BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

MASTER'S THESIS

Brno, 2020 Ing. Vít Hertl

BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

MULTIBAND SOLAR SENSORS DATA ACQUISITION AND CLOUD PROCESSING

SBĚR A CLOUDOVÉ VYHODNOCENÍ DAT Z VÍCEPÁSMOVÝCH SOLÁRNÍCH SENZORŮ

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR AUTOR PRÁCE	Ing. Vít Hertl
SUPERVISOR VEDOUCÍ PRÁCE	Ing. Radovan Holek, CSc.

BRNO 2020

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Master's Thesis

Master's study field **Cybernetics, Control and Measurements**

Department of Control and Instrumentation

Student: Ing. Vít Hertl ***ID:*** 160774

Year of study: 2 ***Academic year:*** 2019/20

TITLE OF THESIS:

Multiband solar sensors data acquisition and cloud processing

INSTRUCTION:

1. Do the literature research about solar sensors connection possibilities and pair measured data with information about energy generated by paired PV panels. Realize the optimal method such as Wi-Fi, Bluetooth, LoRa etc.
2. Experimentally prove functionality of used components and get them ready for test operation.
3. Suggest and describe own application solution for receiving and storing data from multiple sensors and paired PV panels generated power inverters with dedicated connection to cloud.
4. Realize own solution by developing application server back-end and front-end for receiving, storing and displaying data.
5. Validate the solution by evaluating real data.

RECOMMENDED LITERATURE:

RÁČEK, J. Strukturovaná analýza systémů. 1.vyd. Brno masarykova univerzita, 2006 103 s ISBN 80-2010-4190-0

Date of project specification: 3.2.2020 ***Deadline for submission:*** 1.6.2020

Supervisor: Ing. Radovan Holek, CSc.

doc. Ing. Václav Jirsík, CSc.
Subject Council chairman

ABSTRACT

This master's thesis builds upon the foundations laid in the same-titled semestral thesis [1]. At the beginning the fundamental properties of sunlight that are needed for understanding of the performance ratio computation are briefly mentioned. After that the solar sensors developed in ReRa Solutions which provided the solar irradiance measured data are described. In the following literary research part the alternatives to unreliable sensor's Wi-Fi connection are compared and the most suitable connection technology is concluded to be LoRa. In the practical part the design and development process of a single page application solution is described in detail. The application allows the sensors to log measured data into the database and is able to retrieve and display them back to the user in an intelligible form. The application benefits from modern programming languages and frameworks (e.g. Kotlin, Spring, TypeScript, React, Material-UI). The database model was designed with respect to real-world scenario introducing the monitored area concept allowing extensive configuration possibilities. Then the model was transferred into the database in the form of programmable entities. Further, the client-server communication via REST API allowing role-based authentication was implemented. The application GUI allows to configure the environment according to user needs and renders interactive graphs with sensor measured data. The application was deployed in Google Cloud with a separate database.

KEYWORDS

Photovoltaics, performance ratio, solar sensors, Kotlin, React, TypeScript, Spring, LoRa, cloud application.

ABSTRAKT

Tato diplomová práce staví na základech položených v rámci semestrální práce se stejným názvem [1]. Na začátku jsou nejprve uvedeny základní vlastnosti slunečního záření nutné k pochopení výpočtu tzv. performance ratio. Dále jsou popsány solární senzory vyvinuty v ReRa Solutions, které byly zdrojem dat. V následné literární rešerši jsou zkoumány alternativy k nespolehlivému Wi-Fi připojení senzoru a za nejvýhodnější řešení je považována LoRa. V praktické části je detailně popsán proces návrhu a vývoje single page aplikace. Tato aplikace umožňuje jak ukládání dat ze senzorů do databáze, tak jejich opětovné čtení a zobrazení zpět uživateli ve srozumitelné podobě. Aplikace těží z využití moderních programovacích jazyků a frameworků (např. Kotlin, Spring, TypeScript, React, Material-UI). Databázový model, který byl navržen na základně skutečného využití, představuje koncept tzv. monitorované oblasti, což přináší široké konfigurační možnosti. Poté byl model přenesen do databáze ve formě programovatelných entit. Komunikace mezi klientem a serverem podporující autentifikaci na základě uživatelských rolí byla implementována přes REST API. Přes grafické rozhraní aplikace je možné konfigurovat prostředí podle uživatelských požadavků a zobrazit interaktivní grafy obsahující senzory naměřená data. Aplikace byla nasazena v Google Cloudu s oddělenou databází.

KLÍČOVÁ SLOVA

Fotovoltaika, performance ratio, solární senzory, Kotlin, React, TypeScript, Spring, LoRa, cloudová aplikace.

HERTL, Vít. *Multiband solar sensors data acquisition and cloud processing*. Brno, 2020, 99 p. Master's Thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation. Advised by Ing. Radovan Holek, CSc.

Typeset by the thesis package, version 3.05; http://latex.feec.vutbr.cz

ROZŠÍŘENÝ ABSTRAKT

Na začátku této diplomové práce jsou nejprve uvedeny základní vlastnosti slunečního záření potřebné k pochopení výpočtu energie generované fotovoltaickou elektrárnou a tzv. performance ratio, pomocí kterého je možné monitorovat její účinnost. Právě výpočet performance ratio z naměřených dat intenzity ozařování je jedna z funkcí vyvíjené aplikace.

Zařízení použitá k měření intenzity ozařování s komerčním názvem My Solar Scan (MSS) byla vyvinuta firmou ReRa Solutions, nasazena do provozu a ve spolupráci s technikem z firmy nastavena tak, aby posílala data do vyvíjené aplikace. Tato práce přináší nejen popis architektury MSS (senzoru a připojovací krabice), ale snaží se také přijít s alternativou připojení MSS k internetu, protože současná varianta s Wi-Fi modulem není příliš spolehlivá. Alternativní možnosti připojení byly zkoumány v rámci literární rešerše a zahrnovaly různé možnosti např. Wi-Fi, Bluetooth, Zigbee, Sigfox, LoRa nebo NB-IoT. Jejich výhody a nevýhody byly srovnány a v souvislosti s plánovaným využitím MSS a jako nejvýhodnější technologie byla zvolena LoRa především díky relativně nízké provozní a pořizovací ceně a celkem dobrému pokrytí.

Těžiště této diplomové práce leží ve vývoji full-stack aplikace, která je schopná sbírat data z připojených senzorů přes API, ukládat tato data do databáze, počítat hodnoty veličin charakterizujících danou fotovoltaickou elektrárnu a zobrazovat tyto hodnoty zpět uživateli ve formě interaktivních grafů. Popsané funkcionality je docíleno využitím moderních frameworků a programovacích jazyků jako jsou Kotlin a Spring na straně serveru a TypeScript s Reactem na straně klienta.

Základní funkcionalita aplikace byla definována již vytvořením datového modelu. Při jeho vývoji byla věnována pozornost tomu, aby odpovídal reálným možnostem využití. Z tohoto důvodu vznikl koncept tzv. monitorované oblasti, která popisuje konkrétní plochu měřené fotovoltaické elektrárny a umožnuje k této ploše připojit senzor a invertor. Zatímco data ze senzoru byla použita k výpočtu teoretické energie, kterou by daná fotovoltaická plocha měla generovat, z invertoru lze získat informaci o reálně vyrobené energii. Právě srovnáním těchto dvou veličin je možné spočítat dříve zmíněné performance ratio. Aby bylo možné k datovému modelu přistupovat z programového kódu, bylo jej nejprve potřeba definovat pomocí programovatelných entit. Převod datového modelu do kódu je v práci popsán. Následně byl implementován REST kontrolér, který umožňuje přistupovat k aplikaci vzdáleně. REST metody byly následně zabezpečeny s využitím Spring Security a JSON Web Tokenů.

Aplikační grafické rozhraní bylo napsáno s využitím Reactu a Material-UI komponent ve skriptovacím jazyce TypeScript. Možnosti GUI zahrnují nastavení prostředí podle typu konfigurace měřené fotovoltaické elektrárny a zobrazení vypočtených grafů pro předem vybraný senzor. Podporované možnosti zobrazení grafů se liší podle nastavené konfigurace monitorované oblasti. Uživatel má možnost zobrazit pouze

data intenzity ozařování přímo naměřené senzorem. Pokud definuje senzorem monitorovanou plochu fotovoltaických panelů a jejich účinnost, zpřístupní se možnost zobrazení teoreticky generované energie. Pokud je dále monitorovaná plocha propojena s invertorem a dodána jím naměřená hodnota reálné energie, je možné zobrazit porovnání teoretické a reálné energie. Také se zpřístupní graf performance ratia. Zobrazení grafů podporuje také možnost srovnání dat z více senzorů. Interaktivity je dosaženo tzv. brushem, kdy si uživatel může v pomocném grafu vybrat na časové ose data, která chce zobrazit.

Detaily výpočtu grafů s využitím numerické integrace a interpolace jsou uvedeny v další kapitole. Celá aplikace má v budoucnu běžet na soukromém serveru, dočasně byla ale pro testovací účely nasazena na Google Cloudu.

ACKNOWLEDGEMENT

I would like to thank to my supervisor Ing. Radovan Holek, CSc. who guided me, provided me with support and especially helped me with the data model design. Further, I would like to express my gratitude to Erik Haverkamp from ReRa Solutions for cooperation on installing and connecting sensors to developed application. I highly appreciate the support and encouragement from my wife Kateřina Hertlová who created the environment in which this thesis could have been formed. My gratitude also belongs to my brother Vojtěch Hertl for thesis revisions.

Brno 1. 6. 2020 . . . .

author's signature

Contents

Introduction **21**

1 Theoretical part **23**

1.1 Properties of sunlight . . . 23

1.1.1 The Sun radiation spectrum . . . 23

1.1.2 Daily solar irradiance . . . 24

1.2 Photovoltaic installations . . . 26

1.2.1 Types of PV modules . . . 26

1.2.2 Efficiency of PV modules . . . 27

1.2.3 Performance ratio . . . 27

1.3 My solar scan . . . 29

1.3.1 The sensor . . . 30

1.3.2 The connection box . . . 32

1.3.3 Common interface . . . 32

1.4 Sensor connection possibilities . . . 33

1.4.1 Project specific requirements . . . 33

1.4.2 Considerations for choosing optimal connection technology . . 34

1.4.3 Overview of connection technologies . . . 37

1.4.4 Selected connection technology . . . 41

2 Practical part **43**

2.1 Application solution architecture . . . 43

2.1.1 The single page application . . . 43

2.1.2 Used Frameworks and languages . . . 46

2.2 The Data Model . . . 49

2.2.1 Overview entity relationship diagram . . . 49

2.2.2 System entities ERD . . . 51

2.2.3 User specific entities ERD . . . 53

2.2.4 Monitored area concept with ERD . . . 55

2.2.5 Instrument ERD . . . 59

2.2.6 PV panel ERD . . . 59

2.2.7 Inverter and Instrument measurement ERD . . . 60

2.2.8 To-date implemented model . . . 60

2.3 Programming the model . . . 62

2.3.1 Transferring model into database . . . 62

2.3.2 Querying the database . . . 63

2.4 Payload and REST communication . . . 65

2.4.1 Spring REST controller . . . 65
2.4.2 Error handling . . . 67
2.5 Securing the application with login . . . 67
2.5.1 The authentication process . . . 67
2.5.2 Spring method level security . . . 68
2.6 Graphical user interface . . . 68
2.6.1 Create React App . . . 70
2.6.2 React-Spring integration . . . 70
2.6.3 Client-server communication . . . 70
2.6.4 App.tsx . . . 71
2.6.5 The Main page . . . 71
2.6.6 The general section . . . 72
2.6.7 The administration section . . . 76
2.7 Computational base . . . 78
2.7.1 Measurement data flow . . . 80
2.7.2 Computational details . . . 81
2.8 Cloud deployment . . . 83

3 Discussion **85**

4 Conclusion **89**

Bibliography **91**

List of symbols, physical constants and abbreviations **95**

List of appendices **97**

A Content of enclosed CD **99**

List of Figures

1.1 The Sun spectral irradiance 24
1.2 Sun position at the sky 25
1.3 Daily insolation 25
1.4 Block diagram of complete solution 29
1.5 The sensor 31
1.6 The connection box 32
1.7 Range and data rate of wireless technologies 35
1.8 Simplified OSI model 36
1.9 Comparison of LPWAN technologies 40
2.1 Comparison of SPA to traditional Web application 45
2.2 Used Frameworks and languages 46
2.3 ERD overview 50
2.4 System entities ERD 51
2.5 User and instrument STD diagram 52
2.6 User specific entities ERD 54
2.7 Description of monitored area concept 55
2.8 Monitored area ERD 56
2.9 Supported Configurations 58
2.10 Instrument ERD 59
2.11 PV panel ERD 60
2.12 Inverter and Instrument measurement ERD 61
2.13 Method level security configuration 69
2.14 The main page menu 72
2.15 Measurement picker 73
2.16 Irradiance and energy graphs 74
2.17 Energy comparison and performance ratio graphs 75
2.18 Sensor administration page 77
2.19 Monitored areas administration page card 79
2.20 Measurement data flow diagram 80
2.21 Graph computation methods 82

List of Tables

1.1 MSS measured parameters 33
2.1 User STD description 53
2.2 Instrument STD description 53

Listings

1.1 An example of one My Solar Scan reading 33
2.1 Programatically defined Instrument entity 64
2.2 Instrument repository 65
2.3 Instrument controller 66
2.4 Instrument DTO 66
2.5 Controller error handling 67
2.6 New instrument request 71

Introduction

Climate and environmental related challenges have been extensively debated recently. In a response to these challenges the European Commission presented the European Green Deal in December 2019 [2]. The Communication from the European Commission refers to scientific climate researches held by various international institutions such as The Intergovernmental Panel on Climate Change (IPCC) [3]. According to the findings, the European Commission states that the atmosphere is warming and the climate is changing. From that reason the European Green Deal was set out binding the European countries to transform in such way that they have no net emissions of greenhouse gases by 2050. Though this objective may seem very ambitious, it was endorsed by European Council on December 12th [4].

One of the most important key actions leading to fulfillment of European Green Deal is the transition to clean, affordable and secure energy as it accounts to more than 75 % of the EU's greenhouse gas emissions. The solar energy is naturally an affordable possibility but yet it comes with challenges.

The Sun is technically speaking an infinite energy source but ist supply is less predictable compared to conventional power plants. The generated power can not be controlled and it is dependent on daily and seasonal effects. Apart from predictable fluctuations caused by sunrise and sunset, the power output is also fluctuating due to changing weather conditions. In order to smoothly connect various solar plants[1] to the grid, the accurate solar generation forecasting is required [5]. Although the weather forecast should be taken into account, the module level measured data are an important aspect [6].

Monitoring of photovoltaic modules finds utilization also amongst the module owners. Comparing measured data through time may help to reveal potential efficiency losses which may be caused by several degradation and failure modes. One of the key parameters used for monitoring performance is called *Performance Ratio* [7].

When developing a sensor, an efficient way of measuring and storing data should be considered. Fortunately, with the explosive growth of the Internet of Things (IoT) technologies come the new connection possibilities minimizing energy consumption while maintaining long range and reliability [8]. Several connection technologies ranging from widely known Wi-Fi, Bluetooth, over Cellular to Low Power Wide Area Networks (LPWAN) are compared in this thesis with respect to planned utilization.

The data which were collected and computed in this thesis come from the sensors developed by ReRa Solutions company. The description of current architecture is

[1]Including small sources as on solar roofs of family houses.

brought together with the drawbacks and optimization procedures that the sensors might undergo in future.

The main focus of this thesis is to develop an application listening for the sensors input data, log them into the database, and display them back to the user in the graphical form i.e. graphs. The practical part guides through the whole process of development. It starts with the overview of an application architecture and used languages and frameworks. Detailed description of the data model design reflecting the real-world scenario and its transfer into the database in the form of programmable entities follows. Next sections describe the client-server communication via REST API and mention the authentication methods. After that the focus is shifted towards the applications GUI from environment configuration to selected graphs rendering. The last two sections bring details about the computation methods and touch the cloud deployment procedure.

1 Theoretical part

1.1 Properties of sunlight

1.1.1 The Sun radiation spectrum

Light is a part of electromagnetic spectrum and can be described both as waves with a certain wavelength or as energy packages - *photons*. This concept is called *wave-particle duality*. The relation between wavelength λ and photon energy E is given by

$$E = hf = \frac{hc}{\lambda}, \tag{1.1}$$

where h is Planck constant, c is the speed of light in vacuum and f is frequency [9].

The spectral properties of sunlight can be described by *spectral irradiance* which tells how much energy in 1 s (i.e. power) from the wavelength interval strikes the unit area. The relation states

$$F(\lambda) = \Phi E \frac{1}{\Delta\lambda}, \tag{1.2}$$

where $\Phi = \frac{\#\,\text{photons}}{\text{m}^2\text{s}}$ is the photon flux [10].

By integrating (1.2) over the chosen wavelength range we get the *radiant power density*[1]. By setting the lower resp. upper integral bound to 0 resp. ∞ the overall irradiance through the whole spectrum is obtained [10]:

$$H = \int_0^\infty F(\lambda)\mathrm{d}\lambda. \tag{1.3}$$

The Sun spectrum irradiating the Earth is standardized and referred to as Air Mass Zero (AM0) with corresponding irradiance of 1353 Wm^{-2}. Due to the scattering and absorption in the atmosphere the irradiance at the Earth surface is different mainly depending on the distance which the light has to travel through the atmosphere. And because the traveled distance is determined by the angle of incidence Θ[2] the quantity *Air mass* has been established as

$$AM = \frac{1}{\cos\Theta}. \tag{1.4}$$

The standard spectrum at the Earth's surface is called AM1.5G[3]. It corresponds to the angle of incidence $\Theta = 48°$. AM1.5G irradiance is computed to be approximately 970 Wm^{-2}, however the the standard spectrum has been further normalized to give 1000 Wm^{-2} [10].

[1]Also simply called *irradiance*.

[2]The Θ is the angle between the light beam and the vertical to the Earth surface. $\Theta = 0°$ when the Sun is directly overhead.

[3]The letter 'G' stands for global. Global spectrum comprises both direct and diffuse part of the light.

Furthermore, the spectral properties of sunlight are similar to the radiation of black body with the temperature around 5762 K [9]. The curve corresponding to black body radiation wraps the spectral irradiance of AM0 and AM1.5G as can be seen in Figure 1.1.

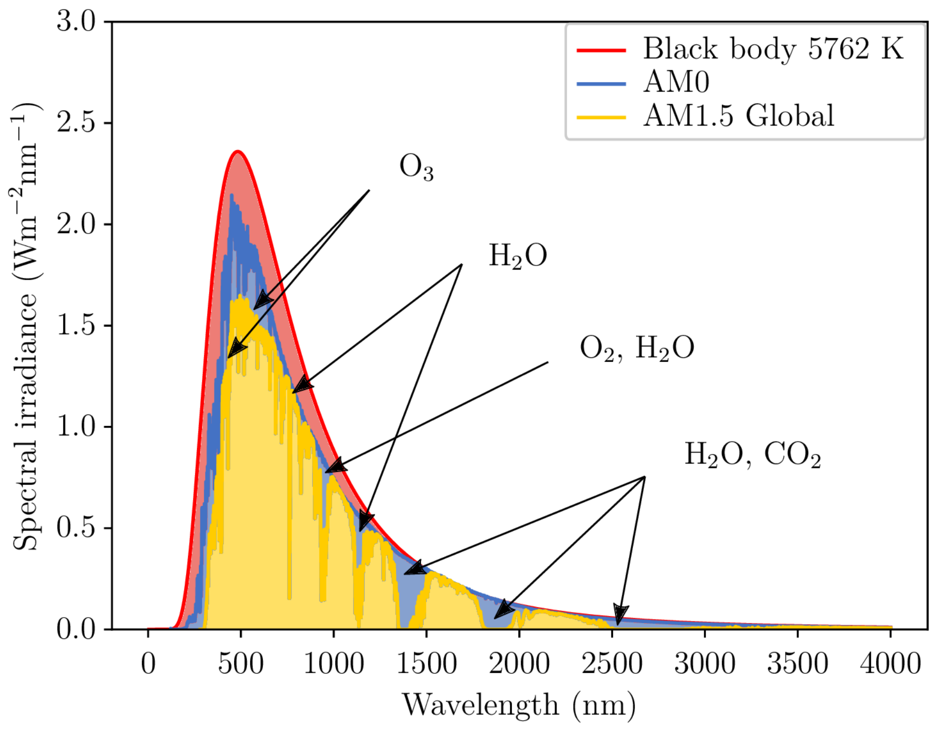


Fig. 1.1: The Sun spectral irradiance. AM0 is the spectrum outside the atmosphere [11], AM1.5 Global is the standardized spectrum incident on the Earth surface [12]. Both curves are wrapped with Planck's blackbody radiation spectrum at 5762 K [10].

1.1.2 Daily solar irradiance

It is generally known that the Earth is orbiting the Sun. From the human point of view the Sun moves over the sky. This phenomenon will be referred as the motion of the Sun. The Sun position in any time of the day may be described by two angles α_s-the Sun azimuth and γ_s-the Sun elevation (see Fig. 1.2). In photovoltaics (PV) the zero azimuth belongs to the south. Due to the tilt of the Earth's axis the days are longer in the summer than in the winter. In other words, the Sun reaches higher elevation in the summer. It is also imaginable that γ_s is very low in the morning, it reaches its highest value around noon a then declines again. Since there is a simple relationship between γ_s and Θ (from Eq. 1.4) $\Theta = 90° - \gamma_s$, it can be seen that the Air mass is the smallest when the Sun elevation is maximal (i.e. $\gamma_s = 90°$). To sum up, the Sun irradiance is proportional to the Sun elevation which changes during the

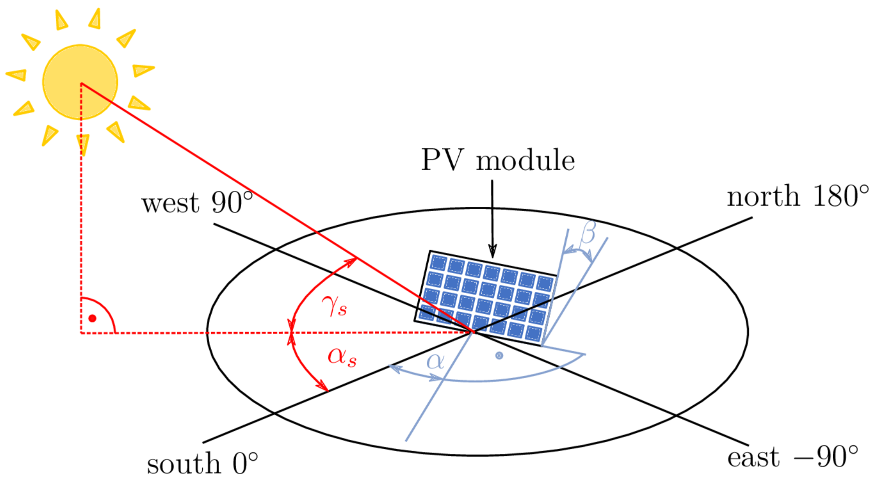


Fig. 1.2: Definition of the Sun position at the sky. Two descriptive angles α_s-the Sun azimuth and γ_s-the Sun elevation are present. PV module tilt and azimuth which play important role when designing the solar system are also displayed. Taken and modified from [13]

day and also with season. Figure 1.3 brings the descriptive comparison of irradiance on day time dependence for four important days in the year [13]. It can be seen that

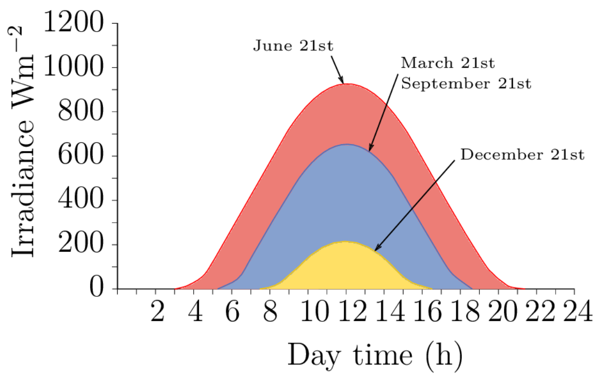


Fig. 1.3: Daily insolation in Berlin taken in four important days through the year. Taken and modified from [13].

on June 21st the irradiance reaches its maximum 1000 Wm^{-2} which corresponds to AM1.5G. From measured irradiance more common form of data used in photovoltaic system design can be computed - the *solar insolation*. The solar insolation is the total amount of solar energy received at a particular location during specified time period. It comes with the units of energy density (kWh/m^{-2}day). By integrating the daily irradiance curves over day time the daily solar insolation values are obtained.

