BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

RESPONSIVE VISUALIZATION OF IOT DATA USING DASHBOARDS

RESPONZIVNÍ VIZUALIZACE IOT DAT POMOCÍ OBRAZOVEK TYPU DASHBOARD

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **MARKO OLEŠÁK**
AUTOR PRÁCE

SUPERVISOR **Ing. PETR JOHN**
VEDOUCÍ PRÁCE

BRNO 2025

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

161069

Institut:	Department of Information Systems (DIFS)
Student:	**Olešák Marko**
Programme:	Information Technology
Title:	**Responsive Visualization of IoT Data Using Dashboards**
Category:	Web applications
Academic year:	2024/25

Assignment:

1. Study the Internet of Things (IoT) and Smart City.
2. Study existing approaches to visualization. Focus primarily on visualizing data from smart devices using dashboards. Study existing types of visualizations suitable for their design.
3. Analyze the requirements for visualizing data collected from IoT devices. Delve into the issues of visualizing the current state of groups of smart devices and historical data of individual devices.
4. Present a proposal for web components and a set of views using these components that can effectively process and graphically visualize data from smart devices in a user-friendly manner, with a primary focus on mobile device users.
5. Implement and integrate the proposed solution.
6. Test the resulting solution for usability. Suggest possible extensions.

Literature:

- Greengard, S. (2015). The Internet of Things. MIT Press, ISBN 978-026-2527-736.
- Few, S. (2006). Information Dashboard Design: The Effective Visual Communication of Data. Sebastopol, USA: O'Reilly, ISBN 978-059-6100-16
- Bureš, M. (2024). Systém pro zpracování dat z chytrých zařízení, Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Jiří Hynek, Ph.D.
- John, P. (2024). Optimising processes in IoT. Brno. PhD thesis proposal University of Technology, Faculty of Information Technology. Supervisor prof. Ing. Tomáš Hruška, CSc.

Requirements for the semestral defence:

- Points 1 - 4.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**John Petr, Ing.**
Head of Department:	Kolář Dušan, doc. Dr. Ing.
Beginning of work:	1.11.2024
Submission deadline:	14.5.2025
Approval date:	22.10.2024

Abstract

This thesis addresses the challenges of monitoring and controlling smart devices within increasingly complex Internet of Things (IoT) environments. The main goal of the work is to design and implement a responsive, scalable, and intuitive dashboard solution, primarily targeted at the smart home domain, capable of catering to the diverse needs of both beginner and advanced users. The developed dashboard is built from scratch and connects to the existing Real-Time IoT (RIoT) system, which handles backend operations. It introduces a user-customizable dashboard layout alongside detailed views for individual devices and device groups, enhancing the granularity and flexibility of system monitoring. Special emphasis was placed on enabling user customization, allowing users to tailor the dashboard layout and content according to their preferences and needs. Furthermore, the dashboard leverages Key Performance Indicators (KPIs) and effective data visualizations to clearly present both real-time and historical data, with support for live updates and scheduled data re-fetching.

Abstrakt

Táto práca sa zaoberá výzvami monitorovania a riadenia inteligentných zariadení v čoraz komplexnejších prostrediach internetu vecí (IoT). Hlavným cieľom práce je navrhnúť a implementovať responzívne, škálovateľné a intuitívne riešenie prehľadovej obrazovky typu dashboard, primárne zamerané na oblasť inteligentných domácností, ktoré dokáže uspokojiť rôznorodé potreby začiatočníkov aj pokročilých používateľov. Vyvinutý dashboard je implementovaný od základu a je prepojený s existujúcim systémom Real-Time IoT (RIoT), ktorý zabezpečuje backendové operácie. Dashboardové vyhotovenie prináša používateľsky prispôsobiteľné prostredie spolu s detailnými pohľadmi pre jednotlivé zariadenia a skupiny zariadení, čím zvyšuje úroveň granularity a flexibilitu monitorovania systému. Dôraz bol kladený na užívateľskú prispôsobiteľnosť, vďaka čomu si používatelia môžu prispôsobiť rozloženie a obsah dashboardu podľa svojich preferencií a potrieb. Okrem toho, dashboard využíva kľúčové ukazovatele výkonnosti (KPIs) a efektívne vizualizácie pre jasné zobrazenie údajov v reálnom čase a historických dát, s podporou aktualizácií v reálnom čase a plánovaného opätovného načítania dát.

