh	Eq. (12)
I(t)	The total energy imported from the main grid for a specific timestep t.	Wh	Eq. (12)
price(t)	The function provides the price for energy per Wh which is sold to the main grid for a specific timestep t.	Cent per Wh	Eq. (12)
cost(t)	The function provides the cost for energy per Wh which is imported from the main grid for a specific timestep t.	Cent per Wh	Eq. (12)

Table 1: Describes every Symbol

5 Conclusions

The power girds of the future will be different from the existing ones. More information will be used to make better decisions, more renewable energy sources will be integrated and more automation will happen. Also, microgrids will be a part of the energy concepts of the future. The simulation system presented in this report could help to gain a better understanding about different aspects of microgirds. The first section provided a motivation and introduction to microgirds. The next section contains basic information which are necessary to understand the problem, the functional requirements and an architecture description. The architecture is split in three layers. The webfrontend, the logic layer which contains the simulation and the database layer. It also features a reliable approach to integrate an external weather component. Section 3 contains a description of our implementation.

In the future more custom modules could be developed, to integrate even more suppliers and consumers.

References

[1] EMD Internation A/S. *WindPRO / ENERGY Modelling of the Variation of Air Density with Altitude through Pressure, Humidity and Temperature*. 2005. URL: `http://www.emd.dk/files/windpro/WindPRO_AirDensity.pdf`.

[2] Petr Beckmann. *A History of Pi*. 1976.

[3] Gerard Borvon. *History of the electrical units*. 2011. URL: `http://seaus.free.fr/spip.php?article964`.

[4] Schweizerische Eidgenossenschaft. *Bundesgesetz über das Messwesen*. 2018. URL: `https://www.admin.ch/opc/de/classified-compilation/20101915/`.

[5] enbw. *Gut zu wissen: durchschnittlicher Stromverbrauch im Haushalt*. 2018. URL: `https://www.enbw.com/strom/stromverbrauch-senken`.

[6] enertiv. *What is Peak Demand?* 2018. URL: `https://www.enertiv.com/resources/what-is-peak-demand`.

[7] The Royal Academy of Engineering. *Wind Turbine Power Calculations*. 2018. URL: `https://www.raeng.org.uk/publications/other/23-wind-turbine`.

[8] EU-Richtlinie. *EU-Richtlinie 80/181/EWG in den Staaten der EU bzw. dem Bundesgesetz über das Messwesen in der Schweiz*. 2018. URL: `https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX:32009L0003`.

[9] NOAA Environmental Research Laboratories. *PROFS Programm*. 1981. URL: `https://icoads.noaa.gov/software/other/profs`.

[10] Universität Stuttgart Otto Eggenberger. *Leistung des elektrischen Stroms*. 2018. URL: `http://www.iris.uni-stuttgart.de/lehre/eggenberger/eti/index.html`.

[11] PVEducation. *Solar radiation on a tilted surface*. 2018. URL: `https://pveducation.org/pvcdrom/properties-of-sunlight/solar-radiation-on-a-tilted-surface`.

[12] Applied Technology Institute Robert A. Nelson. *The International System of Units*. 2018. URL: `https://www.aticourses.com/international_system_units.htm`.

[13] R Shelquist. *Equations - air density and density altitude*. 2009.

[14] Das Internationale Einheitensystem (SI). "Deutsche Übersetzung der BIPM-Broschüre "Le Système international d'unités/The International System of Units (8e édition", 2006)". In: 117 (2006). URL: `https://www.ptb.de/cms/fileadmin/internet/Themenrundgaenge/ImWeltweitenNetzDerMetrologie/si.pdf`.

[15] Stony Brook University. *Consumption vs. Demand*. 2018. URL: `https://www.stonybrook.edu/commcms/energy/facts/demand`.

[16] weatherbit.io. *The High Performance Weather API for all of you Weather data needs*. 2018. URL: `https://www.weatherbit.io`.

All links were checked last on December 13, 2018.