When further summing solar insolation of all days during the year, the overall yearly values are obtained [10].

Looking back at Figure (1.2) the orientation of PV module described by tilt angle β and azimuth angle α plays also an important role what takes to maximizing the module insolation. As the Sun azimuth and elevation changes all the time, the optimal PV module tilt and azimuth has to be determined. The optimal values can be computed taking the average of many yearly insolations. For Berlin the optimal PV module tilt is $\beta = 30°$ and $\alpha = 0°$ (module is facing directly to south). In this configuration the average yearly insolation reaches approximately 1180 kWh/m^{-2}[4] [13].

The important takeaway for designing the sensor is that it must have the same tilt and azimuth as the PV module. It means that the area of PV modules which the sensor monitors does not play as important role as their orientation.

1.2 Photovoltaic installations

1.2.1 Types of PV modules

The PV modules consist of solar cells wired together and put inside of a frame behind protective glass. Although there are more aspects of dividing PV module types, in further text the division based on used solar cell type will be described. Basically, three main types are recognized:

- mono-crystalline module
- multi-crystalline module
- thin film module (amorphous, microcrystalline, CdTe or CIS module) [13]

Since by the end of 2017, the crystalline silicon (c-Si) was used as manufacturing material for 95 % of global PV module production, it will be focused on in further text [14].

Mono-crystalline Si is usually grown as one crystal in a large cylindrical ingot. In comparison to multi-crystalline Si, it has better material properties but is also more expensive. Multi-crystalline silicon consists of more crystals resulting in the presence of grain boundaries which act as recombination centres. Grain boundaries reduce solar cell performance by blocking charge carrier flow inside the material [10].

The record lab cell efficiencies are 26.9 % for mono-crystalline Si, 22.3 % for multi-crystalline Si, 23.4 % for CIGS and 21.0 % for CdTe solar cells. However, when these cells are combined into PV module the maximum reached efficiency for

[4]This value also incorporates changing weather conditions which are important factors influencing the overall insolation.

mono-crystalline Si is 24.4 %. The average commercial wafer-based silicon modules have efficiency of 17 % (super-mono 21 %).

The global overall module production show the leading role of multi-Si (62 %) followed by mono-Si (33 %). Thin film technology accounts for only 5 % of global market module production. [14].

In California, the noticeable shift towards mono-Si was spotted. The reason can be explained by the narrowing in price between the two technologies. Moreover it is predicted, that in 10 years the mono-crystalline Si will grow to almost 90 % of the global market [15]. This trend can be supported by observed decline of multi-crystalline technology from 70 % (2016) to 62 % (2017) [14].

The leading producer of PV modules is China and Taiwan with 70 %. Europe and North America have only about 7 % [14].

1.2.2 Efficiency of PV modules

The efficiency of a solar cell is given by the following relation:

$$\eta = \frac{P_{\mathrm{mp}}}{P_{\mathrm{in}}} = \frac{FF V_{\mathrm{oc}} I_{sc}}{P_{\mathrm{in}}}, \tag{1.5}$$

where P_{in} is the incident power at the solar cell. P_{mp} is the *maximum power point* which is the point on the solar cell I-V curve where the product of I and V is maximal. Respective current and voltage values in maximum power point are denoted as I_{mp} and V_{mp}. FF stands for Fill Factor which is basically the squareness of the I-V curve.

$$FF = \frac{V_{\mathrm{mp}} I_{\mathrm{mp}}}{V_{\mathrm{oc}} I_{\mathrm{sc}}}, \tag{1.6}$$

where V_{oc} and I_{sc} are open circuit voltage and short circuit current respectively [10].

When talking about efficiency of PV modules, the power of incident radiation is usually expressed as $P_{\mathrm{in}} = AH$, where A is the area of PV modules and H is the irradiance. Usually, the efficiency in standard conditions η_{stc} is stated in the PV module datasheet [13].

1.2.3 Performance ratio

The efficiency of PV panels may be quite easily computed, however other efficiency loses come into account when determining the output of the PV plant as the whole. Additional loses may occur in wires, contacts or in the inverter. At this place another quality descriptive quantity has to be introduced.

The performance ratio incorporates all efficiency loses and informs about overall efficiency of a PV plant. Computed PR allows to compare the energy output of a PV plant with the other PV plants or the same plant over the time period. The plant may be considered to run optimally on being commissioned and therefore its initial PR value should be maximal. Degradation in PR over time may indicate a possible fault.

PR can be computed from simple relation

$$PR = \frac{E_{\text{actual}}}{E_{\text{expected}}}. \tag{1.7}$$

The E_{actual} is the value which the PV plant owner might get from his DC to AC inverter. The data from the developed solar sensor will then be used for computation of expected energy as [7]

$$E_{\text{expected}} = AQ\eta. \tag{1.8}$$

The relation needs further explanation. A is the total PV module area and η is the efficiency of PV panel. Q is the solar insolation which can be computed from sensor measured data. When calibrated and placed close to the monitored module preserving the tilt and azimuth, the radiant power density H incident on the PV module and the sensor should equal. The PV module comprises Si photo detector being able to measure just mentioned H. Integrating H over time which corresponds to the reading on the inverter will produce solar insolation that can be directly substituted into Eq. (1.8).

As mentioned in Section (1.1.2), alignment of the sensor with monitored PV module is essential. Therefore, if there is an example of PV plant having multiple differently tilted PV modules, than for each orientation ideally one sensor should be used (see Section 2.2.4 for monitored areas concept). Then Eq. (1.8) would change to the sum

$$E_{\text{expected}} = \sum_{i}^{n} A_i \eta_i Q_i, \tag{1.9}$$

under the assumption that n is the number of sensors which cover all PV panel areas.

Talking about obtaining the value of Q from H by integration, with the sampling period T_{s} in range of minutes the term summing should be used instead. Solar insolation over the given time period t would be then computed from

$$Q_i = \sum_{j}^{m} H_j t_j, \tag{1.10}$$

where m being the number of summed values can be obtained from $m = t/T_{\text{s}}$ assuming constant time interval t_j[5].

Putting everything together, the final equation which will be used for computation of PR yields

[5]More precise computation method is introduced in Section (2.7.2).

$$PR = \frac{E_{\text{actual}}}{\sum_i^n (A_i \eta_i \sum_j^m H_j t_j)}. \tag{1.11}$$

The factors influencing the PR include environmental factors such as temperature and shading of PV module. However, also conduction losses, efficiency of the inverter and the PV module take part [7].

In order to obtain reasonable values, the proper calibration of solar sensor has to be undertaken on regular basis. As was already stated the sensor is made of Si as well as the majority of PV modules so its absorption and temperature losses should be similar to the monitored module. This is true under assumption that the sensor is kept at the same temperature as monitored PV module, otherwise further correction utilizing temperature measurements would need to be applied.

According to [14], the recent installations may reach as high values of PR as 90 %.

1.3 My solar scan

This section describes current architecture of My Solar Scan (MSS) which is the commercial name of the irradiance measurement device developed at ReRa Solutions. As the project assignment reveals, the sensor is ready to be tested in the field but solid way of connecting it to the cloud is not yet available. Simple schema of the whole solution is displayed in Figure 1.4. The incident solar irradiation is imparting

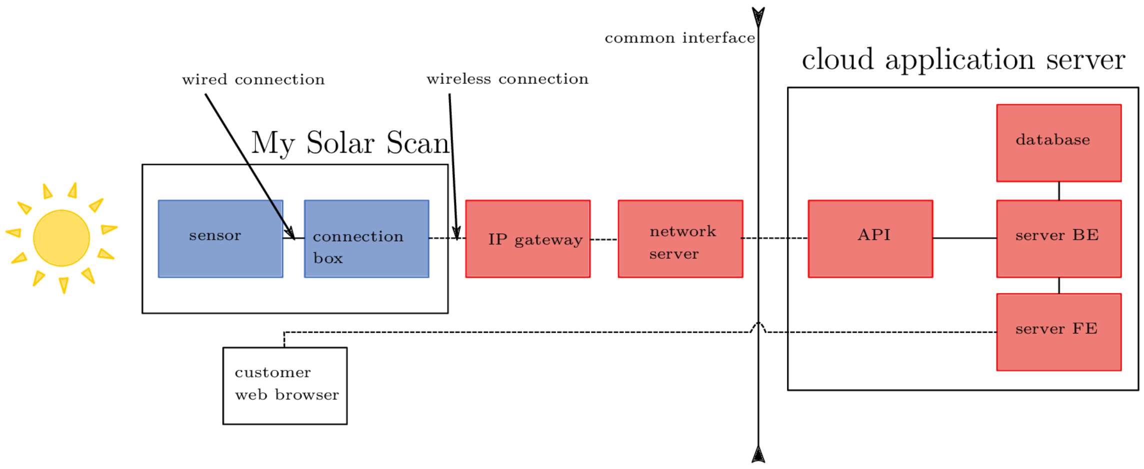


Fig. 1.4: Block diagram of complete solution. The solution consists of My Solar Scan and cloud server connected via IP gateway. Common interface is also depicted. Blue color refers to completed solution whereas red has to be optimized or developed completely new.

the sensor in MSS. The connection box which is wired to the sensor contains CPU which is taking care of triggering the measurement. Once the value is measured by the sensor, it travels to the connection box via wired connection. The connection box comprises Particle Photon which includes micro controller with a Wi-Fi chip [16]. Particle Photon is connected to the local network Wi-Fi router which serves as a gateway to the internet. The network server, in this case Particle cloud [17], serves as a monitoring platform for connected Photon devices. Real measured data can be seen there. Particle cloud then resends data to public application server API. Here the common interface line is drawn. No matter which connection box i.e. connection technology is used behind this interface, the application stays the same[6]. This allows to develop the application on the server independently. The application on the server exposes defined url endpoint to registered devices. Once the endpoint is called with measurement reading data, server back end evaluates and stores the data into database. The back end side also prepares the data to be displayed on server FE. The server FE is the only place where the customer can interact with the system. It resends user driven actions to BE and displays obtained data in user friendly form.

In further text, some of the individual blocks to the left of common interface will be described.

1.3.1 The sensor

The scheme of sensor operation is given in Fig. 1.6a. The most important part for irradiance measurement is a Si photodiode. Si material is convenient since it is very similar to material of monitored PV modules (see Section 1.2.1). When calibrated, the measured quantity is irradiance with units W/m^2. When not calibrated, the measured value is linearly dependent on the irradiance. Since no shielding is provided and the sensor does not track the sun and is oriented with the PV module, measured irradiance is global-tilted (direct plus diffuse).

The other measured quantities are not necessary for the sensor simplest application, but are essential for its autocalibration. The IR photodiode consist of the same Si photodiode only with IR filer.

For measuring UVA and UVB the Veml6075 Light Sensor is used. It converts solar UV light intensity to digital data. The peak sensitivities are at 365 nm and 330 nm for UVA and UVB respectively [18].

As RGB sensor the ISL29125 Renesas Light Sensor with IR Blocking Filter is used. It allows the user to optimize sensitivity based on specific application [19].

[6]Metaphorically speaking this common interface also draws a line between the theoretical and practical part of this thesis.

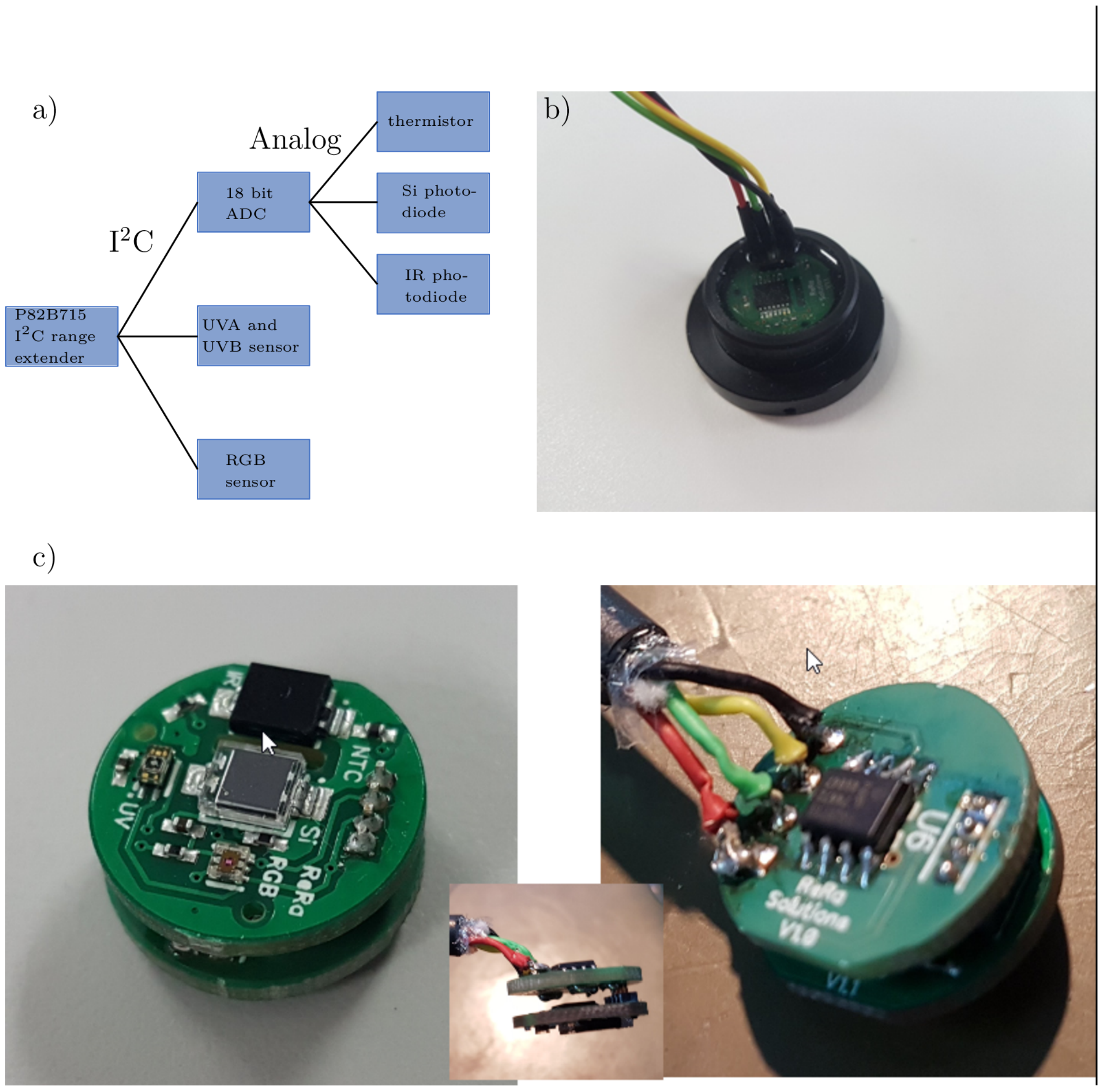


Fig. 1.5: The sensor. (a) The block diagram of sensor parts. (b) the sensor photo, (c) detailed view.

The last measurement which can be taken is with negative temperature coefficient (NTC) thermistor. The temperature measurement starts to play an important role if the same temperature of PV module and the sensor could not be guaranteed since it is one of the main factors influencing the efficiency of PV module and hence PR of PV plant (see Sec. 1.2.3).

1.3.2 The connection box

The connection box comprises Particle Photon. Its ARM Cortex M3 micro controller is the brain of the whole MSS. It can be programmed in C++. Every 60 seconds it triggers the measurement since it is connected via I^2C's SDA and SCL conductors to the sensor. Thanks to Broadcom Wi-Fi chip it can connect to the router which is available at installation site. Supported Wi-Fi standards are 802.11b/g/n [16]. After the connection has been established, it can be programmed and maintained from Particle cloud [17]. The price for Particle Photon is 20 $. When installed, it is powered from the simple adapter. The picture of connection box is displayed in Fig. 1.6.

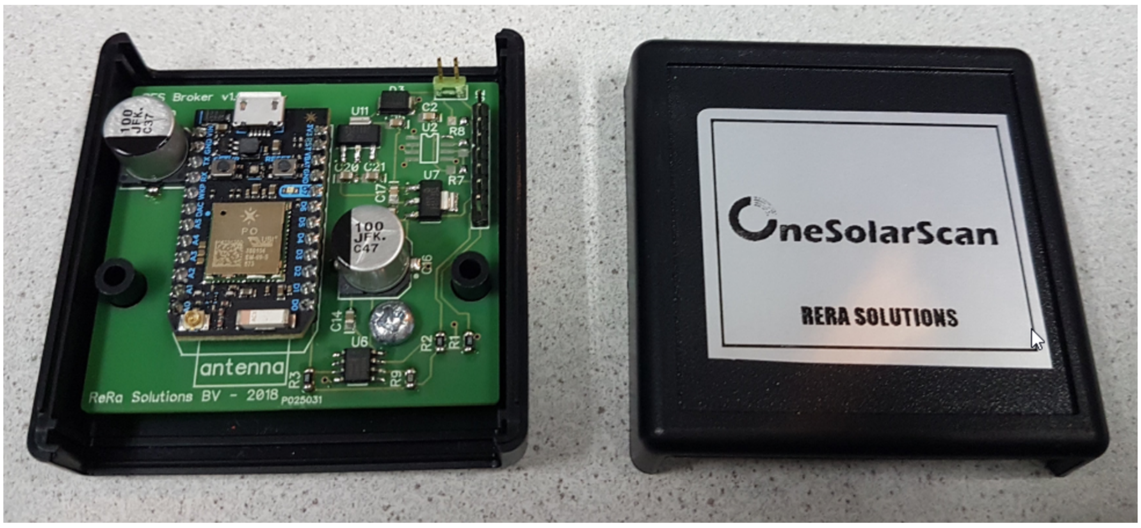


Fig. 1.6: The connection box consists mainly of Particle Photon.

1.3.3 Common interface

The last and yet boundary element of block diagram will be described in this theoretical section. The communication over the interface is carried out by transferring the data packets. Its structure is simple string beginning with ID of corresponding

Listing 1.1: An example of one My Solar Scan reading

```
{3100370012473438323536;
mss_si:4363;mss_ir:1832;mss_red:6415;mss_green:6113;
mss_blue:5680;mss_uva:202.10;mss_uvb:228.20;
mss_ntc:1521203}
```

Photon device followed by 8 measured values. An example of one reading can be seen in Listing 1.1. Description of the reading is provided in Table 1.1.

Tab. 1.1: MSS measured parameters names together with their data types and a short description.

Measurement name	Data type	Measured range
mss_si	long	370-1100 nm
mss_ir	long	750-1100 nm
mss_red	uint	visible red
mss_green	uint	visible green
mss_blue	uint	visible blue
mss_uva	float	UVA
mss_uvb	float	UVB
mss_ntc	long	temperature

1.4 Sensor connection possibilities

With the increasing number of connected devices to the internet, more advanced connection possibilities have to be developed in order to satisfy the demand. Here the topic actually touches the Internet of Things (IoT) which refers to the inter connection and exchange of data among devices [8]. In this section the performance of widely known technologies such as Wi-Fi or Bluetooth will be compared to Low Power Wide Are Network (LPWAN) technologies. At first the connection technology requirement for application in this project will be summarized.

1.4.1 Project specific requirements

In order to be able to choose the optimal connection technology the comprehensive project requirements have to be outlined. Following list displays the requirements sorted by the relevance on connection technology design.

- The payload length of one measurement is around 150 bytes[7].
- The expected reading period is between 1 to 15 minutes.
- One-way communication from MSS to the cloud is desired. Eventual sensor calibration will be done on the server.
- There will always be 1:1 relationship between the sensor and the connection box.
- The scenario in which more MSS's will be used at the same site is probable.
- The expected number of sensors connected to one db is in the thousands.
- The timestamp is logged once the data reach the db.
- Calibration is not required during the installation. MSS will start sending data once it is connected.
- GPS module will not be included in MSS and its position will be manually set during installation together with the tilt and azimuth.

The main reason for doing research over connection technologies are however the problems encountered with current Photon Wi-Fi connection. It appeared not to be strong enough which resulted in data loss. Another problem is its connection to end-user's Wi-Fi router. This solution is very fragile and prone to failure as the end-user can break all communication simply by resetting his Wi-Fi router password.

1.4.2 Considerations for choosing optimal connection technology

Spectrum

Essentially, either licensed or unlicensed spectrum may be chosen for wireless connectivity. The licensed spectrum provides exclusive access to particular channel which guarantees that the communication will be undisturbed by other connected devices. The typical example is a cellular communication. With privileges also come the higher price. On the other hand, unlicensed frequency bands reserved for Industrial, scientific and medical (ISM) applications are free to use. Popular ISM bands include frequencies 2.4 GHz and 5 GHz available worldwide which found implementation in the most familiar wireless connection Wi-Fi. In Europe for example, additional ISM bands with frequencies 433/868 MHz are used [20].

Range and capacity

Whereas the higher-frequency bands offer more bandwith resulting in higher throughput, the lower-frequency waves propagate further bringing higher ranges. In general

[7] Computed from the string representation of one reading (Lst. 1.1)

there always is the tradeoff between the distance to be covered and the data capacity. For quantifying the difference the link between two radios can be described either as line of sight (direct optical path) or non-line of sight (with obstruction in the optical path) [21]. Concerning the range and data rate, multiple concepts have been defined as can be seen in Figure 1.7.

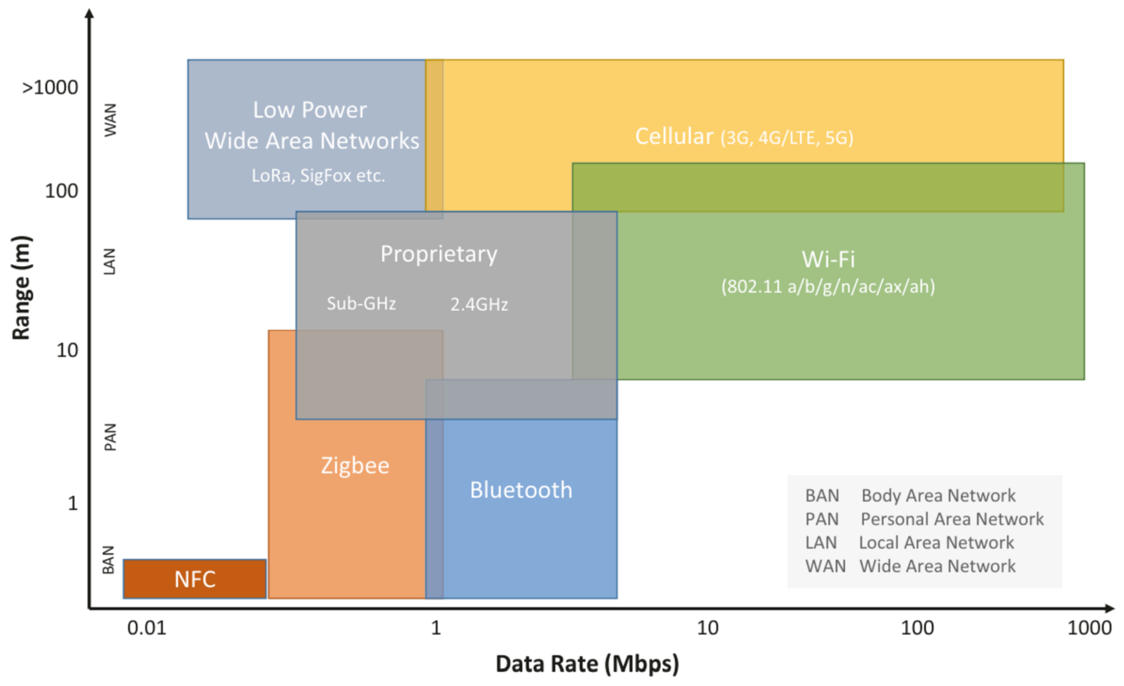


Fig. 1.7: Range and data rate for various wireless technologies. The picture also shows the different ranges naming [21].