Keywords

Internet of Things, Smart Cities, Dashboards, Smart Homes

Klíčová slova

internet vecí, dashboard, prehľadové obrazovky, inteligentná domácnosť

Reference

OLEŠÁK, Marko. *Responsive Visualization of IoT Data Using Dashboards.* Brno, 2025. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Petr John

Rozšířený abstrakt

Internet vecí (IoT) sa dnes dostáva do mnohých oblastí nášho života, od priemyselných aplikácií, cez poľnohospodárstvo, až po každodenné prostredie, kde sa stáva neoddeliteľnou súčasťou inteligentných domácností (Smart Homes) alebo mestských infraštruktúr, z ktorých postupne vznikajú inteligentné mestá (Smart Cities). IoT prináša nové možnosti a prepojenia širokého spektra zariadení, ktoré dokážu komunikovať medzi sebou a poskytovať hodnotné dáta. S týmto trendom vzniká čoraz viac inteligentných domácností, ktoré prinášajú nové možnosti v spôsobe používania každodenných technológií alebo automatizácie, ako sú napríklad osvetlenie, zabezpečenie či možnosť automatického zalievania. Rýchly vývoj a adopcia týchto zariadení však vyvoláva nové výzvy a potreby v oblasti ich správy či monitorovania a rovnako aj vizualizácie dát. Okrem toho, sa IoT v dnešnej dobe často označuje ako kľúčový článok štvrtej priemyselnej revolúcie, čo túto potrebu ešte viac zdôrazňuje a demonštruje schopnosti týchto systémov.

Jedným z hlavných problémov, ktoré používateľov týchto zariadení postihujú, je vysoká miera fragmentácie IoT ekosystému. Väčšina zariadení je úzko viazaná na konkrétneho výrobcu a zvyčajne vyžaduje používanie vlastnej a uzavretej aplikácie. Používatelia, ktorí vlastnia zariadenia rôznych značiek sú potom nútení inštalovať a spravovať rôzne aplikácie s rozdielnými užívatelskými rozhraniami, ktoré okrem toho častokrát vôbec medzi sebou nekomunikujú a automatizácie medzi týmito menšími ekosystémami sú problémové. Výsledkom tejto fragmentácie je neprehľadné, neefektívne prostredie, ktoré sa vzďaľuje od ideológie inteligentnej domácnosti.

Na riešenie tohto problému vznikol systém Real-Time IoT (RIoT), ktorý má cieľ vytvoriť jednotné, rozšíriteľné prostredie pre združovanie heterogénnych zariadení rôznych výrobcov pod jednu platformu. Okrem zberu a ukladania dát tento systém umožňuje definovanie a vyhodnocovanie kľúčových indikátorov výkonnosti (KPIs) v reálnom čase a poskyuje podporu pre vizuálny porgramovací jazyk (VPL), ktorý slúži pre jednoduchú konfiguráciu automatizácií.

S narastajúcou komplexitou IoT systémov, vzniká spomínaná potreba efektívneho a intuitívneho nástroja umožňujúceho ich správu a monitorovanie. IoT však vyžaduje hneď niekoľko požiadavkov, ako napríklad interaktívne ovládanie zariadení či zobrazenie dát nie len v reálnom čase, ale aj umožnenie pozorovania historických dát, ktoré častokrát môžu odhaliť korelácie, opakujúce trendy alebo upozorniť na anomálie. Vzhľadom na rozmanitosť komplexít IoT systémov a nárokov užívateľov, musí byť takýto systém flexibilný a používateľsky prívetivý pre čo najväčšiu doménu užívateľov, od začiatočníkov po pokročilých. Kľúčovým prvkom pre monitorovanie týchto systémov sú prehľadové obrazovky typu dashboard, ktoré umožňujú jednoducho monitorovať aj komplexné systémy pri správnom nastavení.