Body Area Network (BAN), Personal Area Network (PAN), Local Area Network (LAN) and Wide Area Network (WAN) can all be seen in the Figure. BAN refers to the smallest proximity communications such as NFC when the two devices are basically touching each other. The representative of PAN might be Bluetooth handset. LANs are either wired or wireless represented by various Wi-Fi devices. WANs are long range networks spreading across entire globe [20].

Network topology

The network topology refers to the way nodes in a network are arranged. Point-to-point are suitable for connection of two devices and data transfer. Star topology is basically one central node providing for example gateway to the internet for the other nodes. Implementation may be found in station devices (e.g. smartphones, laptops) connected to Wi-Fi access point. Mesh topology allows every node to connect to multiple other nodes. One or more nodes then serve as the internet gateway. The advantages of mesh network are extended network range while maintaining the low transmission power, as the signal can travel through multiple hops. Another benefit is increased reliability by enabling more than one relay paths. The drawbacks are the longer delays.

Interoperability, standards and security

With increasing number of devices from different manufactures the standardization starts to play an important role. Some standards define one or several layers of Open Systems Interconnection (OSI) model[8] (Fig. 1.8), others specify the network end-to-end. The two most important institutions defining standards are The Insti-

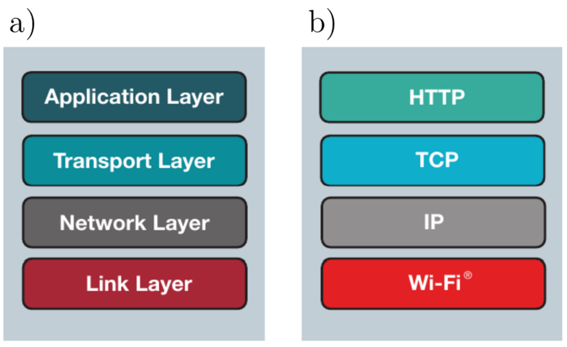


Fig. 1.8: (a) Simplified OSI model, (b) example of TCP/IP protocol stack [20].

tute of Electrical and Electronics Engineers (IEEE) and The Internet Engineering Task Force (IETF). IEEE defined for example 802.x family of standards which incorporates:

- 802.3 - Ethernet specification,
- 802.11 - WLAN specification (also including Wi-Fi),
- 802.15.4 - wireless PAN standrad used for example by Zigbee.

IETF is focused on defining internet standards through request for comments (RFCs). The examples are:

- RFC 791 - IPv4 protocol
- RFC 793 - TCP protocol
- RFC 2616 - HTTP/1.1 [20].

Security is also becoming more important with increasing number of connected devices. Management interfaces should be using HTTPS and the air link communication should be encrypted using network, mesh of link key [21].

[8]OSI model breaks the communication into functional layers allowing easier implementation of interoperable networks [20].

1.4.3 Overview of connection technologies

According to specification in Section (1.4.1) the basic requirements for connection technology for this projects are low data rate, one-way communication, customer interference proof solution, with the lowest price possible and low energy consumption (optional).

Wi-Fi

Wi-Fi is probably the most ubiquitous connection technology nowadays. Its certification programs managed by Wi-Fi Alliance allowed its remarkable interoperability resulting in the huge popularity worldwide. It uses 2.4 GHz and 5 GHz ISM bands. However, the Wi-Fi is by far not only to be used for connecting personal smartphones to the internet through access point. Until recently Wi-Fi and TCP/IP software have been complex and demanding in terms of CPU usage. Nowadays, with the advance of new silicon devices, it is possible to use Wi-Fi even in small microcontrollers. In order to reduce power consumption, advanced sleep protocols have also been introduced. IoT device sending data over Wi-Fi in short intervals can last over a year on two AA alkaline batteries [20]. Low-Power Wi-Fi standard 802.11ah which operates in the sub 1-GHz range (typically 900 MHz) brings extended range with efficient power profile. Due to lower frequencies, the penetration of signal is better. Outdoor range of up to 1 km is promised. However, the lack of global 900 MHz standard slows its deployment down [21]. For example, current solution of connection box with Photon does not support long range, low power 802.11ah standard (see Sec. 1.3.2).

Bluetooth

Bluetooth is maintained by Bluetooth SIG. Two incompatible types of Bluetooth are on the market - Classic Bluetooth and Bluetooth Low Energy (BLE). Classic Bluetooth has been a standard communication technology between laptops and smartphones. Since it uses 2.4 GHz ISM band it can operate over Wi-Fi reaching high transfer rates up to 25 Mbps[9]. As the title suggests BLE reduces power consumption with cost of lower data throughput and it is therefore suitable for IoT applications. The interoperability of Bluetooth devices have been achieved through special Bluetooth SIG certification programs. BLE further offers generic attribute profiles allowing the list definition of characteristics and attributes of given node. In 2014 the Bluetooth 4.2 brought advanced security standards and elliptic curve cryptography for message encryption. In version 4.2 BLE also added IP support

[9]When operating over Wi-Fi it is referred to as Bluetooth High Speed.

profile allowing to connect devices to the internet via BLE gateways. In 2017, mesh profiles have been specified [20].

Zigbee

Zigbee provides a whole connectivity solution with cloud connectivity support. The standard is maintained by Zigbee Alliance which comprises about 400 companies from different industries. Zigbee operates in the 2.4 GHz (global) and 868 MHz (Europe) ISM band and has been designed in mesh topology. Therefore it is able to scale through multihop operations. Fault tolerance and reliability are also increased. Zigbee finds utilization in low data rate sensors and actuators. It minimizes the air time of packets by using short frames which also lead to power consumption reduction [20]. Transmission distances range from 10 to 100 meters. Sub-1 GHz channel ranges up to 1 km. Zigbee supports connection of hundreds of nodes per network (up to 64k) [21].

6LoWPAN

The name of this connection technology means IPv6 Low Power Wireless Personal Area Network. It enables small power devices to communicate with any IP-based device over internet. Since the standard allows multiple optional modes, solutions from different vendors are not interoperable at the local network level which slowed down expansion of this technology. In IP gateway which is needed to connect to internet usually the conversion protocol between IPv4 and IPv6 exists. This mesh-based network topology operates in 2.4 GHz and 868/915 MHz ISM bands offering large network size, reliable communication while maintaining low power consumption [20].

Celluar

Cellular is the first technology on the list using the commercial frequency bands. Its main advantages are a broad coverage, already developed base station infrastructure and continuous connectivity to the backbone network. The cellular technology has a potential to cover thousands of devices per square kilometer. The drawback include fees due to licensed spectrum, higher power consumption and gaps in coverage. Cellular IoT try to overcome high power demand by scaling the existing technology down. For IoT applications, several technologies have been standardized including Long-Term Evolution for Machines (LTE-M) and Narrowband IoT (NB-IoT). Whereas LTE-M has higher bandwidth offering fast speed, NB-IoT is suitable for smaller data rates from infrequent communication hence ideal for sensor devices and will be mentioned in following section. The advantage of cellular technologies is without any doubt the end-to-end security from the device to the cloud server.

In this section the note about the incoming 5G technologies has to be made. The 5G is promised to be suitable for usage in three categories from high-level point of view.

1. Enhanced Mobile Broadband will require high data rates across a wide coverage area (virtual reality)
2. Massive Machine-Type Communications will support large number of devices in a small area (smart cities, smart sensors).
3. Ultra-Reliable Low-Latency Communications require low-latency and high reliability (remote surgery, autonomous vehicles).

From sensoring point of view the Massive Machine-Type Communications are the most promising 5G implementation. 5G will come with solutions which will maximize the arrival rate of data from huge number of connected devices [21].

LPWAN (Sigfox, LoRa, NB-IOT)

Up to this point several IoT connection technologies have been listed. However Low-power wide area network (LPWAN) technologies have been optimized from the beginning for connection of sensors. LPWAN aims at applications requiring low mobility, low rate, small data packets at infrequent intervals optimizing the energy consumption. The cost of radio chipset may be less than 2 EUR and yearly operation cost as low as 1 EUR [8]. Since the LPWAN technologies stand out in the presented list of IoT technologies, the direct comparison will be done between them in this section [21].

Sigfox offers end-to-end IoT connectivity solution. The Sigfox was available in 50 countries in August 2018. The Sigfox provider is similar to mobile operators but it aims directly at IoT sensors. It uses ultra-narrowband (100 Hz) techniques and operates in sub-1 GHz range (Europe 868 MHz). The bidirectional communication is limited since the maximum number of messages per dey are 140 (upload) and 60 (download). Therefore each uploaded message cannot be confirmed. Maximum payload length is 12 B (upload) and 8 B (download). It offers up to 10 km resp. 40 km range in city resp. rural area between the node and the base station with maximal data rate of 100 bps. Sigfox does not support authentication and encryption. To ensure the reliability each message is transmitted multiple times over different frequency channels as the base stations can receive simultaneous messages over different channels [8].

LoRa (Long Range) is a physical layer technology modulating signals in sub-1 GHz range. LoRaWAN is LoRa based communication protocol standardized by

LoRa Alliance. The frequency is the same as in Sigfox case. It offers higher data rates up to 50 kbps allowing also the bidirectional communication and unlimited number of messages/day. Maximum payload length is 243 B. The range is up to 5 km resp. 20 km in city resp. rural area. LoRa also supports AES 128b encryption. Each transmitted message is received by all base stations in the range and the duplicates are filtered at the network server. The network server also checks security and sends acknowledgements to the end devices. Additionally, the LoRaWAN comes with support of localization of end devices from difference in the same message received time by multiple base stations. Generally, with increasing number of base stations in range the reliability but also the network price grow. LoRa comes in three classes

1. class-A (Bidirectional end devices): The lowest power configuration for applications that only require short acknowledgement after uploaded message
2. class-B (Bidirectional end devices with scheduled receives lots): Additional receive windows are open. The network is able to recognize when the end device is listening.
3. class-C (Bidirectional end devices with maximal receive slots): End devices have receive windows opened almost continuously [8].

As was already said, **NB-IoT** uses licensed LTE bands (e.g. 700 MHz, 800 MHz, 900 MHz). Comes with the highest speeds of 200 kbps and Maximum payload length of 1600 B. Supports bidirectional communication and unlimited number of messages/day. The range is up to 1 km resp. 10 km range in city resp. rural area. It enforces standard LTE encryption. NB-IoT can be supported by software upgrade of existing LTE infrastructure as it in fact uses the same protocol yet minimized. It supports up to 100k end devices per cell [8].

To Sum up, the graphical LPWAN comparison can be seen in Fig. 1.9. NB-

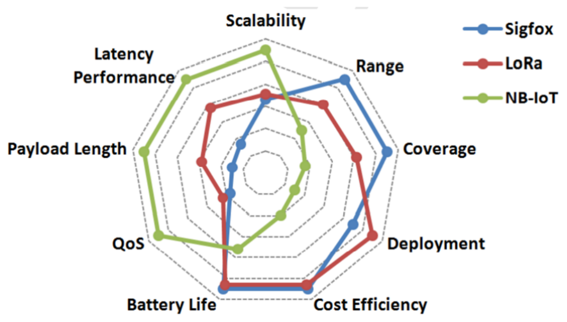


Fig. 1.9: Comparison of LPWAN technologies [8].

IoT offers the best quality of service as it operates on licensed spectrum. LoRa and Sigfox cannot keep up in this factor. On the other hand from the battery lifetime point of view NB-IoT devices consume more power maintaining synchronous communication. For applications requiring low latency LoRa class C and NB-IoT offer better performance. If low latency is not required, LoRa class A and Sigfox are the better choice. NB-IoT offers the highest scalability as it supports up to 100k devices in a cell compared to 50k of LoRa and Sigfox. NB-IoT wins also from payload length aspect. Sigfox's 12 B might not be enough for some applications of IoT. Sigfox however provides the highest range. One city can be covered with one single base station. LoRa's range is smaller needing 3 base stations to cover city of Barcelona. NB-Iot's range is smallest focusing on the places out of reach of typical cellular technologies (indoors). As the price of base stations is very high, its use is limited to the places where the signal already is which can limit places in rural regions. LoRa and Sigfox are mature and more spread technologies comparing to NB-IoT. Ending this comparison with one big LoRa advantage - its flexibility. Unlike the rest LoRa offers public (public base stations) as well as LAN (LoRa gateway) deployment models [8].

1.4.4 Selected connection technology

Concerning the project related specifications (Sec 1.4.1), the most suitable technology appears to be LoRa. As from the beginning most of the MSS's will be deployed in the Netherlands which according to page thethingsnetwork.org[10] already is densely covered with base stations. The current payload length of about 150 bytes which may slightly differ but will always be higher than Sigfox 12 byte limit excludes the use of Sigfox. Despite needing the data for insolation computation, extra low latency of NB-IoT is not needed. The lower price of the communication is more crucial. Concerning the price, current solution includes Photon which is \$ 19 today. Similar device from Particle which allow Celluar connection cost \$ 49.00 (Electron 2G Kit). However, recommended Electron 3G Kit already is \$ 69.00 excluding the price for transfered data. LoRa connection box will require for example RFM95 Ultra-long Range Transceiver Module (from Seedstudio) costing \$ 6.95 and microcontroller as for example Arduino Uno costing \$ 18 (Amazon). LoRa will also bring the desired user interference proof solution. However, with the modular design of MSS, multiple solutions might be tried and the best long term solution will be chosen afterwards.

[10]This site serves for registration of all LoRaWAN gateways as well as the maintenance of end devices.

2 Practical part

This part describes the solution of the cloud application server in detail. The main role of the server is to collect data from all connected sensors through API, store the data to database, compute and evaluate PV plant descriptive values and last but not least display it back to the user in an intelligible form. The server's position in the whole solution is depicted in the block diagram (Fig. 1.4). The design process was based on the project requirements (Sec. 1.4.1). On the top of that other software based requirements were added.

2.1 Application solution architecture

2.1.1 The single page application

Before diving into implementation details it is essential to describe the application use case. In other words, the first step in designing comprehensive and intuitive application is usually the research of end user needs. Despite recommendations taking this project time budget into account, the user testing phase at the beginning has been skipped. However, the application has not been designed out of thin air and the design carefully sticked to market experience and estimates of ReRa Solution expertise. Moreover, the outcome of this project is not ready to sale application but rather the first functional shot which could then be further developed user testing including.

Information architecture

The information architecture is one of the building stones of user-friendly design. It describes the structure of information to be displayed to the end user. And according to [22] it forces the application creator to think so the users do not have to. William Craig in [23] further evolves the information architecture idea and comes up with the best practices in application design. The article states that Every site should have clear **purpose** which it should serve and describing **personas**[1] was also recommended.

The personas

- Site admin. He is considered to be the owner of a PV plant and has access to the data of sensors[2] he owns. More specifically, it is a person with basic

[1]Personas incorporate different types of user of the application and then creating real people fitting them. [23]

[2]From now on the term sensor will be used interchangeably with the MSS. Although the MSS comprises sensor and connection box, the term sensor can be used without any information loss.

IT and technical skills. He has the photovoltaic panels mounted on the roof of his house and is interested if their real output corresponds to the expected output computed from MSS data. He understands what the performance ratio is and is able to insert inverter measured data into an application. He is able to manage accounts of users (e.g. family members) granting and removing their access privileges to site data.

- Site user. The persona managed by site admin. He has access to the site data. The only required skill is to be able to log into web application through web browser.
- (Super) Admin. The maintainer of the whole application. He has access to everything and can modify data of all users in the application. He creates and manages site admin accounts, can lock accounts, set operation state of sensors. After the installation of a new sensor he fills in the necessary data about the sensor and the measurement site.

The purpose of this application is to provide complete monitoring solution for the owner of a PV plant (Site admin). It allows to add a new sensor based on sensor GUID. Once the sensor is added the site admin is considered to be its owner from the application point of view. The application must allow to set important sensor properties (e.g. tilt, gps coordinates...) as well as the properties of the area it monitors (e.g. area, pv panel type, pv panel efficiency...). From the sensor measured data and the entered properties of monitored areas the application will be able to display irradiance data and compute the theoretical energy output of the monitored area. If provided with real energy data from inverter, the theoretical to real energy comparison as well as performance ratio data would be computed and displayed in interactive graphs. In the first version of app design, the inverter data will have to be entered manually by the site owner.

The single page application

Concerning all aspects, the software solution led to the Single Page Application (SPA) design. SPA offer native-like experience while running in the browser. The entire application runs as a single web page bringing two main advantages which were essential for this project (no installation and world wide accessibility). With SPA it is possible to create version of an app, deploy it anywhere in the cloud and provide the end user with the IP address of the cloud machine hosting the app. The end user does not need to install anything except any common web browser. Moreover, after entering credentials at the login page, user specific setting will load no matter which location (and even device) the user connects from.

On the other hand, SPA do not work the same as casual web pages. The main difference is in client-server communication (see Figure 2.1). In classic web appli-

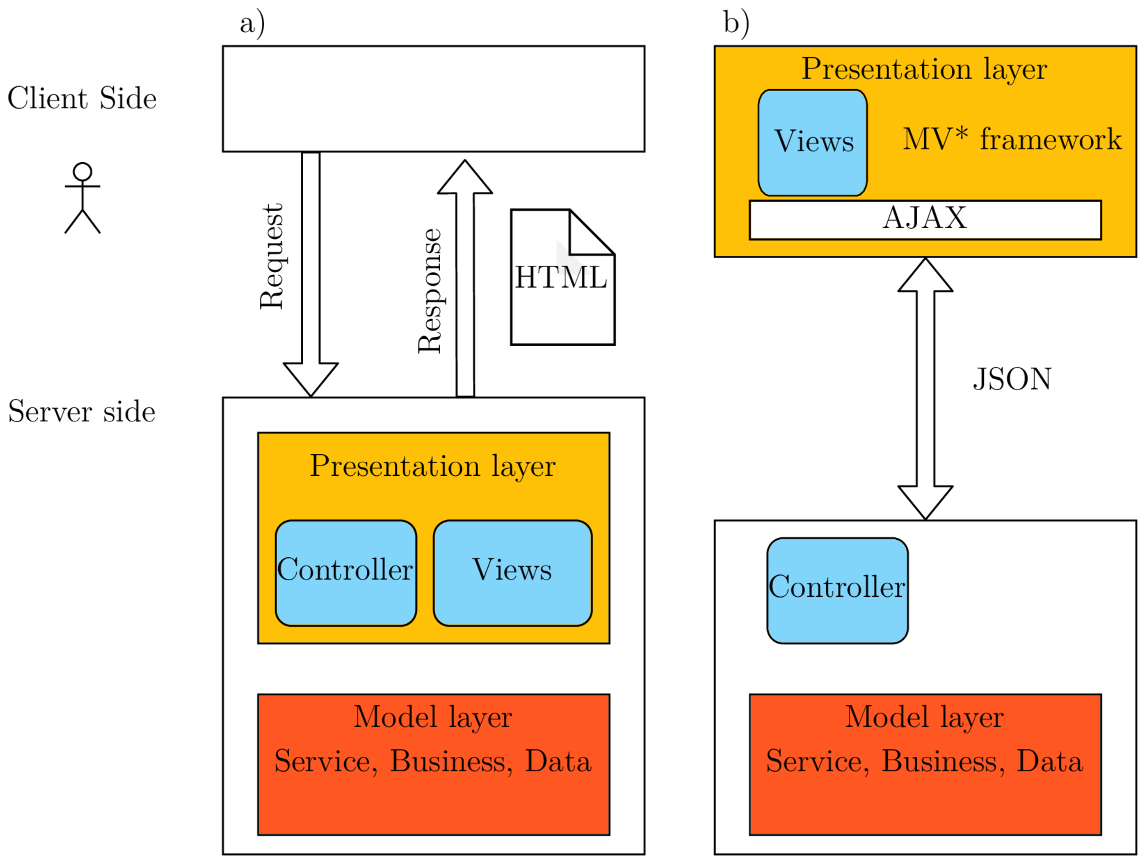


Fig. 2.1: Comparison of SPA (b) to traditional Web application (a). In (a) the server creates whole HTML view and sends it to the client. In (b) only JSON data are exchanged between client and the server the resulting view is constructed at client side (browser) [24].

cation the server assembles the whole HTML page on client request and then all the data are sent back to the client. In SPA the communication between client and the server has the form of JavaScript Object Notation (JSON) and the resulting view is created at client side. When loaded the SPA downloads initial html file (the *s*hell or *t*emplate) which can contain more child *r*egions. When new data are send to the client the resulting view is assembled by javascript MV* framework[3] by replacing the old regions with the new ones. Moreover the Asynchronous JavaScript And XML (AJAX) allows to render the view asynchronously. It means that the

[3]MV stands for Model (data, business logic. . .)and View (visual representation of model data). In the context of the cited book the MV stayed the same for more examples but the rest has changed (e.g. MVC - Model View Controller, MVP - Model View Presenter. . .) thus using the ’*’ placeholder [24].

page does not need to reload during receiving data from server which results in even faster response [24].

In this project React [25] framework was used as an alternative to MV*. Instead of replacing old regions React updates already existing ones [24].

2.1.2 Used Frameworks and languages

Until this point only theory about SPA was provided. However, the information that the application was developed as single page does not say practically anything about the concrete implementation, since there are many options what takes to framework selection. The description will start at the depths of server side leaving the user interface to the end. For clarity, the logos of the frameworks and languages are shown in Figure 2.2.

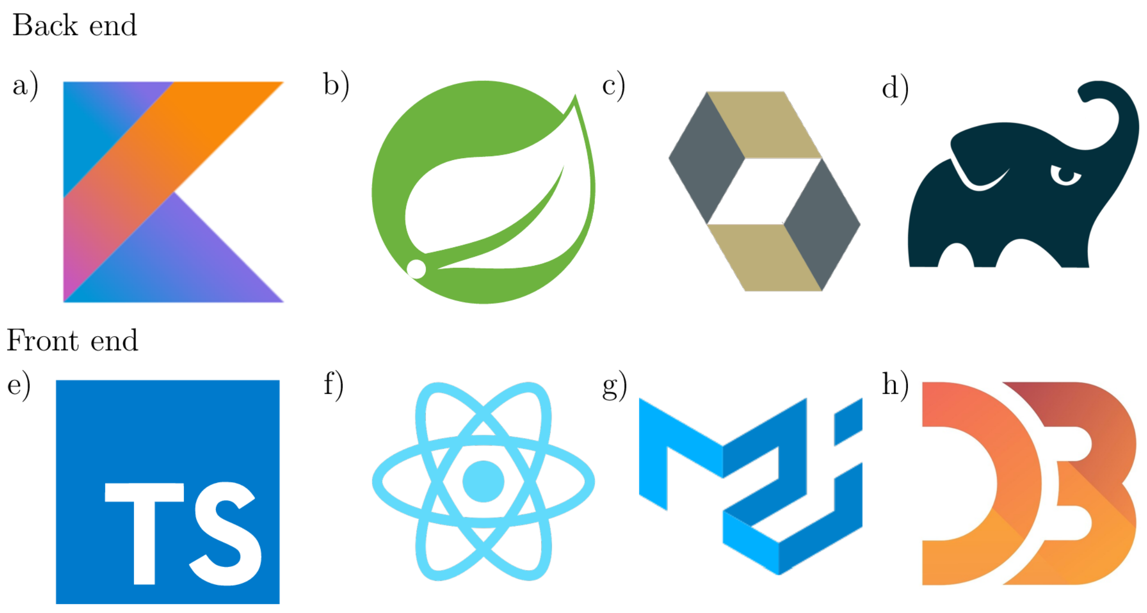


Fig. 2.2: Used Frameworks and languages. (a) Kotlin [26], (b) Spring [27], (c) Hibernate [28], (d) Gradle [29] (e) Typescript [30], (f) React [25], (g) Material-UI [31], (h) D3.js [32].

Kotlin

Kotlin was chosen as the back end (BE) programming language. It is statically typed, general-purpose language developed by JetBrains. The main feature of Kotlin is Java interoperability, which means that it can coexist with Java project and even use Java Frameworks. In other words, Java developers might benefit from smooth learning curve while having modern concise language on their hands. One of very useful properties of Kotlin is without any doubt the type safety which prevents familiar Null pointer exceptions before the code has even been compiled [26].