Táto práca sa zaoberá návrhom a implementáciou týchto obrazoviek, ktoré adresujú spomínané požiadavky a problémy. Navrhovaný dashboard je postavený na predstavenom existujúcom RIoT systéme, ktorý zabezpečuje backendové operácie pre túto prácu. Pôvodný RIoT systém obsahoval aj jednoduché užívatelské prostredie v podobe webovej aplikácie, avšak neobsahovalo žiadne dátové vizualizácie ani samotný dashboard. Táto práca teda rozširuje jeho predošlú funkcionalitu, ktorá bola zameraná prevažne na spomínané KPIs, o dashboard, ktorý nadväzuje na predošlé funkcionality ako napríklad vizualizácie stavov skupín pomocou KPIs a pridáva monitorovanie a ovládanie zariadení.

Dôležitou prioritou návrhu bolo zabezpečenie vysokej flexibility a prispôsobiteľnosti, aby dashboard vyhovoval širokej škále používateľov. Neodlučiteľnou súčasťou čoho sú aj implementované interaktívne formuláre pre vytváranie nových vizualizácií alebo úpravy

existujúcich vizualizácií. Tieto formuláre poskytujú živý náhľad v závislosti od aktuálnej konfigurácie a umožňujú detailnú úpravu vizualizácií. Samotný dashboard využíva mrežový dizajn, ktorý automaticky prispôsobuje počet stĺpcov podľa dostupnej šírky obrazovky. Pre menšie zariadenia a zariadenia s dotykovými obrazovkami bolo implmentované vertikálne posúvanie a rôzne prispôsobenia gestových akcií, ako napríklad dlhé podržanie na grafe, či horizontálne gestá pre prepínanie medzi rôznymi kategóriami dashboardu.

Okrem hlavného prehľadu systém umožňuje zobrazenie detailných informácií o jednotlivých zariadeniach alebo ich skupinách. Pri rozsiahlych systémoch nie je prakticky možné vizualizovať všetky parametre každého inteligentného zariadenia na hlavnom prehľadovom pohľade. Miesto toho, dashboard poskytuje interaktívne prvky, ktoré po kliknutí poskytujú detailné informácie o danom zariadení.

Návrh prehľadových obrazoviek, či detailných stránok však nesie niekoľko limitácií a pravidiel, ako napríklad hierarchické rozloženie informácií alebo farebná paleta vizualizácií, ale aj použitie vhodných interakcií pre zjednodušenie používania. Napríklad, kľúčové informácie sú strategicky umiestnené v hornej ľavej alebo strednej časti obrazovky, čím upútavajú prirodzenú pozornosť užívateľa.

Výsledné riešenie bolo testované užívateľmi, ktoré odhalilo niekoľko chýb prevažne v oblasti užívateľskej použiteľnosti. Na základe týchto výsledkov bola väčšina týchto problémov týkajúcich sa navigácie, zrozumiteľnosti vizualizácií či intuitívnosti ovládania adresované iteratívnymi úpravami.

Responsive Visualization of IoT Data Using Dashboards

Declaration

I declare that I have prepared this bachelor thesis independently under the supervision of Mr. Ing. Petr John. I have cited all literary sources, publications and other sources which I have used in this thesis. Artificial Intelligence tools were used in the form of Grammarly and Writefull extensions for spell-checking and textual enhancements, and GitHub Copilot for enhanced auto-complete features, and debugging purposes, especially for complex TypeScript errors.

.....
Marko Olešák
May 12, 2025

Acknowledgements

I would like to express my sincere gratitude to Mr. Ing. Peter John for his patience and helpful views during the process of preparing this thesis, which significantly influenced the quality of the outcome. I would also like to thank my parents for their support throughout my academic journey, and my partner for constant encouragement and understanding.

Contents

List of Figures

Chapter 1

Introduction

The rapid development of technology has brought along the Internet of Things (IoT) era. This revolutionary concept is changing industries, cities, and even the style of everyday living. IoT is believed to be the core of the fourth industrial revolution, often referred to as Industry 4.0, where interconnected devices and systems are meant to enable unseen levels of automation, efficiency, and decision-making based on collected data. IoT enables communication between devices, systems, and users, creating new opportunities and innovations in many spheres. From smart homes to industrial automation and smart cities, the IoT revolutionizes the way we interact with technology. In the context of smart homes, IoT provides new possibilities for automation, such as in the form of smart energy management systems (HEMS) or security, and enhances the general user comfort.

As the number of IoT devices increases, so does the complexity of managing and monitoring these systems. Effective monitoring and control are called for, and they clearly unveil a critical gap in the market. While many existing solutions exist, they often provide strictly predefined visualizations or lack other aspects such as responsiveness or user-friendliness, which might not be sufficient in many cases.

Currently, a new platform called Real-Time IoT (RIoT) is under development, aiming to provide an open and flexible solution capable of integrating devices from various brands into a single unified system, eliminating the need for multiple brand-specific applications. However, the platform's frontend remains underdeveloped and lacks essential features, including device state visualizations.

The main objective of this thesis is to design and implement a dashboard solution for the mentioned system alongside detailed views for devices and their groups. The solution will focus on proposing a responsive, intuitive, scalable dashboard that enables efficient monitoring and control of IoT systems, especially within smart homes. By incorporating Key Performance Indicators (KPIs) from the backend system, combined with comprehensive data visualizations and responsiveness, the dashboard should be robust and usable by a large user domain, from beginners to advanced, regardless of the device used.

This thesis first introduces the concepts of IoT and general usage examples in Chapter 2. Following that, dashboard design principles, including different dashboard types, their design, and data visualization principles, are discussed in Chapter 3. Existing solutions, with their weaknesses and strengths, are analyzed in Chapter 4, and building on these insights and the derived requirements, Chapter 5 presents the proposed solution. Finally, Chapter 6 describes the implementation process, addresses its challenges, and showcases the final dashboard and its components.

Chapter 2

Internet of Things

Defining the Internet of Things (IoT) with a single definition is challenging, to say the least. There is no common or widely recognized IoT definition globally. The many existing definitions can vary dramatically, mainly by the overall scope of the definition. While some argue that almost anything can be part of IoT, others are against it.

Experts from the European Union defined the IoT [11] as a global, distributed network of physical objects that are capable of sensing or acting on their environment and are able to communicate with each other or machines. Meanwhile, Oracle defines the IoT[1] as the network of physical objects—"things"—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet.

In this thesis, we will define the Internet of Things as a network of entities or devices capable of measuring and acting on the environment around them and communicating by sending data to the Internet or another IoT entity. These entities can range from simple sensors, such as those measuring temperature, humidity, or luminance commonly found in smart homes, which are our primary focus, to more complex systems like industrial machinery, connected vehicles, and advanced healthcare devices. IoT networks in an industrial setting are also often referred to as the Industrial Internet of Things (IIoT) or described as the platform underlying Industry 4.0, the fourth industrial revolution [33].

Although IoT is a broad term, it is just one part of the larger concept known as the Internet of Everything (IoE) [31]. IoE extends further and expands the scope of IoT by joining people, processes, data, and things into a vast, highly interconnected network. IoE connects people-to-people, people-to-machines and machines-to-machines. All this allows for real-time data exchange and smarter interactions, resulting in IoE being seen as the next phase of IoT and often even visualized as society's future.

2.1 History of the Internet of Things

The first attempt at adding intelligence and sensors to a physical object happened in the 1980s at the computer science university of Carnegie Mellon [8]. Back then, members of the Computer Science Department modified a soda vending machine to remotely track whether or not it needed refilling and monitor the temperature of the beverages. This project is often marked as the first IoT device.

[1] Oracle – What Is the Internet of Things? Available at https://www.oracle.com/internet-of-things/

Since the mid-1990s, researchers have begun to explore the opportunities of making humans and machines more interconnected. Later, in 1997, Kevin Ashton [20] focused on exploring radio-frequency identification (RFID), which would allow physical objects to be identified via a simple passive tag and radio-frequency signals. In 1999, Kevin Ashton was the first person to use the term "the Internet of Things" in a speech. Since then, billions of devices have been put to use in IoT, and by the end of 2024, the number of IoT devices is expected to exceed 18.8 billion [32].