Spring and Spring Boot

According to the user manual the key element of Spring is support at application level providing the plumbing resources for app development [27]. Spring provides first-class support for Kotlin. In this project the main benefit was seen in dependency injection mechanism which Spring provides. Dependency injection is a pattern which implements Inversion of Control (IoC) principle used in software engineering. IoC enables a framework (Spring) to take control of the flow of the program. It allows to decouple the execution of a task from its implementation, increases program modularity and makes it easier to switch between multiple implementations and code testing. The instantiated and assembled objects managed by Spring IoC container are called beans. The job of the container is to construct dependencies between the beans [33], [34].

Spring Boot is a framework which helps to create stand-alone applications. It comes with embedded Tomcat server and greatly simplifies the process of deploying the application [35].

Hibernate

In majority full stack applications the database access needs to be sooner or later solved. Spring comes with transaction support and one of supported implementations is Hibernate. Hibernate Object Relational Mapping (ORM) is concerned with data persistence as it applies to relational databases using Java Database Connectivity (JDBC) interface. Hibernate is also an implementation of Java Persistence API (JPA) which is a standard for ORM. Using Hibernate enables to develop persistent classes mapping database entities to objects and benefit from idioms such as inheritance, association, composition etc. It is a stable ORM solution with high performance [28].

Gradle

The number of dependencies to external libraries grows with the project and optimal maintenance starts to pose a challenge. Fortunately, open source build automation tool Gradle helps to solve this issue. Gradle brings highly customizable, fast and powerful solution and allows the build scripts to be written in Kotlin DSL[4]. It

[4]Domain-specific Language.

downloads project dependencies, assembles the project and also provides an easy way to build deployable *jar* file [29].

TypeScript

From depths of Server back end the focus is now shifted to front end (FE) view. This project's FE was developed in TypeScript open source language. TypeScript builds on JavaScript which is one of the world's most used tools and adds static type definition. This behavior greatly improves the readability and code safety since the wrongly typed code will not even get compiled. During the build process the TypeScript code is transformed into JavaScript allowing it to run in any Browser [30].

React

React is a JavaScript library for building user interfaces (UIs), but has also support for TypeScript. React helps to create interactive user interfaces by rendering just the components whose state has changed. Only just component-based approach brings a number of benefits. Each component maintains its own state and is reusable. This leads to encapsulation of responsibility. React distinguishes between input data passed to the component from parent components called *props* and data maintained by the component itself called *state*. React also supports modern JSX (or TSX) XML-like syntax which allows to write JavaScript code in the form similar to HTML [25].

Material-UI

Even though it is naturally possible to create own components and style them exactly as the solution demands for every part of user interface, in smaller projects like this one it is convenient to make use of a component library. This approach greatly improves efficiency helping to utilize prefabricated components which can be modified according to specific requirements but leaving the low-level CSS styling aside. Material-UI is one of the most popular component libraries developed by Google. The examples provided can easily be utilized in React based project. Thanks to its popularity it is not difficult to search for the help from the community [31].

D3.js

Since an important aspect of this app was to display sensor measured data graphically to the user in a graph, D3.js library was utilized. It allows to manipulate

Document Object Model (DOM) and apply data driven transformation. It was developed especially to be able to handle large data sets for interaction and animation [32]. An interesting approach is to connect D3 and React letting D3 to generate coordinates for individual parts of the graph curve and then use React for actual chart rendering [36]. This approach combines advantages of both libraries and was also used in this project.

2.2 The Data Model

After clarifying the purpose of the application next logical step is to design the data model thoroughly. This step is very important not only because during the data model creation many design-connected questions are reveled but also that changing the data model during development is an expensive operation, as it usually affects a big part of an application. An overview Entity Relationship Diagram (ERD) will be followed by detail views of closely related entities which will be used as a tool to explain application functionality. It is important to state that not all the entities have already been implemented in the software solution, since the theoretical data model design have naturally outrun the implementation process. All the entities will however be described as they have been theoretically designed and the actual state of the application model will be provided at the end for comparison.

2.2.1 Overview entity relationship diagram

Firstly, an overview ERD is shown in Figure 2.3. It reveals all entities[5] needed for reasonable application functionality. The importance of the diagram stems not only from comprehensive overview of all entities, their mutual relationship types and as a tool for grasping the extent of the application, but it also serves as a signpost. The diagram was divided into areas (marked with different shades of grey) which connect closely related entities and it is these areas which will be described in detail.

Chosen database type is MySQL 5.7 since it runs at the server where the final version of app is aimed to be deployed. In the whole solution only non-identifying relationships were used and the integrity of entities is ensured by ubiquitous primary key. During the design the emphasis was placed on the usage of primitive field types. For numerical values Int and Float were used. For text values NVarChar was used with three possible maximal lengths (15, 30 and 100).

[5]MSS is represented as Instrument entity, as it is composed of Connection box and the Sensor itself.

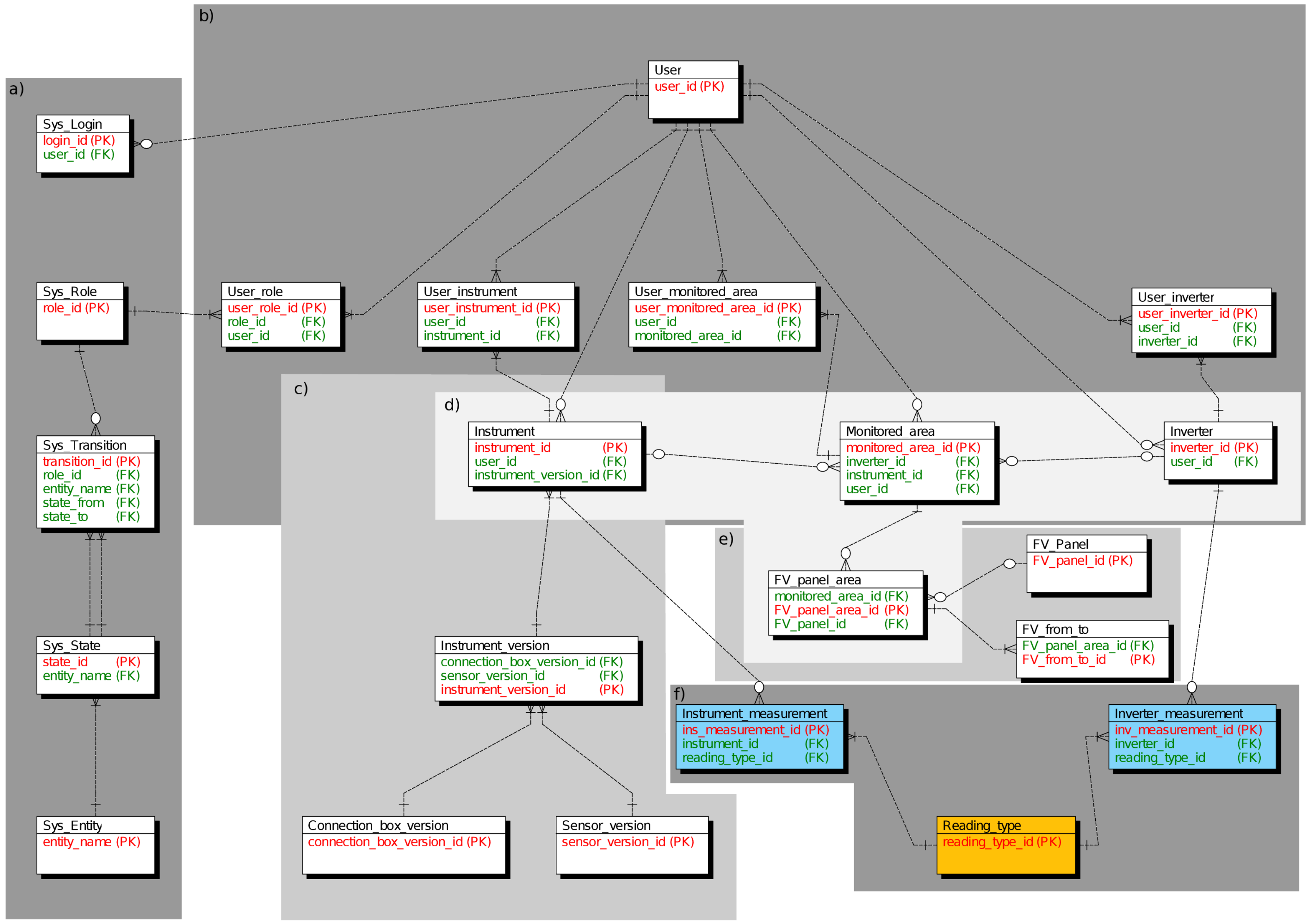


Fig. 2.3: ERD of the whole solution. This overview only shows primary and foreign keys. Detailed ERD views will be provided further according to grey boxes border: (a) System entities, (b) user specific entities, (c) instrument, (d) monitored area, (e) pv panel area, (f) measurements.

2.2.2 System entities ERD

Data model excursion starts with system entities (Fig. 2.4). Their purpose might not be instantly seen as they do not describe the real world objects. `Sys_login` entity serves only as audit-logging mechanism as it only contains date audited information about who logged into the app. This entity should be updated whenever a successful log in attempt occurs.

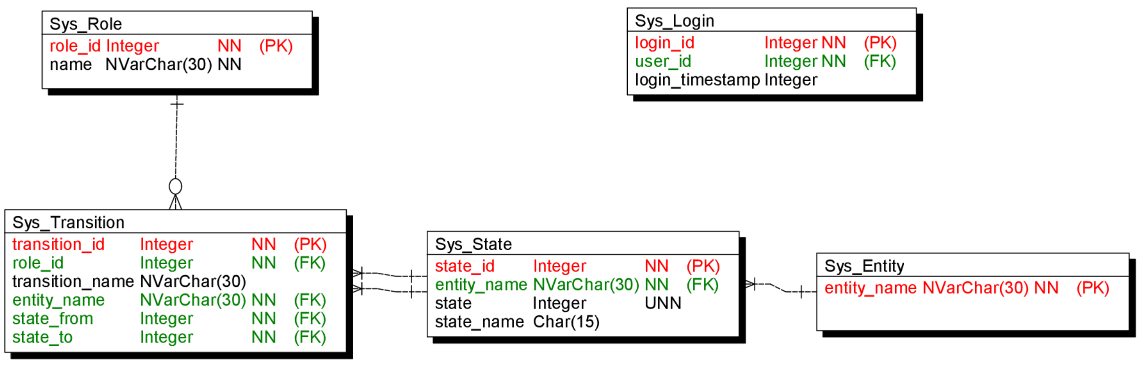


Fig. 2.4: System entities ERD.

The rest system entities take care of some non-system entities state. `Sys_State` maps the state of chosen entity. It contains state number and its description. `Sys_Entity` relates to the entity which state is maintained of. The most complex table is `Sys_Transition` as it covers information about state transitions of entities. Values in `state_from` and `state_to` columns act as foreign keys and relate to the `state_id` bringing the information about initial and terminal state of the transition. Since entity names in one database scheme are unique, `entity_name` also serves as foreign key. `Sys_Role` serves as the translation element between `User_role` table (not visible in this diagram) and system context. Presence of its id in `Sys_Transition` allows to specify which roles can modify the state of selected entity. To make this system work, `state` column must be present in this entity (see `Instrument` and `User` entities in Fig. (2.6).

This state of entities changes according to State Transition Diagrams (STD) which display the states and transition processes between them. STDs for `User` and `Instrument` entity are shown in Figure (2.5). STD comprises of states denoted by circles and state transitions denoted by arrows. Each state transition process is specified by initial and terminal state, condition under which the transition can be performed (e.g. required privileges), and description of the transition process itself. All process specifications for `User` resp. `Instrument` are provided in Tab 2.1 resp. Tab 2.2. Required roles for transition are taken from the Information architecture

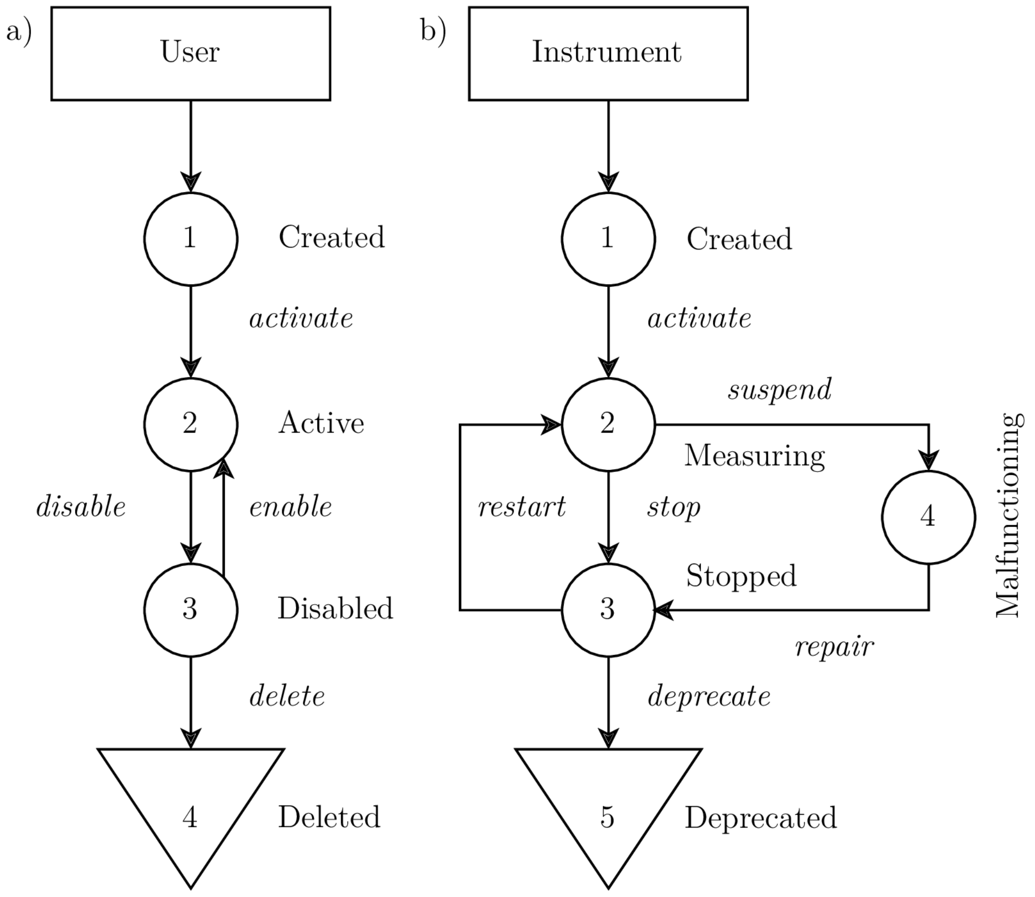


Fig. 2.5: User and instrument STD diagram. The diagram shows States (circles) and transition processes (arrows) of entities (a) User, (b) Instrument. Both entities are put in state Created on creation. Detailed description is provided in Tables 2.1 and 2.2.

section (2.1.1). In ideal case all the transitions are maintained by the user of the app or automatically in order to lower the app maintenance cost. However, in case of Instrument *repair* process the super admin intervention will be most certainly needed, as the malfunctioning state needs to be manually checked.

Tab. 2.1: User STD description according to Figure 2.5a.

Transition name	Initial state	Terminal state	Required role	Automatic/ manual	Description
activate	Created (1)	Active (2)	super admin	a/m	Activates user giving him access to the application. Only active users can log in.
disable	Active (2)	Disabled (3)	site admin	a/m	Disables user due to inactivity of suspicious behavior.
enable	Disabled (3)	Active (2)	site admin	a/m	Enables disabled user if reason of his disablement no longer prevails.
delete	Disabled (3)	Deleted (4)	super admin	a/m	Deletes user. This account will never be restored. Processed manually or automatically after delay.

Tab. 2.2: Instrument STD description according to Figure 2.5b.

Transition name	Initial state	Terminal state	Required role	Automatic/ manual	Description
activate	Created (1)	Measuring (2)	site admin	m	Activates created instrument. Only instrument in Measuring state logs data into the database.
stop	Measuring (2)	Stopped (3)	site admin	m	Stops instruments measurement.
suspend	Measuring (2)	Malfunctioning (4)	site admin	a/m	Suspends measurement. Similar to stop but this transition is rather used when unexpected action occurs (e.g. wrong data logged).
restart	Stopped (3)	Measuring (2)	site admin	m	Restarts stopped measurement.
repair	Malfunctioning (4)	Stopped (3)	super admin	m	Switches state to stop if the reason which caused the instrument to malfunction has been repaired. When the instrument can not be repaired, deprecate process will immediately follow
deprecate	Stopped (3)	Deprecated (5)	super admin	a/m	Deprecates instruments which cannot be repaired or which are no longer supported.

2.2.3 User specific entities ERD

The entity which stands in the middle of the whole data model is `User`. This entity represents concrete people who have access into the app. The unique constraints are applied to the `username` and `email` fields which means that the app will not allow to register two users with same `username` or `email`. The app user represented by `User` entity controls his `Instrument`, `Inverter` and `Monitored_area` entities. There is always only one user per one household who 'owns' the site comprising MSS, inverter, and PV panels. This user has site admin role and maintains all 1:N relationships with `Instrument`, `Inverter` and `Monitored_area`. In order to allow family members to access the same entities the site admin can give and maintain access to his property with the use of M:N associative entities `User_instrument`, `User_inverter` and `User_monitored_area`. As the title suggests, the `User_role` serves as an associative entity mapping M:N relationship between `User` and `Sys_Role`, as it associates users with their roles. If not maintained automatically, only user with super admin role can modify this entity. The details of `Instrument`, `Inverter` and `Monitored_area` entities will be provided in next Section (2.2.4) where the whole concept is described in detail.

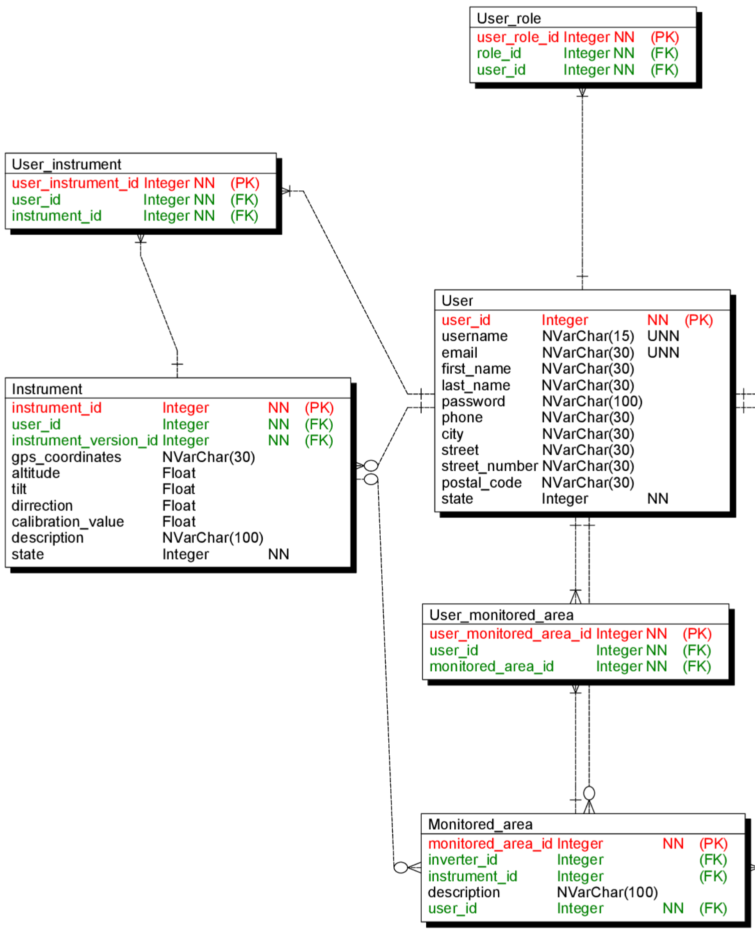


Fig. 2.6: User specific entities ERD.

2.2.4 Monitored area concept with ERD

This section represents the core of the model design as it tries to map the real life scenario to the database entities. For the sake of clarity, the monitored area concept is introduced (see Figure 2.7) before diving into details of ERD. The monitored area

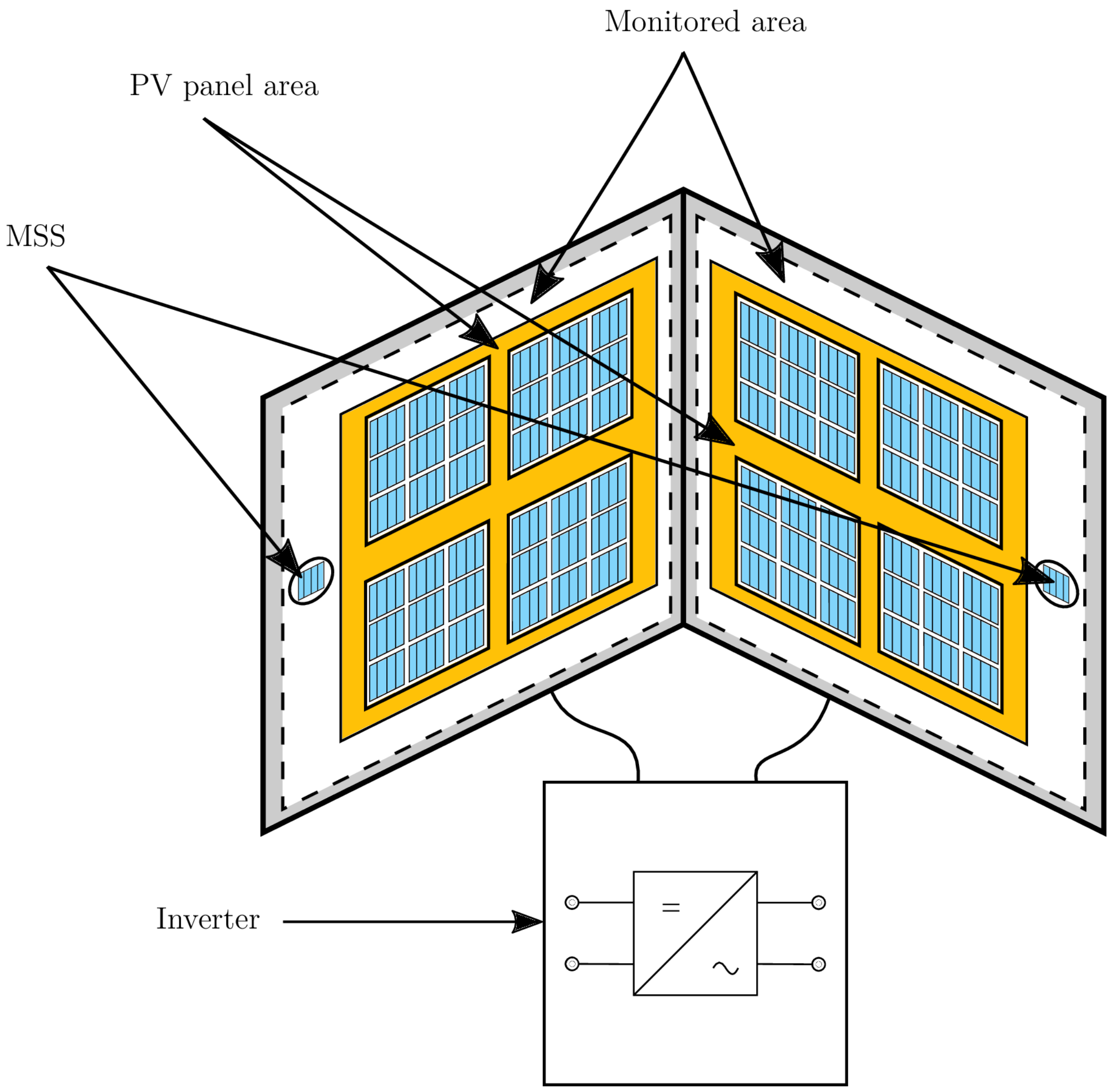


Fig. 2.7: Description of monitored area concept. The picture shows a roof of a house. The white area with dashed boundary represents monitored area which wraps around PV panels, MSS and inverter connection

is a part of a house which packs up MSS, PV panels and inverter connection together. Introduction of monitored areas allows to divide PV panels regions in such a way that individual as well as an overview energy output of these regions can be computed. Connection to inverter further provides the possibility to compare monitored area theoretical energy to inverter-measured real energy. From the theoretical-to-real energy comparison performance ratio can be computed (see Section 1.2.3 for theoretical description).