2.2 Usages and Needs

For many years, these smart devices have been generating vast amounts of data, which can significantly impact automation, enhance efficiency, or provide environmental benefits in almost any sphere. Among others, this progress led to the development of Smart Cities, further described in Section 2.5.

With such a large number of entities operating, there is an urgent need for unique identifiers [19]. From a practical point of view, each device is assigned a unique identifier (UID) and often an IP address to facilitate seamless communication in internet-based networks and easy distinction between entities. These entities can connect using numerous interfaces, such as wires, cords, satellites, and other communication technologies, described in Section 2.3.

> "Connected devices don't necessarily have to connect to the Internet of Things, but they increasingly do. What's more, they extend connectedness far beyond computers and into all the nooks and crannies of the world [19]."
>
> ---
>
> *Samuel Greengard*

Real-World Use

IoT devices can drive significant efficiency gains in real-world applications with correct data analysis and visualizations. These visualizations oftentimes provide a real-time overview of the current state of a system and, if done correctly, alert the user before something goes wrong. When it comes to large production lines or factories [34], any downtime is critical. IoT makes it possible to continually monitor production and provide alerts when something is malfunctioning or about to break. Furthermore, IoT enables deeper optimization and increases efficiency in this sector, even if by small percentages, saving significant assets.

Predictive maintenance is also one of IoT's usages. In the railway sector [16], smart trains equipped with sensors generate large data sets while riding on the tracks. These data sets can be further processed, and the data will represent the train's condition either directly or indirectly. By doing so, delays can be reduced by preventing breakdowns and enhancing maintenance scheduling.

Radio-frequency identification (RFID) [19], further discussed in Section 2.3, has become a vital part of many industries. In retail stores, RFID tags are used to track products and make inventory tracking much easier. RFID also keeps items from being stolen with special gates that act as a reader and sound an alarm when a tag crosses. Another great example is the aircraft manufacturer Airbus. Using RFID, their smart factory system tracks tools, logistics, and even wing production in real-time.

Home energy management systems (HEMS) [45] allow households to monitor, control, and optimize their energy consumption by utilizing the capabilities of their smart homes. These systems often integrate renewable energy sources, such as solar or wind, alongside energy storage devices such as batteries. Energy storage devices can further help smart homes save money by charging them during off-peak times and then by discharging electricity during peak times when prices are higher.

2.3 Types of Devices

There are thousands of sensors surrounding us each day, even without us directly noticing them. They range from simpler ones like home weather stations that we quickly glance at when deciding what to wear for the day, to more complex ones like surveillance cameras. When combined, they make our lives safer, more convenient, and more efficient by automating tasks and providing real-time data insights. Sensors produce different types of data, thus requiring different visualizations. Some produce merely basic numerical values, while others output GPS coordinates or live camera feeds.

Among the numerous taxonomies for categorizing IoT devices is the division based on their "smartness" [18], creating two distinct groups, Smart Objects and Not-Smart Objects.

Smart Object, also known as Intelligent Product, is an entity capable of being identified while also having the ability to interact with the environment around it. Not only that, it is also capable of independently acting in an intelligent way under different conditions. These devices are also embedded with an operating system. They often combine sensors and actuators to gather data, process it, and perform actions. Examples of Smart Objects include smartphones, microcontrollers, and intelligent home devices.

The Not-Smart Objects are without any intelligence, making them unable to act based on the environmental conditions. Nevertheless, these objects still form a critical part of IoT. The category of Not-Smart Objects can be further divided into sensors and actuators.

- Sensors are devices capable of detecting and measuring physical parameters, such as sound, temperature, and many others.
- Actuators can be further divided into two categories: mechanical devices and actions. Mechanical actuators are capable of performing operations either on themselves or on other devices, such as a servomotor. Actions, on the other hand, refer only to the capabilities of a device, such as sending a message or controlling LED lights.

Another taxonomy for IoT devices suggests that they can be systematically classified according to their communication range [5]. Long-range communication technologies offer the advantage of enabling data transmission from remote locations. This category of devices is also often referred to as Low-Power Wide Area Networks (LPWANs). In contrast, short-range communication technologies benefit smaller networks and devices located closer to each other, such as smart light bulbs or flood sensors in a moderately sized smart home.