ERD diagram (Fig. 2.8) clearly shows the mutual relationships between `Monitored_area` and its connected `Instrument`, `Inverter` and `PV_panel_area` entities. All these entities have its owner i.e. site admin. In order to preserve freedom in site-specific design, monitored area can comprise zero or one of each instrument/inverter. The same applies from the other side of relationship, instrument and inverter can monitor zero to many monitored areas. Before the real case scenario examples are provided, the entities fields will be explained.

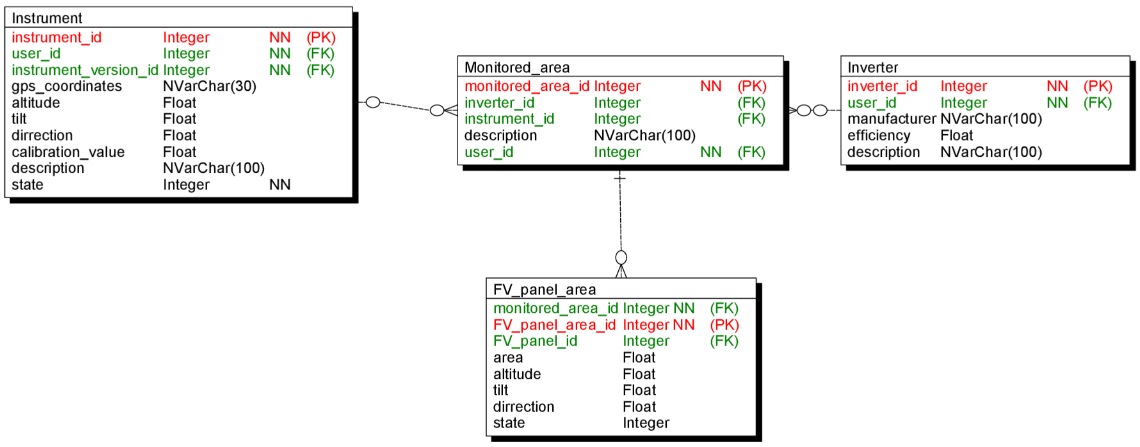

Instrument
instrument_id Integer NN (PK)
user_id Integer NN (FK)
instrument_version_id Integer NN (FK)
gps_coordinates NVarChar(30)
altitude Float
tilt Float
dirrection Float
calibration_value Float
description NVarChar(100)
state Integer NN
Monitored_area
monitored_area_id Integer NN (PK)
inverter_id Integer (FK)
instrument_id Integer (FK)
description NVarChar(100)
user_id Integer NN (FK)
Inverter
inverter_id Integer NN (PK)
user_id Integer NN (FK)
manufacturer NVarChar(100)
efficiency Float
description NVarChar(100)
FV_panel_area
monitored_area_id Integer NN (FK)
FV_panel_area_id Integer NN (PK)
FV_panel_id Integer (FK)
area Float
altitude Float
tilt Float
dirrection Float
state Integer

Fig. 2.8: Monitored area ERD.

`Monitored_area` entity serves as an association entity for M:N relationships. Aside from foreign keys of these entities, only string `description` filed is present. It help the end-user to orientate among personal areas. The same field is included in both `Instrument` and `Inverter`.

`Instrument` represents MSS or 'the sensor'. Fields `gps_coordinates` and `altitude` provide information about sensor location. `tilt` and `direction` directly influence any photodiode output so the tilt and direction of the sensor and PV panels should be the same (Section 1.1.2). From these and also calibration reasons, it is important to store these information about each sensor. As already described (Section 1.3.1), the sensor log values linearly dependent on the irradiance thus it must comprise `calibration_value` for recalculation. `Instrument` can be in different states (Section 2.2.2). Some additional information about the entity will be given in Section 2.2.5.

`PV_panel_area` models PV panel regions inside monitored area. They can not exist without parent `Monitored area`. Ideally, the `tilt` and `direction` are the same as paired sensor. In order to support for example the case of a house using one sensor for monitoring two differently tilted PV panel areas (Fig. 2.9c), these fields have to be present[6]. The field `area` is the numerical value representing dimensions of PV panels in PV panel area. More information will be provided in Section 2.2.6.

`Inverter` represents user inverter. None of the fields are in the to-date version of the app. But `Inverter` entity plays an important role when logging real energy measurement data.

Supported configurations

Up to this stage only general entities and their mutual relationships have been described. This section brings supported real world implementation examples. The corner stone of the description is the Figure 2.9. It provides both pictures of real world scenario and its corresponding ERD.

Part (a) should represent the most widespread configuration. Every part of the roof with PV panels is wrapped by its own monitored area with unique MSS. Moreover all monitored areas are connected to unique inverter. When thinking this case more deeply this would mean that PV plant owners would most likely need to have more inverters in one house. In reality, the house has only one inverter with more inputs, so energy from west and south part of the roof would be handled separately. For the purpose of model each input can however act as unique inverter.

Configuration in (b) should also be used widely. The owner of the house buys MSS for each part of the roof. The resulting configuration is more monitored areas connected to one inverter which then provides overall real energy values of the whole house.

In (c) the compromise configuration is depicted. As already many times stated, in order to provide proper PV panels monitoring the tilt and direction of the MSS and the panels should match. However, there might also be use case when part of the roof is just a little differently oriented and it does not pay off to mount separate MSS. Since the inverter has information of all the energy and no connected parts can be excluded there should be a way how to support this real case scenario. However, this would require corrections in energy output which are not implemented in the app yet.

The list of supported configurations ends with (d). This configuration is not expected to be used often. It describes the scenario when two people live next to each other in terraced houses ideally with the same roof orientation. Then the

[6]In described configuration the computation corrections will have to be implemented and are not yet part of the app. These values however will allow this functionality in the future.

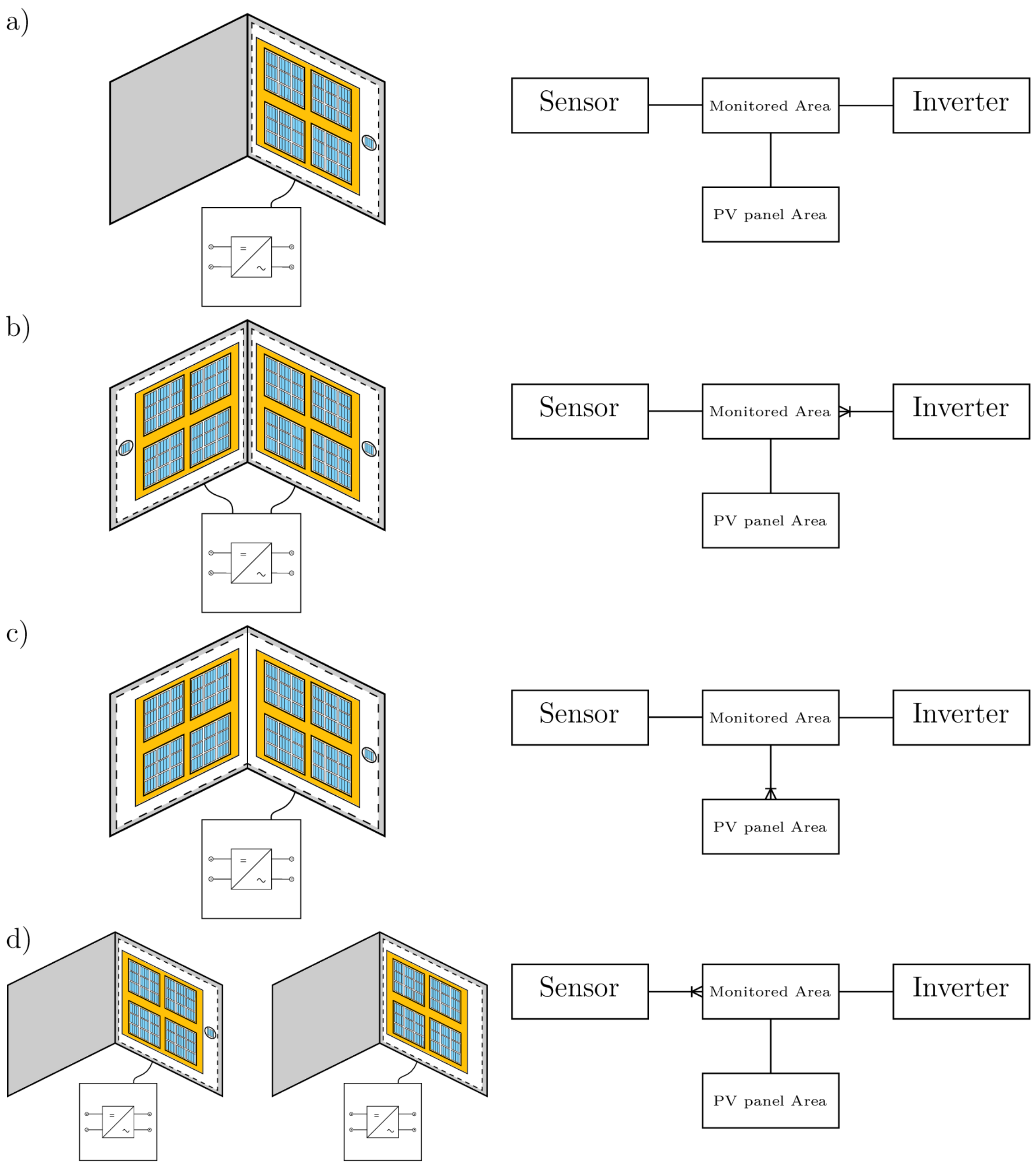
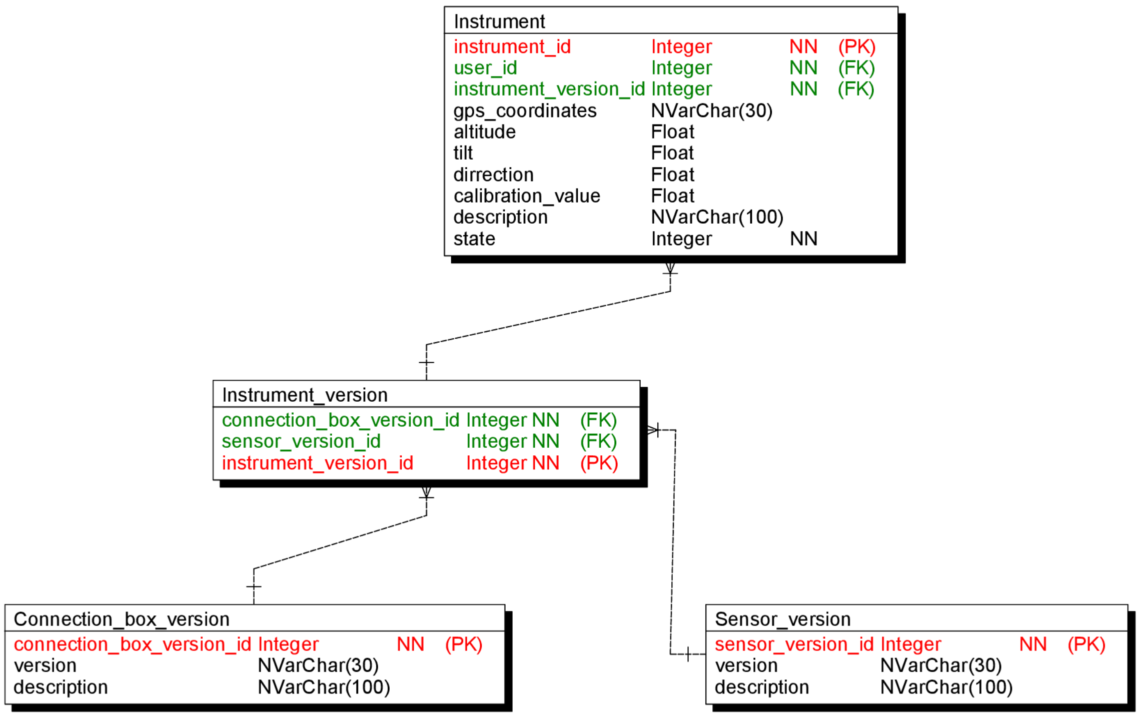


Fig. 2.9: Supported configurations. The figure compares real world scenarios (left) to corresponding ERD (right). (a) One monitored area connected to one MSS and inverter. (b) The roof divided into more monitored areas all with their own MSS connected to one shared inverter. (c) Monitored area contains more differently tilted PV panel areas, all monitored by one MSS. (d) One MSS monitors more areas which are all connected to their own inverter.

neighbor owning MSS can provide the other one with expected irradiance. The non-owner just uses his house specific monitored area which may differ in the surface area or PV panel type. Each of the neighbors have information of real energy output from their own inverters.

2.2.5 Instrument ERD

Since the fields of `Instrument` entity have already been explained (Sec. 2.2.4), Figure 2.10 only adds the information about MSS versioning. Each `Instrument` (MSS) comprises connection box and sensor. Both together carry information about instrument version. This approach ensures modularity. When for example the connection box version is replaced with LoRa module, the `Sensor_version` can stay the same while the change in `Connection_box_version` will impact `Instrument_version`.



Instrument
instrument_id Integer NN (PK)
user_id Integer NN (FK)
instrument_version_id Integer NN (FK)
gps_coordinates NVarChar(30)
altitude Float
tilt Float
dirrection Float
calibration_value Float
description NVarChar(100)
state Integer NN
Instrument_version
connection_box_version_id Integer NN (FK)
sensor_version_id Integer NN (FK)
instrument_version_id Integer NN (PK)
Connection_box_version
connection_box_version_id Integer NN (PK)
version NVarChar(30)
description NVarChar(100)
Sensor_version
sensor_version_id Integer NN (PK)
version NVarChar(30)
description NVarChar(100)

Fig. 2.10: Instrument ERD.

2.2.6 PV panel ERD

`PV_panel_area` was also described in Sec. 2.2.4. This ERD brings the details about `PV_panel`. The idea behind is that `PV_panel` will be shared between all households, since the types will most likely repeat. The important quantity for energy output configuration is `efficiency`. The other `PV_from_to` entity serves to the purpose of date auditing. In other words, the computation of the output energy has to take the changes of PV panel areas throughout time. The scenario of an user buying extra panels after some time is probable. This behavior is supported by `state` in which the panel area was found between the `from` and `to` timestamps.

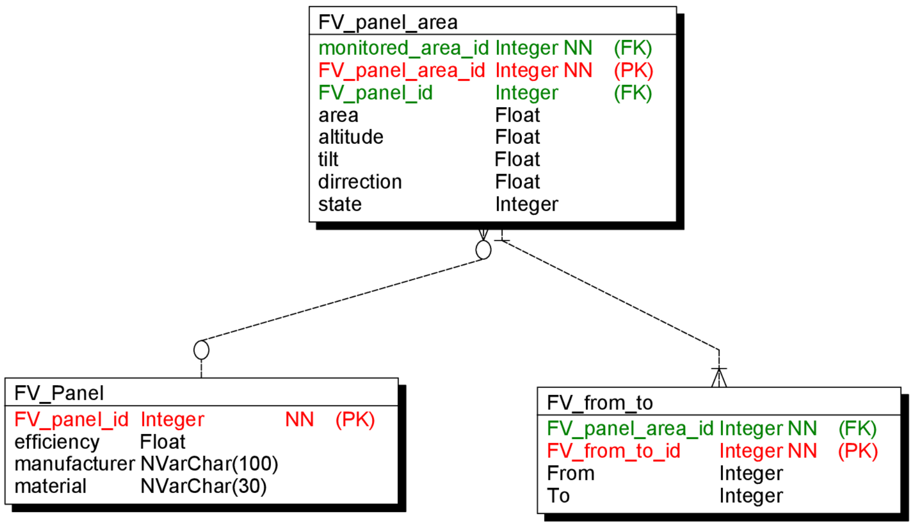

FV_panel_area
monitored_area_id Integer NN (FK)
FV_panel_area_id Integer NN (PK)
FV_panel_id Integer (FK)
area Float
altitude Float
tilt Float
dirrection Float
state Integer
FV_Panel
FV_panel_id Integer NN (PK)
efficiency Float
manufacturer NVarChar(100)
material NVarChar(30)
FV_from_to
FV_panel_area_id Integer NN (FK)
FV_from_to_id Integer NN (PK)
From Integer
To Integer

Fig. 2.11: PV panel ERD.

2.2.7 Inverter and Instrument measurement ERD

The tables in the last ERD will certainly contain the biggest amount of data since `Instrument_measurement` and `Inverter_measurement` tables store all data measured. Each read measurement value contains `utc_timestamp`. The `Reading_type` entity carries information about the type of reading. Each instrument measurement entry actually contains 8 different readings (Sec. 1.3.3), inverter reading brings the ninth type. This approach ensures the possibility to add more reading types or deprecate some unused without changes in the data model.

2.2.8 To-date implemented model

Not all entities described in previous sections are essential to the proper functioning of the app. Some just make its usage smoother. Even though all entities play their role and definitely should stay in the theoretical design model, following list provides comparison of the extent of to-date implemented model to the theoretical one.

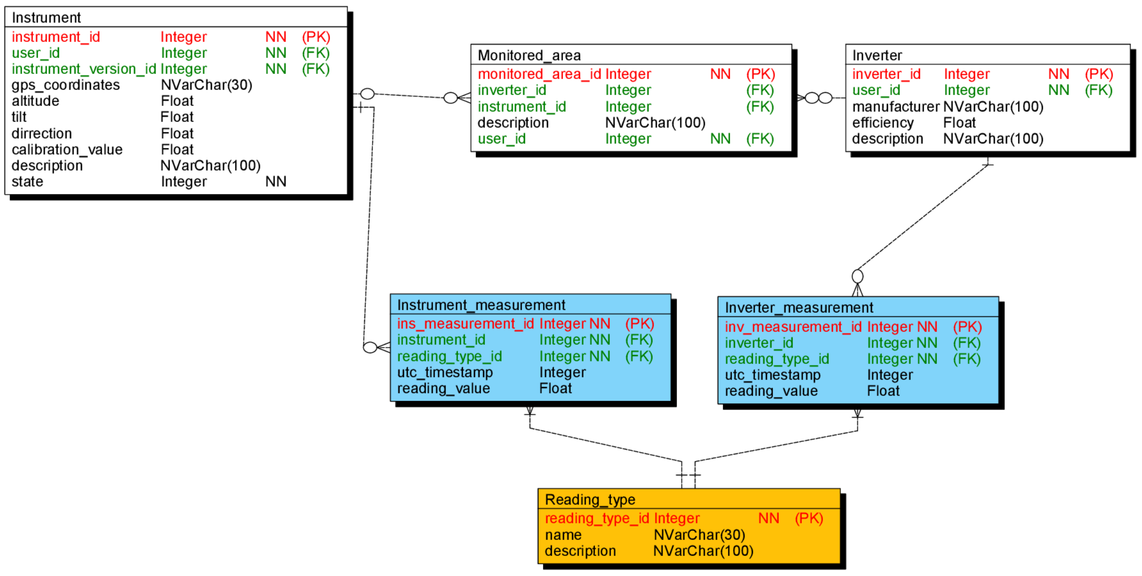

Instrument
instrument_id Integer NN (PK)
user_id Integer NN (FK)
instrument_version_id Integer NN (FK)
gps_coordinates NVarChar(30)
altitude Float
tilt Float
dirrection Float
calibration_value Float
description NVarChar(100)
state Integer NN
Monitored_area
monitored_area_id Integer NN (PK)
inverter_id Integer (FK)
instrument_id Integer (FK)
description NVarChar(100)
user_id Integer NN (FK)
Inverter
inverter_id Integer NN (PK)
user_id Integer NN (FK)
manufacturer NVarChar(100)
efficiency Float
description NVarChar(100)
Instrument_measurement
ins_measurement_id Integer NN (PK)
instrument_id Integer NN (FK)
reading_type_id Integer NN (FK)
utc_timestamp Integer
reading_value Float
Inverter_measurement
inv_measurement_id Integer NN (PK)
inverter_id Integer NN (FK)
reading_type_id Integer NN (FK)
utc_timestamp Integer
reading_value Float
Reading_type
reading_type_id Integer NN (PK)
name NVarChar(30)
description NVarChar(100)

Fig. 2.12: Inverter and Instrument measurement ERD.

System entities

`Sys_Role` - implemented with two roles: user and admin

`Sys_Transition` - part of db schema but not yet used

`Sys_State` - part of db schema but not yet used

`Sys_Entity` - part of db schema but not yet used

`Sys_Login` - not implemented yet, these data are logged into audit log

User specific entities

`User_role` - implemented in full extent

`User` - implemented without selected fields e.g address-connected, state

`User_instrument` - not implemented yet

`User_inverter` - not implemented yet

`User_monitored_area` - not implemented yet

Monitored area entities

`Inverter` - fully implemented

`Monitored_area` - fully implemented

`Instrument` - almost fully implemented only the state is not maintained by Sys entities

`PV_panel_area` - implemented without state

Instrument entity

`Instrument_version` - not implemented yet
`Connection_box_version` - not implemented yet
`Sensor_version` - not implemented yet

PV panel entities

`PV_panel` - fully implemented
`PV_from_to` - not implemented yet

Inverter and instrument entities

`Instrument_measurement` - fully implemented
`Inverter_measurement` - fully implemented
`Reading_type` - fully implemented

2.3 Programming the model

2.3.1 Transferring model into database

Designing functional theoretical model is without any doubt important step when building an application. Another step of the same importance is to transfer this model into real database. The use of frameworks (Sec. 2.1.2) make this process easier but cautious approach still needs to be taken.

An example of `Instrument` entity definition is shown in Listing (2.1). The center of interest is mapping the database entity to programmatically defined structure i.e. class. On line 3 the definition of such a class starts. The data class is special type of class tailored to purpose of holding data. It automatically defines standard functionality derivable from data[26]. Kotlin further allows to define properties of a class in primary constructor. The construction of a data class is apparent from the listing. All the properties are defined by the `var` keyword followed by property name[7] and type specification behind colon.

Until now the construction of basic data class not anyhow connected to the database entity was described, it is the annotations that define the mapping [37].

`@Entity` - Specifies that the class is entity.
`@Table (name = "Instrument")` - Specifies primary database table for entity; parameter `name` allows to specify the mapped table name.
`@Id` - Specifies primary key of an entity.

[7]For example id (line 6), gpsCoordinates (line 10)...

`@Column(name = "gps_coordinates")` - Specifies the mapped column for a persistent property or field. Parameter `name` allows to specify the mapped column name. If not provided, the name of the property is used as column name.

Further, the field constraints can also be defined by annotations [38].

`@NotBlank` - the annotated element must not be null and must contain at least one non-whitespace character.

`@Size (max = MEDIUM_STRING_FIELD_LENGTH)` - The annotated element size must be between the specified boundaries. `MEDIUM_STRING_FIELD_LENGTH` is this project defined constant.

Two important annotations have been omitted from description on purpose as they define the mutual entities relationships. Referring to Figure 2.3 and up-to-date implemented model (Sec. 2.2.8) the `Instrument` entity has M:1 relationship with `User` entity and 1:N relationships with `Monitored_area` and `Instrument_measurement`. The link to `User` is realized through `user` property which is type of `User`[8] (line 29). The `@ManyToOne` annotation specifies the expected arity and `@JoinColumn(name = "user_id")` defines the column in `Instrument` table under which the foreign key will be mapped. In similar manner the `@OneToMany` annotation specifies the other side of relationship (e.g. line 31, line 33). In this case, the `Instrument` has property `instrumentMonitoredAreas` which is a set of `MonitoredAreas`[9]. This set carries information about the number of monitored areas which are currently linked to this `Instrument` instance. The `Instrument` class needs to contain methods of adding (line 35) and removing (line 47) a `MonitoredArea` from the relationship.

When the model is correctly defined via annotated classes and needed settings are set in application properties file, the Hibernate framework takes care of executing required DDL[10] statements to the database creating required schema.

2.3.2 Querying the database

After the model has been established the way of executing queries through DML[11] needs to be found. Programmatically this can be achieved via implementing repositories (Listing 2.2).

[8]The name of the data class mapping User entity.

[9]The name of the data class mapping Monitored_area entity.

[10]Data Definition Language deals with database schemas and and descriptions (e.g. statements CREATE, ALTER. . .) [39]

[11]Data Manipulation Language deals with data manipulation and includes most SQL statements (e.g. SELECT, INSERT. . .) [39]

Listing 2.1: Programatically defined Instrument entity

```kotlin
@Entity
@Table(name = "Instrument")
data class Instrument(
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        var id: Long = 0,
        @NotBlank
        @Size(max = MEDIUM_STRING_FIELD_LENGTH)
        @Column(name = "gps_coordinates")
        var gpsCoordinates: String,
        @NotBlank
        @Column(unique = true)
        var guid: String,
        @NotBlank
        var altitude: Float,
        @NotBlank
        var tilt: Float,
        @NotBlank
        var direction: Float,
        @NotBlank
        @Column(name = "calibration_value")
        var calibrationValue: Float,
        @Enumerated(EnumType.STRING)
        @Column(length = SHORT_STRING_FIELD_LENGTH)
        var state: InstrumentState,
        @NotBlank
        @ManyToOne
        @JoinColumn(name = "user_id")
        var user: User
) {
    @OneToMany(mappedBy = "instrument", cascade = [CascadeType.ALL],
        orphanRemoval = false)
    var instrumentMonitoredAreas: MutableSet<MonitoredArea> = mutableSetOf()
    @OneToMany(mappedBy = "instrument", cascade = [CascadeType.ALL],
        orphanRemoval = false, fetch = FetchType.LAZY)
    var instrumentMeasurements: MutableSet<InstrumentMeasurement> =
        mutableSetOf()
    fun addMonitoredArea(monitoredArea: MonitoredArea) {
        val instrumentMonitoredArea = MonitoredArea(
                id = monitoredArea.id,
                instrument = this,
                user = monitoredArea.user,
                inverter = monitoredArea.inverter,
                description = monitoredArea.description
        )
        instrumentMonitoredArea.monitoredAreaPvPanelAreas = monitoredArea.
            monitoredAreaPvPanelAreas
        instrumentMonitoredAreas.add(instrumentMonitoredArea)
    }
    fun removeMonitoredArea(monitoredArea: MonitoredArea): MonitoredArea {
        instrumentMonitoredAreas.remove(monitoredArea)
        return monitoredArea
    }
    // some rows skipped
}
enum class InstrumentState {
    CREATED, MEASURING, STOPPED, MALFUNCTIONING, DEPRECATED
}
```

Listing 2.2: Instrument repository

```
1 interface InstrumentRepository: JpaRepository<Instrument, Long> {
2     fun findInstrumentById (id: Long): Instrument?
3     fun findByUser(user: User): MutableSet<Instrument>
4     fun existsByGuid(guid: String): Boolean
5     fun findByGuid(guid: String): Instrument?
6 }
```

`InstrumentRepository` is an interface which implements `JpaRepository` from Spring data JPA. This framework adds an extra abstraction layer on the top of JPA provider (in this case Hibernate) helping to significantly reduce boilerplate code needed to implement data access layer [40]. The simplification is indeed apparent as the only function which needs to be defined when it is desired to retrieve an `Instrument` of a given id is `findInstrumentById` (line 2). The frameworks take care of database access returning desired data class.

2.4 Payload and REST communication

This section describes the interlayer between the client and the server. The request-response pattern follows REST[12] architecture.

2.4.1 Spring REST controller

In Spring the way of exposing REST endpoints of an application is to create a `@RestController` annotated class (Listing 2.3). Annotation `@RequestMapping` (line 2) defines the url which will be exposed. Http method (e.g. GET, POST, PUT…) is defined (line 6) via `@PostMapping` and the request body by `@RequestBody` annotation of `instrument:InstrumentDto` argument (line 10). The controller acts only as an entry point to the application thus no logic is implemented in associated function `addInstrument` (line 8) as it calls the same-named function on `InstrumentService` api (line 12). In case of successfully persisting the Instrument into database, the controller will return status code of 201 CREATED, whereas unsuccessful operation is followed by 400 BAD REQUEST code. The example of a Data Transfer Object[42] (DTO) used in `InstrumentController` is shown in Listing (2.4). Comparing to `Instrument` entity (Lst. 2.1) the properties are very similar.

[12]Representational State Transfer [41].

Listing 2.3: Instrument controller

```
@RestController
@RequestMapping("/api/instruments")
class InstrumentController(
        private val service: InstrumentService
) {
    @PostMapping
    @PreAuthorize("hasRole('USER')")
    fun addInstrument(
            @CurrentUser currentUser: UserPrincipal,
            @Valid @RequestBody instrument: InstrumentDto
    ): ResponseEntity<ApiResponse> {
        return when (val result = service.addInstrument(currentUser,
            instrument)) {
            is InstrumentServiceOperationResult.Success ->
                buildCreatedResponseEntity("/api/instrument/{guid}",
                result.value)
            is InstrumentServiceOperationResult.Error -> ResponseEntity
                (ApiResponse(false, "Sensor␣could␣not␣have␣been␣created
                "), HttpStatus.BAD_REQUEST)
        }
    }
    // rows omitted
}
```

Listing 2.4: Instrument DTO

```
data class InstrumentDto(
        val id: Long,
        val state: InstrumentState,
        val guid: String,
        val gpsCoordinates: String,
        val altitude: Float,
        val tilt: Float,
        val direction: Float,
        val calibrationValue: Float,
        val monitoredAreas: Set<InsInvMonitoredAreaDto>
)
```

Listing 2.5: Controller error handling

```
@ControllerAdvice
class RestResponseEntityExceptionHandler : ResponseEntityExceptionHandler() {
    // rows omitted
    @ExceptionHandler(value = [MonitoredAreaWithoutPvAreaException::class])
    fun handleMonitoredAreaWithoutPvAreaException(ex:
        MonitoredAreaWithoutPvAreaException): ResponseEntity<ErrorDto> {
        logger.error(ex.message, ex)
        return ErrorDtoResponseEntityBuilder(ExceptionType.
            MonitoredAreaWithoutPvAreaException)
                .withMessage(ex.message)
                .withDetail(ex.monitoredAreaId)
                .buildResponseEntity()
    }
    // rows omitted
}
```

2.4.2 Error handling

When an unchecked exception occurs, the server returns status code 500-Internal Server Error. These responses are generally not wanted since they tell that something unexpected has happened on the server side.

Fortunately, the Spring framework comes with the solution once again as it allows to define `ResponseEntityExceptionHandler` which provides centralized exception handling across all `RequestMapping` methods. The example of handling a custom exception is provided in Listing 2.5. If custom made

`MonitoredAreaWithoutPvAreaException`

is thrown anywhere in the application and its handling mechanism is implemented in exception handler, then the controller returns defined response.

2.5 Securing the application with login

In order to develop a full stack application allowing users to access their personal data, the authentication methods must be introduced. The approach taken in this project was username-password authentication based on Json Web Token[43] (JWT). Since the description of security solution with code examples would take unnecessary amount of space and is not the point of interest in this project a simplified approach referring to flow chart diagram was preferred.

2.5.1 The authentication process

In Figure (2.13) the authentication process is depicted. The client i.e. the app user sends HTTP request to the controller e.g. by clicking some button in the application

GUI[13]. Every request made to any app endpoint firstly hits the `JwtAuthenticationFilter` which tries to get the authentication token from the request header. If the operation is not successful (token is `null`) the authentication process is skipped proceeding with execution of `Controller` method. However, if `getJwtFromRequest` returns parsed token in the `String` form `JwtTokenProvider` is called in order to validate the token. If invalid, the process again proceeds to the `Controller`. Otherwise, the token claims are parsed and from them the user id is retrieved. This is serves for finding the user in the database and is executed via `CustomUserDetailsService` which then accesses the database through `UserRepository`. Custom made class `UserPrincipal` implements `UserDetails` interface and its `create()` method returns user details among others also the user role. These details are understood by Spring and created authentication token via `UsernamePasswordAuthenticationToken` can be then set into the Spring security context via `SecurityContextHolder`. Only then the authentication process is completed and `Controller` method can proceed with execution. The last part in the flowchart is `JwtAuthenticationEntryPoint`. This class starts to play the role if the user is trying to access the resource which he is not authorized for. Then the class takes care that HTTP response code 401 Unauthorized is returned. However, the app exposes also some resources for which the authentication is not required (e.g. `/signin` endpoint). In this case authentication is ignored and no 401 is returned.

2.5.2 Spring method level security

The whole security configuration happens in `ApplicationSecurityConfiguration` class which extends Spring security's `WebSecurityConfigurerAdapter` where resources demanding authentication are defined. Any controller method which needs authentication can be then secured by adding `@PreAuthorize` annotation (see Listing 2.3, line 7). Very simplifying step is to define custom `@CurrentUser` annotation which wrapped around Spring's `@AuthenticationPrincipal` allows to access currently authenticated user in the controllers (line 9).

At this place the credit has to be given to awesome Spring Security tutorial by Rajeev Singh who very understandably describes this issue at [44].

2.6 Graphical user interface

Previous sections were about application back end. It is the REST controller which allows interactions between the server and the client. This section focuses on GUI through which these interactions may be controlled.

[13]Graphical user interface.

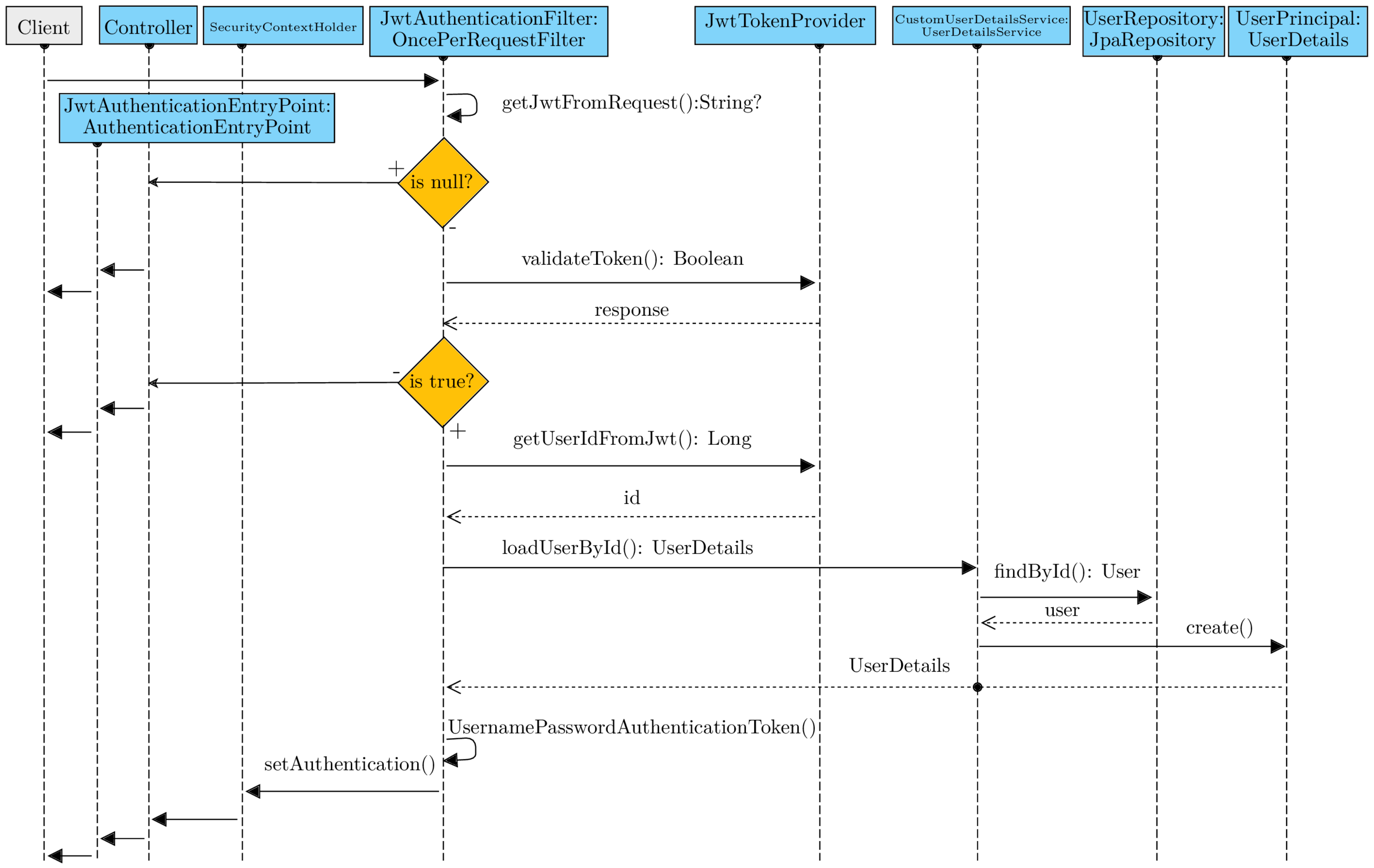

Client
Controller
SecurityContextHolder
JwtAuthenticationFilter: OncePerRequestFilter
JwtTokenProvider
CustomUserDetailsService: UserDetailsService
UserRepository: JpaRepository
UserPrincipal: UserDetails
JwtAuthenticationEntryPoint: AuthenticationEntryPoint
getJwtFromRequest():String?
is null?
+
-
validateToken(): Boolean
response
is true?
-
+
getUserIdFromJwt(): Long
id
loadUserById(): UserDetails
findById(): User
user
create()
UserDetails
UsernamePasswordAuthenticationToken()
setAuthentication()

Fig. 2.13: Flowchart of authentication process in the application. Blue rectangles represent classes and interfaces from which they inherit (behind colon). Dashed line represents time flow. Arrows show function calls (the parameters are omitted).

2.6.1 Create React App

First of all, the front end development (dev) environment will be briefly described. Without any doubt there are many tools which simplify the development. In this project Create React App was used [45]. This approach is very straightforward as it only requires to have Node Package Manager [46] (npm) in version ≥ 8.10 installed on the dev machine. To create a new app execution of this code block is required

```
npm init react-app my-app --template typescript
```

This command also introduces TypeScript into the project. After the project is created the skeleton of a React App can be accessed by default on `http://localhost:3000` when running

```
npm start
```

The app running at given url updates automatically when the source code has changed. After the development process has ended the production build may be executed by

```
npm run build
```

This process will produce JavaScript and CSS files inside `build/static` directory.

2.6.2 React-Spring integration

When properly configured, the Spring application runs on its own Tomcat server accessible for example on `http://localhost:8080`. Very simply put, in order to have Spring displaying the content of React bundle, it expects that all the resources have been put into `resources/static` folder. This can easily be ensured by copying the content of Create React App production build i.e. `build/static`. Executing this step allows the web browser to load React app when accessing `http://localhost:8080`.

2.6.3 Client-server communication

This section introduces methods of accessing resources exposed by REST controller (Sec. 2.4.1). In this project all the communication between client and the server happens through REST api. Every controller endpoint, HTTP method and required body type has to match in order the interaction could be successful. An example of a request used for new instrument creation is shown in Listing (2.6). The lines 2-9 show request body type (line 2), url (line 5), method (line 6) (compare to Listing 2.3). The `InstrumentDto` type is stored in different file but shown here in the same listing (compare to Listing 2.4).

Listing 2.6: New instrument request

```
// ApiRequestStore.ts
createNewInstrument = (body: InstrumentDto): Promise<ApiResponse> =>
    this.createRequest<ApiResponse>(
        {
            url: `${BASE_URL}/api/instruments`,
            method: "POST",
            body: JSON.stringify(body)
        }
    );
// RequestBodyData.ts
export type InstrumentDto = {
    id: number,
    state: InstrumentState
    guid: string ,
    gpsCoordinates: string ,
    altitude: number,
    tilt: number,
    direction: number,
    calibrationValue: number,
    monitoredAreas: InsInvMonitoredAreaDto[]
}
```

2.6.4 App.tsx

The description will start with the topmost element in the component hierarchy - `App.tsx`. This three letter React component actually encapsulates the whole front end code and is rendered directly into `index.tsx`. Its responsibilities are top level routing, logging the user in and out, redirecting non-authenticated user back to login page and handling general error dialogues. The application `MuiTheme` is also specified here.

2.6.5 The Main page

Another high level component is `MainPage.tsx`. This is the place reserved only for authenticated user. Main page defines the layout of all the pages excluding the login page. On the left side of the Main page the Drawer menu is rendered leaving the space for all content next to it on the right (Fig. 2.14). Since the main purpose of the app is to display data in graphs, it is possible to collapse the drawer into thin bar on the left part of the screen leaving more space for content rendering. The menu is divided into two sections - general and administration. While the general section is planned to be accessible by all users having the access to the app, the administration section will be restricted for site admins only. This division will allow user to display the data leaving the site configuration out of reach. The drawer menu acts as a router to the app. Clicking desired item (e.g. My Site, Sensors. . .) will redirect user to chosen page whose content will be rendered on the right.

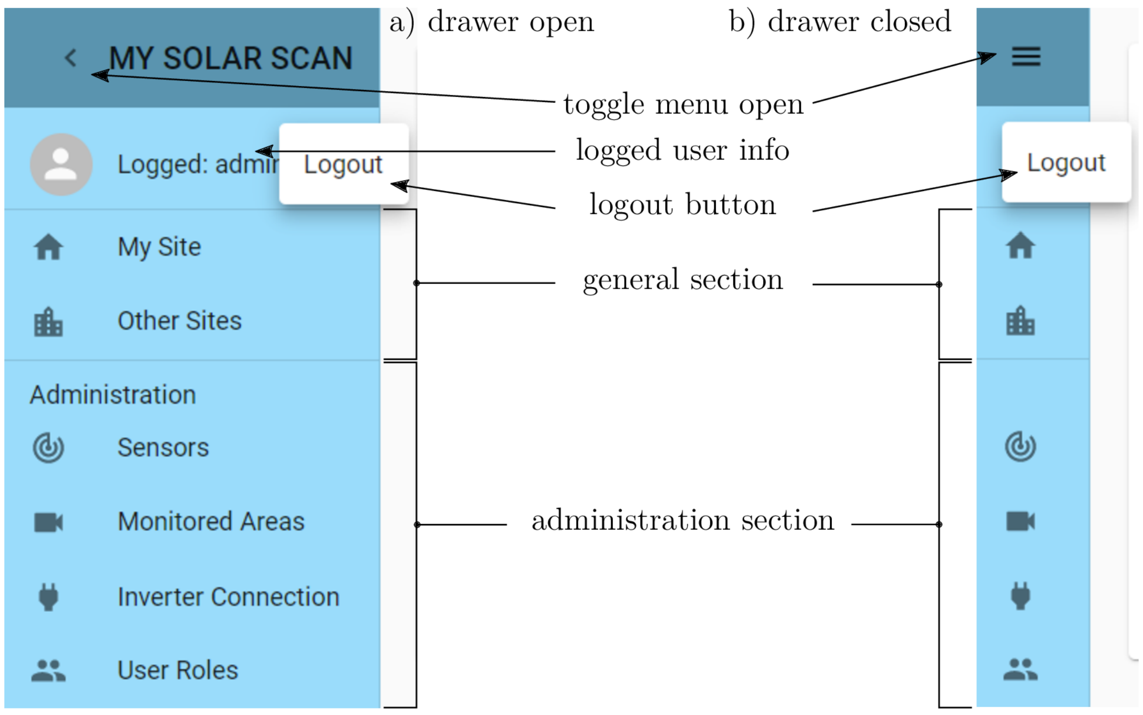


Fig. 2.14: The main page menu with the drawer (a) open, (b) closed.

2.6.6 The general section

My Site page acts as a home page as the logged user is redirected here. It takes care of displaying the graphs based on the site data.

Measurement picker

On the top of the page a measurement picker is situated (Fig. 2.15a). The purpose of measurement picker is to provide user with available measurements to display. The options are loaded from back end where the displaying logic is maintained. The picker has two sections of sensors connected to inverters and inverter not connected sensors. This division is based on the monitored area configuration logic. The sensors will be displayed in inverter-connected part if the monitored areas of all of them match the areas monitored by inverter. However difficult this might sound the real (inverter-measured) to theoretical (sensor-measured) energy comparison can only be made if all the photovoltaic panels connected to one inverter are monitored by some sensor. Only then the energy is not 'lost'. For clarity this is shown in diagram (Fig. 2.15b).

It will very often happen that the sensors are not fully connected[14] to any inverter. Then it is still possible to pick a measurement. Moreover it is allowed to

[14]Matching all monitored areas

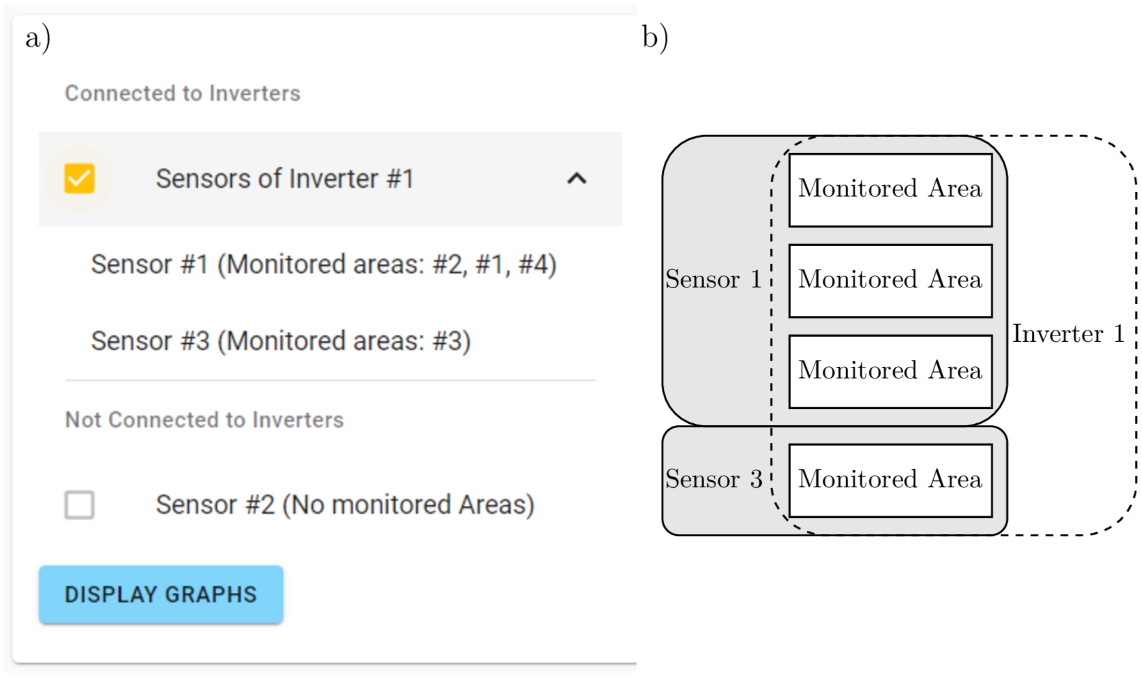


Fig. 2.15: The user can pick a measurement to display in measurement picker (a). Diagram of sensors with the same monitored areas as inverter (b)

pick more sensors and compare their measured values. All current user sensors must be present either in inverter-connected or in inverter-not-connected section.

Supported graphs

Choosing a sensor(s) in measurement picker and pressing `DISPLAY GRAPHS` will trigger the action of real-data-graph computation (details in Section 2.7.2). Based on the configuration there are 3 types of generated graphs listed from the most general ones which do not require much conditions to be met, up to the most specific performance ratio graph.

1. **Sensor irradiance and energy:** A separate graph is rendered for every sensor. If the sensor has ever measured any data, the irradiance over time is displayed. If the sensor is further connected to a monitored area with PV panel areas of defined size and panel type, this sensor specific theoretical energy is displayed. This stacked area chart keeps the information about individual monitored areas outputs (Fig. 2.16).
2. **Theoretical to real energy comparison:** Theoretical energy graph is displayed if all picked sensors meet criteria from (1.) and there is non-null intersection of their operating time intervals. Theoretical energy allows to compare theoretical output of PV panels monitored by selected sensors. Further, if the data from inverter (in overlapping time interval) are present, the comparison

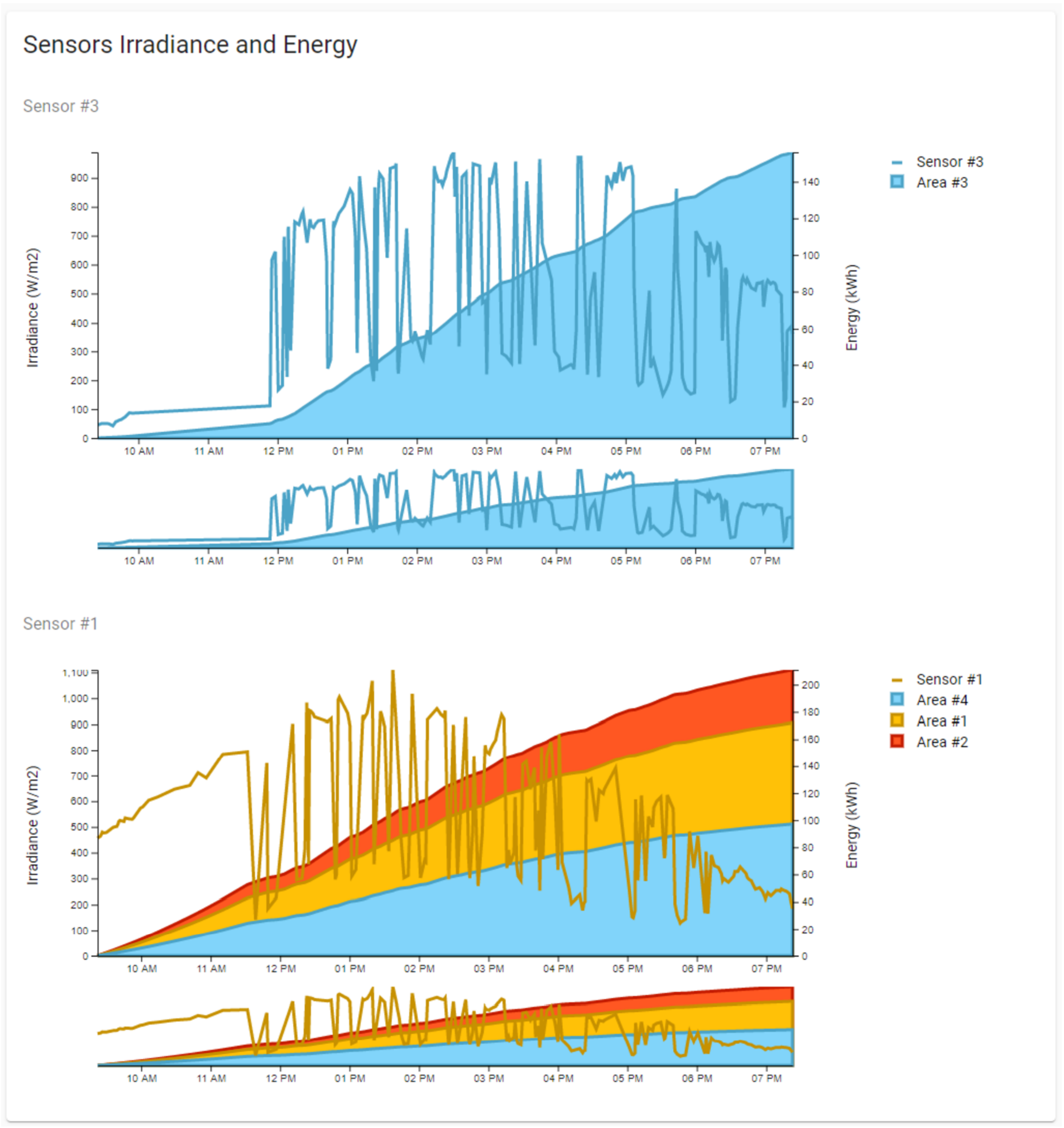


Fig. 2.16: The graphs of irradiance and energy computed from real irradiance data of sensors operating in Culemborg, near Utrecht, The Netherlands. Measured on May 25th. Monitored areas properties are fictional.

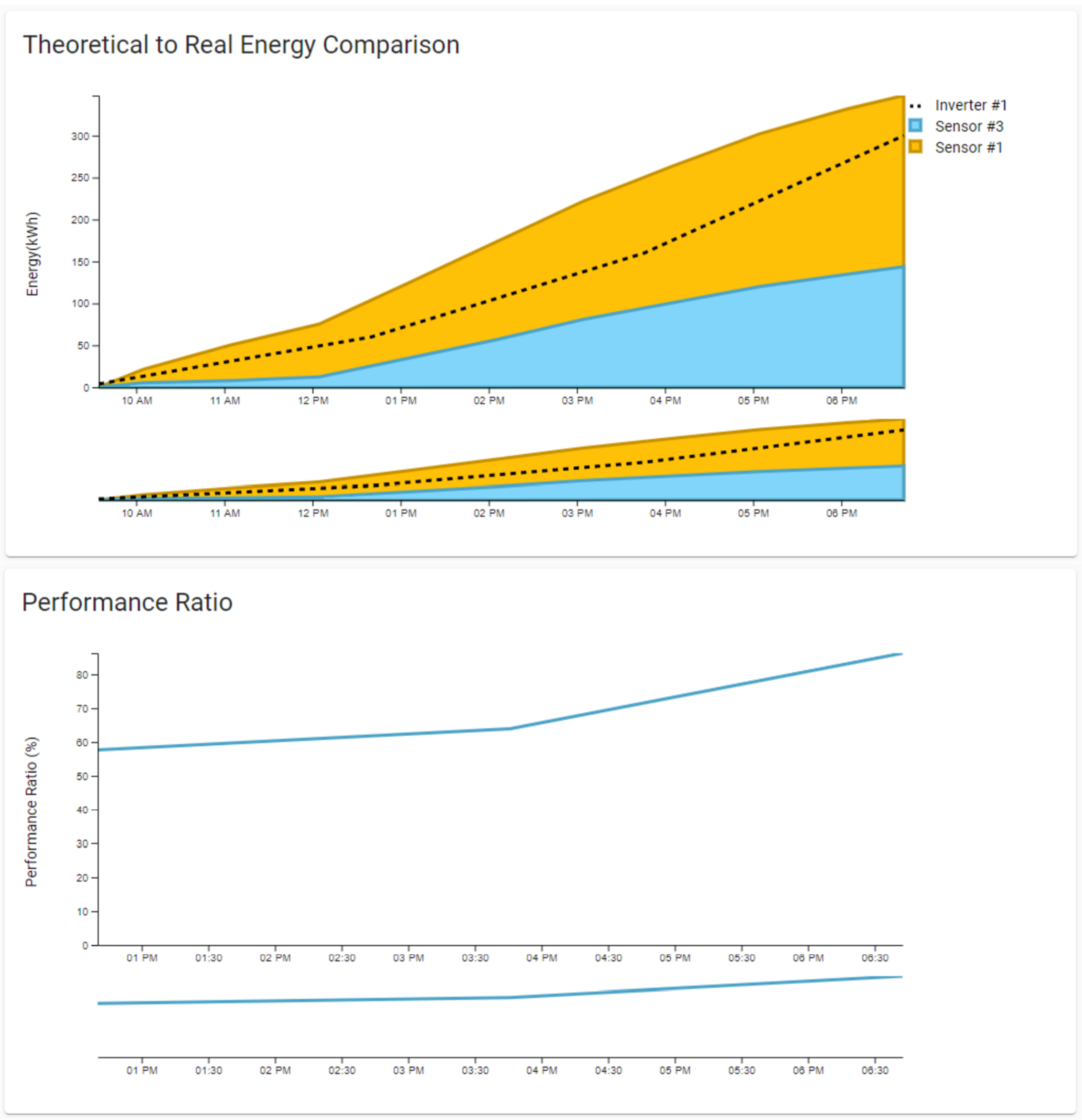


Fig. 2.17: Theoretical to real energy comparison and performance ratio graphs. The source of theoretical energy data is the same as in Fig. 2.16, inverter data are fictional.

of theoretical to real energy graph is displayed (Fig. 2.17).

3. **Performance ratio:** If all criteria from (2.) are met, also the performance ratio chart is displayed (Fig. 2.17).

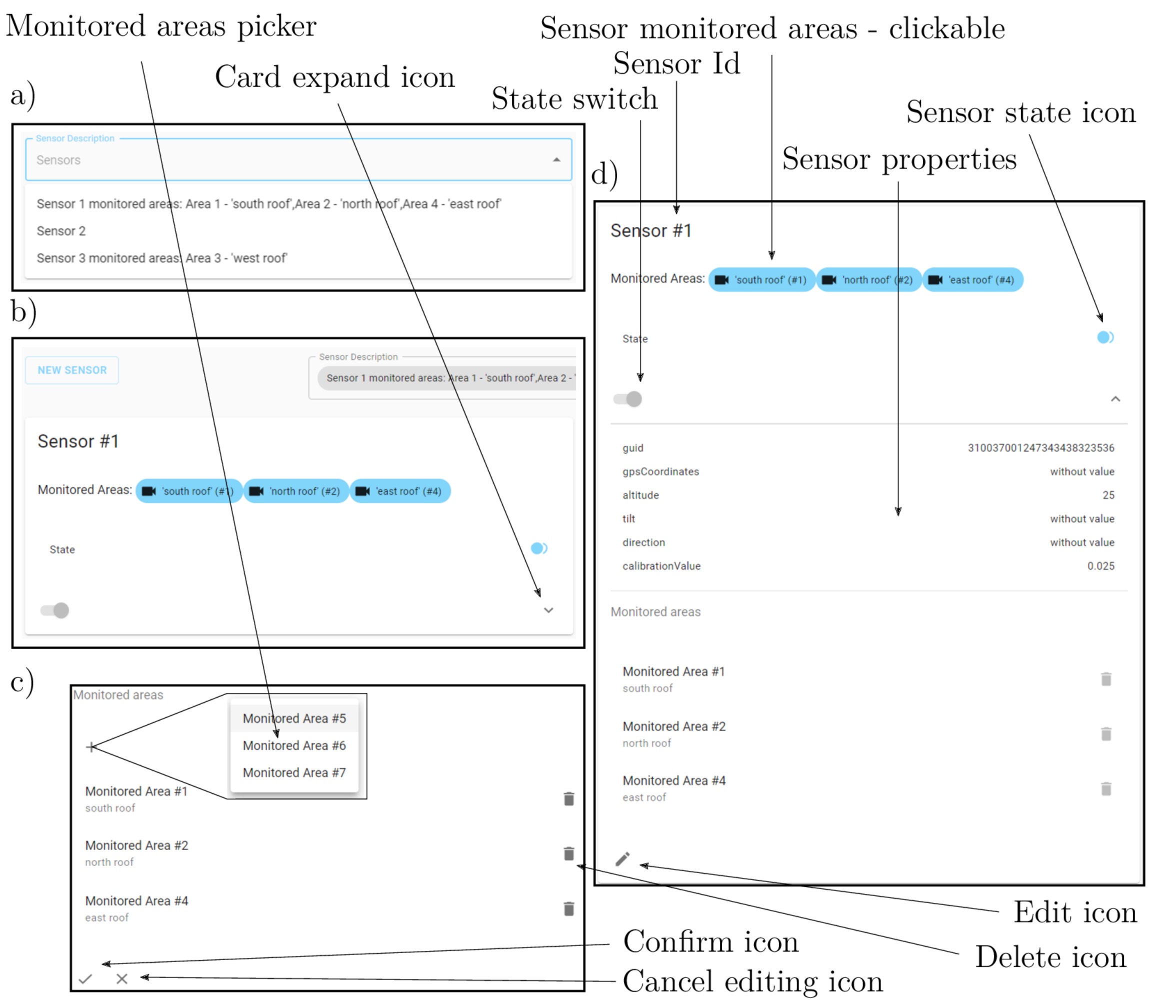
All supported graphs contain the brush which allows to select time interval to zoom in. The smaller same-shaped graph with only x-axis allows this interactive selection. The brush window once created can be modified simply by dragging its boundaries or the window itself as the whole. This feature might not seem essential when only one-day measurement is displayed but when measuring data for months, it will become crucial. All charts are computed with d3.js (see Sec. 2.1.2) and the brushing feature added with the help of [47] and [36].

2.6.7 The administration section

While the general section serves as a tool for data visualization, it is the administration where the type of displayable graphs is configured. Moreover it is not only about visualizing data but the administration section also reflects the real world situation of user site configuration. All three implemented parts (i.e. Sensors, Monitored Areas, Inverter Connection) have similar layout.

Sensors

The snippets from sensor administration page can be seen in Figure (2.18). On the top of the page there is `NEW SENSOR` button (b) which opens new sensor dialog on-click. The same image shows the Sensor card in its non-expanded form. In the to right corner the autocomplete filter is to be seen whose detail is shown in (a). This component allows to filter the content of the whole sensors page. It supports searching by any part of item title and selecting multiple items. By clicking the Card expand icon, the Sensor card detail is unpacked (d). Besides monitored areas chips, sensor id and sensor state, it also displays sensor properties. Clicking the edit icon (bottom left) switches into editable mode, where all properties can be modified. The example of adding monitored area to the sensor is shown in (c). When clicking at the '+' sign the monitored area picker appears offering available areas. This list is handled at the back end side and displays only such monitored areas which have not been assigned to any other sensor yet. It is also possible to unlink monitored area from the sensor via the delete icon making in accessible for other sensors. In editable mode also the state of the sensor might be changed. Every sensor state is set to CREATED on creation, state switch will remain on the left. Switching it for the first time will result in changing state to MEASURING. Clicking it again will result in switching state to STOPPED. Simply put, once the state has been switched from



Sensor 1 monitored areas: Area 1 - 'south roof',Area 2 - 'north roof',Area 4 - 'east roof'
Sensor 2
Sensor 3 monitored areas: Area 3 - 'west roof'

Sensor #1

guid	310037001247343438323536
gpsCoordinates	without value
altitude	25
tilt	without value
direction	without value
calibrationValue	0.025

Monitored areas

Monitored Area #1
south roof

Monitored Area #2
north roof

Monitored Area #4
east roof

Fig. 2.18: The snippets of sensor administration page. (a) The autocomplete component for filtering the page content, (b) sensor card not expanded, (d) sensor card detail expanded, (c) monitored areas card part in editable mode. The Figure does not reflect real component distribution. The black outline is only present for image clarity.

CREATED to MEASURING only switching between MEASURING-STOPPED can be handled.

The editing can be finished by one of two ways. Clicking the cancel icon results in discarding all the changes without any call to the server. Provided that the confirm icon has been clicked, then the data are sent to the server with PUT method updating desired values. After the update is completed, the whole page is re-rendered.

In order to simplify the transition to sensor connected monitored areas, the chips at the top of the card are made clickable. Clicking it triggers redirect to monitored areas page automatically setting the filter to the areas connected to the sensor from which redirected. Further, the action sets the actually clicked area as the first in the filter and automatically expands it.

Monitored Areas

The layout of Monitored areas page is very similar to the sensor page layout. The new monitored area dialog can be opened by the button in the top left corner and the areas cared about can be filtered by the same autocomplete component. Naturally, the difference is in the card content. Figure (2.19) shows monitored area card in editable mode. The monitored area description serves as the card title. Total area property is computed as the sum of all PV panel areas. Type of PV panel can be chosen from the picker in the corresponding column. To-date only Area and PV panel properties are considered in computation leaving the rest for the future development. All the icons have the same functionality as in case of sensor card. In the same manner it is possible to get redirected to sensor or inverter page by clicking the respective chip. Note that the redirection is only available in non-editing mode since this constraint simply prevents the loss of updated data[15].

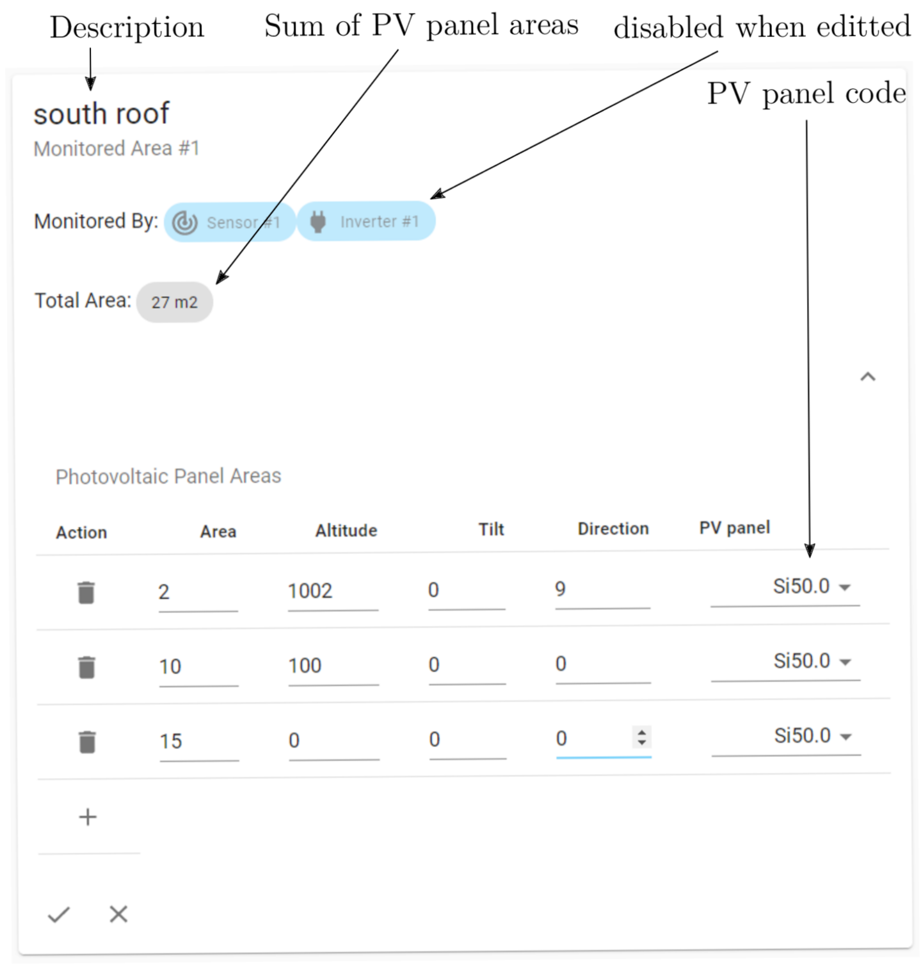
Inverter connection

This page is only in its beginnings. It allows to add a new inverter, but the filtering and a form for manually adding the inverter measured data still needs to be added

2.7 Computational base

This section aims to explain how the resulting graphs are computed based on input irradiance data.

[15]This behavior is the same in Sensors page.



south roof

Monitored Area #1

Monitored By: Sensor #1 Inverter #1

Total Area: 27 m2

Photovoltaic Panel Areas

Action	Area	Altitude	Tilt	Direction	PV panel
🗑	2	1002	0	9	Si50.0
🗑	10	100	0	0	Si50.0
🗑	15	0	0	0	Si50.0

Fig. 2.19: Monitored areas administration page card

2.7.1 Measurement data flow

Before diving into computational details the measurement data flow diagram will be introduced (Fig. 2.20). The description will start from inverter terminator. In this context it represents external entity sending data into the system. In most recent implementation the data have to be manually added directly to the database (here Measurements). The *Measured data* from MSS terminator are automatically processed. The first process (1. Measurement controller) represents the REST controller. When MSS sends data to exposed endpoint in correct form, the Measurement controller sends *Accepted data* for data parsing. Data parser breaks accepted data transfer object, verifies its structure and saves *Parsed data* into the database.

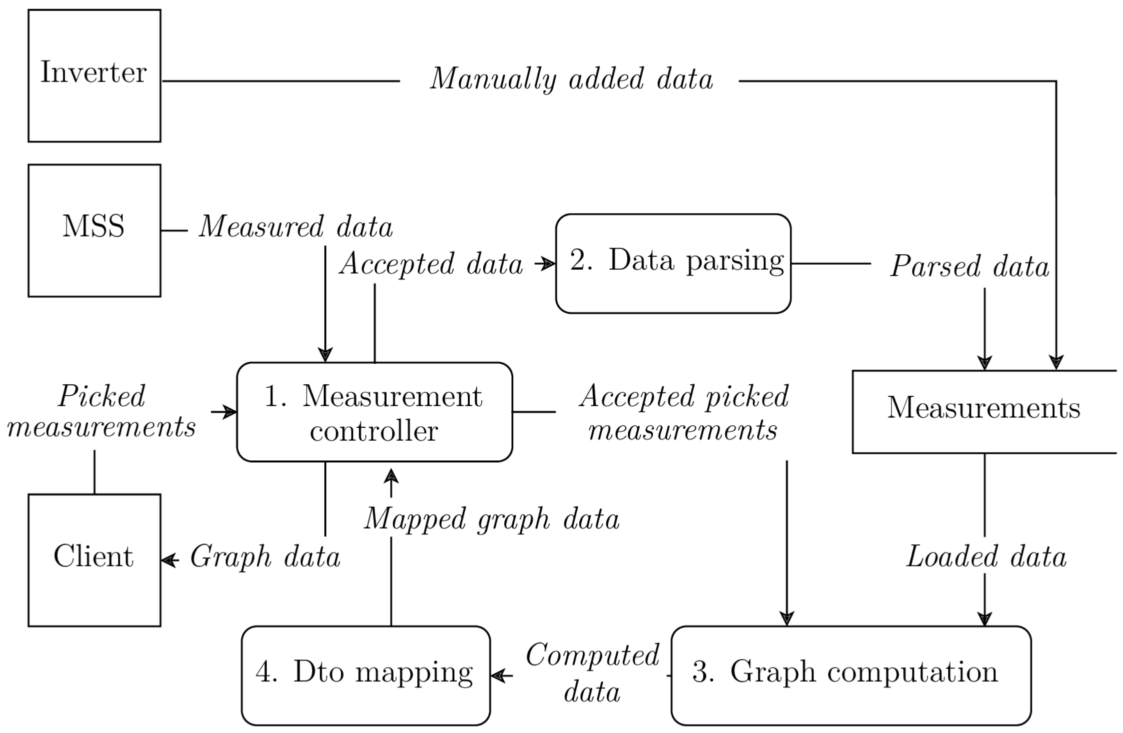


Fig. 2.20: Measurement data flow diagram. Squares represent terminators (external entities), rectangles represent processes executing actions on data flows (arrows) and the rectangle with missing border annotated Measurements stands for measurement tables in database. Notation according to [48].

In previous paragraph the processes that fill data into the database have been described. The third terminator - Client represents the whole React front end app. When the user clicks DISPLAY GRAPHS button in the GUI, this action triggers call to the measurement controller with *Picked Measurements* representing selected sensors ids (see Fig. 2.15). *Loaded data* corresponding to picked sensors measurements are then used by Graph computation process. When finished the *Computed data* needs to be mapped to dto in order to be sent back to the client via Measurement controller. *GraphData* included in dto are exactly in the form the d3.js framework needs them for graph rendering. In further text the '3. Graph computation' process will be the point of interest.

2.7.2 Computational details

In Section (2.6.6) all supported graphs are shown. The way how they were computed is now to be explained. The computation of performance ratio as was outlined in Section (1.2.3) will be the starting point.

The sensor measures solar irradiance. The energy which is the figure of merit can be computed by integrating the irradiance over time. Since the data are processed by machine the integration was substituted by summing. One of the possibilities of numerical integration is the trapezoidal rule which approximates under the curve area by a trapezoid. In mathematical terms

$$Q_i = \int_{t_{i-1}}^{t_i} H(t)\mathrm{d}t \approx (t_i - t_{i-1}) \cdot \frac{H(t_i) + H(t_{i-1})}{2} \tag{2.1}$$

The equation just specified the part in the sum of Equation (1.10). Figure (2.21a) shows trapezoidal rule graphically. This integration is done for every sensor. The energy of one PV panel area is computed as in Eq. (1.9) by multiplying Q by PV panel area size and corresponding PV panel efficiency. If the sensor monitors a single monitored area with more PV panel areas, then the output of a monitored area is simple sum of all PV panel areas outputs. If further more monitored areas are monitored by the sensor, the overall output is the sum of all monitored areas outputs. As can be seen in Figure 2.16, the information about individual monitored areas outputs is displayed in the energy chart of a given sensor.

More difficult situation occurs when it is needed to compare energy outputs of more sensors. Until now the computation was executed in the same time base as the differences between individual monitored areas were only in constant values of PV panel area efficiency and size. Firstly, only the data from the intersection of time intervals in which the to-be-compared sensors have measured can be used for further computation (Fig. 2.21b). Secondly, there is a problem of sensors logging the data in different times. Ideally, if the sensors logged data in the same time and there were no outages, the comparison would be as simple as comparing two values. However both phenomenons naturally occur in reality. The way of overcoming this issue is depicted in Figure (2.21c). Provided that only the data from intersecting time intervals have been picked, the common time interval is divided into multiple partial time intervals[16] with the center marked by T. The idea is apparent from the

[16] The value 10 min was chosen by default.

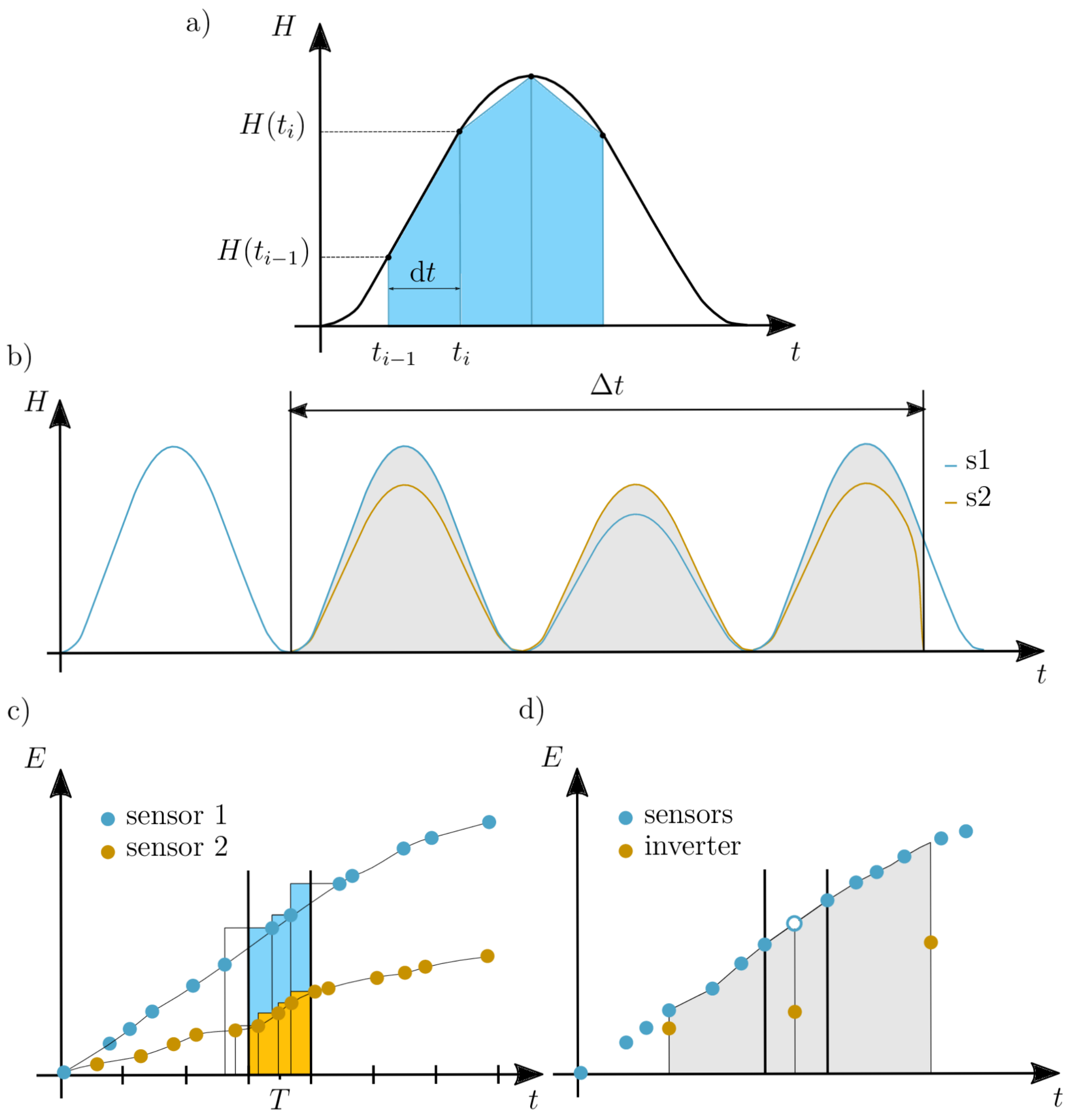


Fig. 2.21: Graph computation methods. (a) trapezoidal integration, (b) time intervals intersection, (c) Energy comparison, (d) interpolation for real energy comparison.

Figure. In the partial time interval the average energy for each sensor is computed by weighted average. For sensor 1 this would mean summing all three blue rectangle areas divided by partial time interval size. The resulting energy is paired with timestamp T. When this is done for all sensors, the energies are guaranteed to have the same timestamp and can be therefore directly compared or summed.

The last obstacle was to compare theoretical to real energy (Fig. 2.21d). Assuming that reading from inverter (added manually in days) will be logged much less frequently than sensor reading (minutes), the theoretical energy with the same timestamp as inverter reading can be obtained by interpolation. Dividing these two values finally gives the desired performance ratio.

2.8 Cloud deployment

The application is intended to run at a private server in the future. Until then it was deployed in Google Cloud which offers $300 free credit lasting for 12 month, which is ideal for testing purposes [49]. This project required a Compute Engine which is a virtual computer instance running Linux. Then the application needs a database without which it cannot exist. The database solution in Google Cloud, called Cloud SQL was also established. After that it was needed to solve routing so the Compute Engine can connect to the database. When the environment was set, it was possible to load the application in the form of `.jar` archive and run it on the Compute Engine. After running the application it is immediately accessible at the Compute Engine public IP on the specified port.

3 Discussion

This part aims to summarize the results of this master's thesis, repeat the key features of the solution and point out which real world problems it solves. The future directions of this project will also be outlined.

Starting with very brief theoretical fundamentals the spectrum of the Sun is described and the main descriptive quantities explained (Sec. 1.1.1). The daily solar irradiance (Sec. 1.1.2) which is dependent on the motion of the Sun defines the angles reflecting current Sun's position. The theoretical fundamentals are concluded with the explanation of performance ratio (Sec. 1.2.3) which is the figure of merit in terms of photovoltaic installations and its values are computed by the application developed in this project.

My Solar Scan (MSS) is the commercial name of the irradiance measurement device developed in ReRa Solutions. Its parts (sensor and connection box) are described (Sec. 1.3) together with an example of a reading (Lst. 1.1).

In order to carry out a meaningful literary research the project requirements as specified according to the market expertise of ReRa Solutions are provided in Section (1.4.1). The current MSS architecture bottleneck appears to be the Wi-Fi module used in the connection box. Thus the literary research (Sec. 1.4.3) compares multiple connection technologies e.g. Wi-Fi, Bluetooth, Zigbee, Sigfox, LoRa, NB-IOT... Discussing the pros and cons and this project specific requirements the optimal connection technology was chosen to be LoRa due to its low price and relatively dense net cover (Sec. 1.4.4).

The main focus in this project was however on the full-stack application which the whole Practical part is dedicated for. The software solution developed in this master's project is able to collect data from all connected sensors through API, store the data into the database, compute PV plant descriptive values based on the PV plant configuration and provide the user with feedback in the form of interactive graphs. The application information architecture describing the personas and the app purpose is given in Sec. (2.1.1). The application was developed as the single page. Modern programming languages and frameworks were used in order to make the app work smoothly and leave space for future development. The server back end is written in Kotlin programming language with the help of Spring framework. The responsiveness of GUI is ensured by React written in type safe form of JavaScript - TypeScript (Sec. 2.1.2).

The foundations of the application were laid when specifying the data model (Sec. 2.2). Multiple ERD and STD diagrams are described in the text. The most important part of the model design is the introduction of Monitored area concept (Sec. 2.2.4). It tries to reflect real-world use case scenario keeping the model general enough to support multiple configurations as well (Fig. 2.9). The model design section is concluded with to-date implemented model which copies the theoretical design (Sec. 2.2.8). The ways of transferring the model into database as well as programmatically executing database queries are described in Sec. ()2.3). The way of accessing the application (and the database) via REST API is explained in the Section (2.4). The application was provided with a login mechanism allowing to load user-specific environment. The authentication was implemented with the help of a Spring Security using JSON Web Tokens (Sec. 2.5).

Finally, after the description of server backend the focus was shifted to the client side - the GUI (Sec. 2.6). It starts with an explanation of client-server communication and individual parts of the GUI are described via application GUI snippets. The most important are the graphs of sensors irradiance and energy computed from real-measured irradiance data by the sensors installed by the ReRa technicians in the Netherlands (Fig. 2.16). The same data were used for computation of theoretical to real energy comparison and performance ratio graphs (Fig. 2.17). Next, the way of configuring the user specific site requirements in the administration section is presented in the Section (2.6.7).

The whole practical part is concluded with an explanation of a data flow starting in MSS device, hitting the measurement controller of the app responsible for logging the data into the database ending with computed values served to the end user in the from of graphs (Sec. 2.7.1). Special care is dedicated to the computational details (Sec. 2.7.2), where the numerical integration of sensor-measured irradiance is explained together with the way of comparing the data from more sensors measuring in different times. For theoretical to real energy comparison interpolation was used. Finally, the short description of deploying the application into cloud was given in the Section (2.8).

Limitations and Future outlook

The product of this project is fully-functioning application which is able to process real data. Even though the solution is still not ready for the distribution to the customer it lays the foundations to build upon. Some features were not implemented intentionally as the time for this project was strictly given, other limitations came up while developing the application.

The first future step will be to fully implement the database model including and using all system as well as the MSS versioning and PV panel area state tables. This step will bring better control of any state change and will make the system more modular. It will also help to create user-friendly labelling of all the entities in the application GUI (e.g. using mss type and version), while now primary entities ids are used instead. As the amount of logged data grows, the pagination feature which limits the amount of single response received data will start to be crucial. The welcomed feature would also be the real-time load of sensor measured data. On of the possible technologies which could be convenient in this case allowing the server to stream the data automatically to the client is called Serve-sent-events [50].

Also, the application does not support the deleting of sensors and monitored areas specified in the administration section via GUI. The measurement picker in My Site page could benefit from the autocomplete component already utilized for filtering e.g. in the Sensors page.

What takes to security, an important step for the user authentication process is running the application server on https protocol with trusted certificate. This step must certainly be implemented before exposing the application to the public. The application is prepared to support multiple user roles but the GUI counterpart is not ready yet. In the future the components of the application would be visible only to users with required privileges. The same applies for validations of any user input data. The data should be validated before letting the user to save any form.

The space for future development is without any doubt in the way of computing the graph data. The inaccuracies introduced by averaging theoretical energy across sensors in spite of its comparison could be minimized with the use of interpolation. Further, the possibility to mathematically compensate different tilt of PV panels and the sensors would allow to use the solution in wide range of PV installation. In order to provide precise values, the sensors need to be calibrated. Even though this can be done manually, the autocalibration methods would allow the sensor to calibrate on daily basis and therefore produce more trustworthy results. The autocalibration process backed by machine learning has already been studied at Radboud University [51].

Last but not least a proper testing of application code at the level of unit and integration tests need to be extended before going public.

4 Conclusion

In the first part, the literary research comparing different sensor connection possibilities was carried out according to the requirements. Multiple connection technologies (Wi-Fi, Bluetooth, Zigbee, Sigfox, LoRa, NB-IOT. . .) were studied and compared. Among all of them the Long Range (LoRa) technology has been concluded as the most suitable for this application mainly due to its low price, low power, high range and relatively dense net cover.

The device measuring solar irradiance with commercial name My Solar Scan (MSS) was developed by the ReRa Solutions company which partnered this project. In the cooperation with ReRa Solutions technician multiple MSS devices have been installed throughout the Netherlands. The data from these sensors served then as the input to the application.

This master's thesis focus was mainly on the application solution. The resulting software solution is a full-stack application running in the cloud which is able to collect data from from all connected sensors through an API, store these data into the database, compute PV plant descriptive values according to the PV plant configuration and display the result of the computation to the user in the form of interactive graphs. While the process of receiving the data from the sensors have been fully automated, the data from the inverter still need to be entered manually. Multiple modern frameworks have been used during the development. The application includes the back end part which does all the database related operations and computes the data and the front end part which displays the data to the user. The application was designed in such way that it supports different user PV plant configurations. The resulting application has been deployed to the Google Cloud together with its database allowing the remote access and maintaining high availability.

As already mentioned, the MSS devices were installed and fully running. Moreover, some of them have been configured to send data directly to developed cloud-hosted application. These real data were then used for computations of the graphs providing the client with an graphical overview of an output of so far fictional PV panel ares.

More detailed discussion together with the limitations and future outlook was provided in the previous Section (3).

Bibliography

[1] Hertl, V.: *Multiband solar sensors data acquisition and cloud processing*, Semestral Thesis, Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation, Brno, 2020, URL https://www.vutbr.cz/studenti/zav-prace/detail/122868.

[2] *Communication on The European Green Deal*, Tech. Rep., Brussels, 2019, URL https://ec.europa.eu/info/files/communication-european-green-deal_en.

[3] Masson-Delmotte, V. and et al: *An IPCC Special Report on the impacts of global warming of 1.5 C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty*, Tech. Rep. sr15, Intergovernmental Panel on Climate Change, Geneva, 2019, URL https://www.ipcc.ch/sr15/.

[4] *European Council conclusions, 12 December 2019*, Tech. Rep., 2019, URL http://www.consilium.europa.eu/en/press/press-releases/2019/12/12/european-council-conclusions-12-december-2019/.

[5] Kaur, T.: *Solar PV Integration in Smart Grid - Issues and Challenges*, 2015, doi:10.15662/ijareeie.2015.0407008.

[6] Shuda, J., Rix, A. and Booysen, M. T.: *Module-level Monitoring of Solar PV Plants using LoRa Wireless Sensor Networks*, 2018.

[7] *Performance ratio, Quality factor for the PV plant*, URL http://files.sma.de/dl/7680/Perfratio-TI-en-11.pdf.

[8] Mekki, K., Bajic, E., Chaxel, F. and Meyer, F.: *A comparative study of LPWAN technologies for large-scale IoT deployment*, ICT Express **5**(1), (2019), pp. 1–7, ISSN 2405-9595, doi:10.1016/j.icte.2017.12.005, URL http://www.sciencedirect.com/science/article/pii/S2405959517302953.

[9] Gray, J. L.: *The Physics of the Solar Cell*, in *Handbook of Photovoltaic Science and Engineering*, pp. 61–112, Wiley-Blackwell, 2005, ISBN 978-0-470-01400-4, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/0470014008.ch3.

[10] Honsberg, C. and Bowden, S.: *A collection of resources for the photovoltaic educator*, 2018, URL http://www.pveducation.org.

[11] *2000 ASTM Standard Extraterrestrial Spectrum Reference E-490-00*, URL `http://rredc.nrel.gov/solar/spectra/am0/ASTM2000.html`.

[12] *Reference Solar Spectral Irradiance: ASTM G-173*, URL `http://rredc.nrel.gov/solar/spectra/am1.5/ASTMG173/ASTMG173.html`.

[13] Haselhuhn, R. and Maule, P.: *Photovoltaic systems - energetical handbook*, Czech photovoltaic asociation, Pilsen, 2017, 1st ed., ISBN 978-80-906281-5-1.

[14] Bruno Burger, Klaus Kiefer, Christoph Kost et al.: *Photovoltaics Report - Fraunhofer ISE*, Tech. Rep., Fraunhofer Institute for Solar Energy Systems, ISE with support of PSE GmbH, Freiburg, 2019, URL `https://www.ise.fraunhofer.de/de/veroeffentlichungen/studien/photovoltaics-report.html`.

[15] Feldman, D. and Margolis, R.: *Q1/Q2 2019 Solar Industry Update*, Tech. Rep., The National Renewable Energy Laboratory, 2019.

[16] *Particle Photon datasheet*, URL `https://docs.particle.io/datasheets/wi-fi/photon-datasheet/`.

[17] *Particle | Device Cloud*, URL `https://www.particle.io/device-cloud/`.

[18] *VEML6075 UVA and UVB Light Sensor with I2C Interface*, Tech. Rep. 84304, Vishay Semiconductors, 2016.

[19] *ISL29125 Digital Red, Green and Blue Color Light Sensor with IR Blocking Filter*, Datasheet FN8424, Renesas, 2017.

[20] Lethaby, N.: *Wireless connectivity for the Internet of Things: One size does not fit all* p. 16.

[21] Cheruvu, S., Kumar, A., Smith, N. and Wheeler, D. M.: *Connectivity Technologies for IoT*, in *Demystifying Internet of Things Security: Successful IoT Device/Edge and Platform Security Deployment* (edited by Cheruvu, S., Kumar, A., Smith, N. and Wheeler, D. M.), pp. 347–411, Apress, Berkeley, CA, 2020, ISBN 978-1-4842-2896-8, doi:10.1007/978-1-4842-2896-8_5, URL `https://doi.org/10.1007/978-1-4842-2896-8_5`.

[22] Ruiz, J.: *Information Architecture. The Most Important Part of Design You're Probably Overlooking.*, 2019, URL `https://blog.prototypr.io/information-architecture-the-most-important-part-of-design-youre_probably-overlooking-20372ade4fc0`.

[23] William Craig: *Information Architecture 101: Techniques and Best Practices*, 2010, URL `https://www.webfx.com/blog/web-design/information-architecture-101-techniques-and-best-practices/`.

[24] Scott, E.: *SPA Design and Architecture: Understanding Single Page Web Applications*, Manning Publications, Shelter Island, NY, 2015, 1st ed., ISBN 978-1-61729-243-9.

[25] *React – A JavaScript library for building user interfaces*, URL `https://reactjs.org/`.

[26] *Reference - Kotlin Programming Language*, URL `https://kotlinlang.org/docs/reference/index.html`.

[27] *Spring Framework Documentation*, URL `https://docs.spring.io/spring/docs/current/spring-framework-reference/`.

[28] *Hibernate. Everything data. - Hibernate*, URL `https://hibernate.org/`.

[29] *Gradle User Manual*, URL `https://docs.gradle.org/current/userguide/userguide.html`.

[30] *The starting point for learning TypeScript*, URL `https://www.typescriptlang.org/docs/home`.

[31] *Material-UI: A popular React UI framework*, URL `https://material-ui.com/`.

[32] Bostock, M.: *D3.js - Data-Driven Documents*, URL `https://d3js.org/`.

[33] Crusoveanu, L.: *Inversion of Control and Dependency Injection with Spring*, 2016, URL `https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring`.

[34] Thai, N. N.: *What is a Spring Bean?*, 2018, URL `https://www.baeldung.com/spring-bean`.

[35] *Spring Boot*, URL `https://spring.io/projects/spring-boot`.

[36] Meeks, E.: *D3.js in Action: Data visualization with JavaScript*, Manning Publications, Shelter Island, NY, 2017, 2nd ed., ISBN 978-1-61729-448-8.

[37] *javax.persistence-api 2.2 javadoc (javax.persistence)*, URL `https://javadoc.io/doc/javax.persistence/javax.persistence-api/latest/index.html`.

[38] Gunnar Morling: *Jakarta Bean Validation specification*, 2019, URL https://beanvalidation.org/2.0/spec/.

[39] *MySQL What is DDL, DML and DCL?*, 2014, URL https://www.w3schools.in/mysql/ddl-dml-dcl/.

[40] Ramesh Fadatare: *What Is the Difference Between Hibernate and Spring Data JPA? - DZone Java*, URL https://dzone.com/articles/what-is-the-difference-between-hibernate-and-sprin-1.

[41] *Representational state transfer*, page Version ID: 956443795, 2020, URL https://en.wikipedia.org/w/index.php?title=Representational_state_transfer&oldid=956443795.

[42] Martin Fowler: *Catalog of Patterns of Enterprise Application Architecture: Data Transfer Object*, URL https://martinfowler.com/eaaCatalog/dataTransferObject.html.

[43] *JSON Web Tokens*, URL http://jwt.io/.

[44] Rajeev Singh: *Spring Boot + Spring Security + JWT + MySQL + React Full Stack Polling App - Part 2*, 2018, URL https://www.callicoder.com/spring-boot-spring-security-jwt-mysql-react-app-part-2/.

[45] *Create React App · Set up a modern web app by running one command.*, URL https://create-react-app.dev/docs/getting-started.

[46] *About npm | npm Documentation*, URL https://docs.npmjs.com/about-npm/.

[47] Bostock, M.: *Focus + Context*, 2020, URL https://observablehq.com/@d3/focus-context.

[48] Ráček, J.: *Strukturovaná analýza systémů*, vol. I, Masarykova univerzita, 2006, ISBN 978-80-210-4190-5, URL https://is.muni.cz/publication/708337.

[49] *Cloud Computing Services*, URL https://cloud.google.com/.

[50] *Using server-sent events*, URL https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events.

[51] Bosma, J.: *Improving a Machine Learning Algorithm to Predict Solar Irradiance by building and applying specialised data science tools*, Tech. Rep., Radboud University, Nijmegen, 2019.

List of symbols, physical constants and abbreviations

6LoWPAN	IPv6 Low Power Wireless Personal Area Network
AJAX	Asynchronous JavaScript And XML
BAN	Body Area Network
BE	Back End
BLE	Bluetooth Low Energy
CSS	Cascading Style Sheets
DDL	Data Definition Language
DFD	Data Flow Diagram
DML	Data Manipulation Language
DOM	Document Object Model
DSL	Domain-specific Language
DTO	Data Transfer Object
ERD	Entity Relationship Diagram
FE	Front End
FF	Fill Factor
GUI	Graphical user interface
IEEE	Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoC	Inversion of Control
IoT	Internet of Things
ISM	Industrial, scientific and medical
JDBC	Java Database Connectivity
JPA	Java Persistence API
JSON	JavaScript Object Notation
JWT	Json Web Token
LAN	Local Area Network
LoRa	Long Range
LPWAN	Low Power Wide Area Network
MSS	My Solar Scan
NB-IoT	Narrowband IoT
LTE-M	Long-Term Evolution for Machines
NTC	Negative Temperature Coefficient
ORM	Object Relational Mapping
OSI	Open Systems Interconnection
PAN	Personal Area Network

PR	Performance Ratio
PV	Photovoltaics
REST	Representational State Transfer
RFC	Request For Comments
SPA	Single Page Application
STD	State Transition Diagram
UI	User Interface
WAN	Wide Area Network

List of appendices

A Content of enclosed CD **99**

A Content of enclosed CD

The CD contains:

1. the pdf version of the thesis,
2. built application in `app-built` folder,
3. application source code `app-source`. In electronic version as `app-source.zip`.

Please note that due to large amount of files only simplified dir tree is provided including the most important files only.

```
/ ...................................................... root folder of enclosed CD
├── DP-Hertl-Vít-160774.pdf ................................... main document
├── app-built
│   ├── my-solar-scan.jar ............. assembled application executable in Java 8
│   └── application.properties .................... application settings for Spring
└── app-source ........................................... application source code
    └── my-solar-scan
        ├── backend ................................ application back end - Kotlin
        │   ├── src.main.kotlin.com.rera.mss .......... back end source code folder
        │   │   ├── config ........................ app constants and security config
        │   │   ├── controller ..................................... REST controller
        │   │   ├── exceptions .................................. exception handling
        │   │   ├── measurement ............................... computational base
        │   │   ├── model ................................ programmed db entities
        │   │   ├── payload ................................................. Dtos
        │   │   ├── repository ............................... db access interfaces
        │   │   ├── security ......................................... security related
        │   │   ├── service ....... services standing between controller and repositories
        │   │   └── MySolarScanApplication.kt ... application launcher, main method
        │   ├── src.main.resources ............................ back end resources
        │   └── src.test.kotlin.com.rera.mss ................ back end test folder
        ├── frontend ........................... application front end - TypeScript
        │   ├── src ..................................... front end source code folder
        │   │   ├── api .............................. client side REST request base
        │   │   ├── app ................................ highest level app component
        │   │   ├── common ..... components shared accross app e.g. loader, error dialog
        │   │   ├── components ........................... all the React components
        │   │   │   ├── administrationSection ....... sensor, inverter, monitored area
        │   │   │   │                                 components
        │   │   │   ├── generalSection .... my site component with all types of graphs
        │   │   │   └── mainPage ........................... layout page with drawer
        │   │   └── constants ......................................... FE constants
        │   └── package.json ................................... npm dependencies
        ├── .gitignore
        ├── build.gradle.kts .................. definition file of gradle build scripts
        ├── gradlew
        ├── gradlew.bat
        └── settings.gradle.kts ........ gradle project'and subprojects specification
```