Long-Range Communication Technologies

LoRaWAN (Long-Range Wide Area Network) [22] is widely recognized in the IoT space for its low power consumption and long-range communication capabilities. Allowing devices

to communicate over 15 kilometers in rural areas or even further, depending on the antenna and environmental obstacles. While its data transfer rate is rather low by modern standards, it remains suitable for transmitting numerical sensors' data. The operating frequencies depend on the region of use, typically 433, 868, or 915 MHz, with only 868 MHz and 433 MHz permitted in Europe. Devices using this communication technology need a gateway in order to capture the sent data from the device over the specified frequency, decode it, and hand it over to a desired platform. This behavior and the general gateway schema are shown in Figure 2.1.

Cellular networks [2], such as GSM, 3G, 4G, are another option for long-range IoT communication. IoT devices with corresponding cellular modules can connect directly to the internet via vast mobile networks that have already been built. That provides them with extensive coverage in urban areas, where the cellular signal is good. While these networks offer higher data transmission rates and broad coverage, they typically consume more power than other technologies, making them less ideal for battery-powered devices or for machine-to-machine communication. Nevertheless, cellular networks are ideal for use cases, such as surveillance and security systems, benefiting from high bandwidth and allowing customers to access live feeds from anywhere.

SigFox [2] is a low-power wireless technology tailor-made for sensor or machine-to-machine communication. Using Ultra Narrow Band (UNB) technology allows for communication ranges up to 50 kilometers. However, this technology is only designed for low data transfers, and its data transfer speed only ranges from 10 to 1000 bits per second.

Short-Range Communication Technologies

Bluetooth[2] [2] is one of the most commonly used low bandwidth, low latency, short-range communication protocols. Bluetooth has been widely used since 1999, when the first Bluetooth device was released. Nowadays, it is used in various consumer electronics, such as headphones, portable speakers, or smartwatches. Bluetooth Low Energy (BLE), which is a subset of Bluetooth, is a better alternative for IoT. It was designed to maximize battery life, thus reducing its power consumption as much as possible. Devices using BLE can last even multiple years on a single charge, based on their application. BLE also supports a star network topology with an unlimited number of nodes. The star topology [10], however, does not increase the range of the network as all nodes must communicate with the central device.

Wi-Fi [36, 27] is also widely used communication technology in IoT. When smart devices are connected within the same local area network (LAN), Wi-Fi provides quick and easy access. Wi-Fi also offers much greater transfer speeds, which is beneficial in many use cases. However, a key drawback is its relatively high power consumption compared to other short-range technologies, which can be a downside for battery-operated devices.

Zigbee [25] is a suite of high-level communication protocols that are very well-known in the smart home community. Zigbee gets a lot of its traction due to its energy efficiency, especially when compared to technologies like Wi-Fi, as it relies on low-power digital radios. These digital radios do not have high bandwidth, as they were designed for relatively small-scale projects, nor do they have the greatest range. However, this technology allows for meshing, allowing data to be relayed from one device to another, eventually reaching more distant devices.

[2]Bluetooth® Wireless Technology homepage – https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

Radio-frequency identification (RFID) [5, 41] uses frequency waves to transmit data between a passive tag and a reader. RFID tags contain a microchip and an antenna, which enables communication back to the reader. This technology revolutionized, along with others, the retail industry, making inventory management easier and allowing contactless payment systems to exist. Active RFID tags also exist, and they have a much greater communication range. However, these active tags need their own power source, usually a battery.

Near-field communication (NFC) [5, 39] is very similar to RFID and is indeed only a subset of RFID, but unlike RFID, NFC can operate exclusively over short distances within only a few centimeters. NFC's biggest success has to be contactless payments, but NFC tags are also becoming popular in smart home automations. These simple NFC tags can trigger automations upon tapping them with a phone.

Of course, these are not all communication protocols. Other notable examples include Ethernet, which provides reliable, high-speed communication over wired networks. Each protocol has its unique strengths and weaknesses, with trade-offs mainly in range, bandwidth, power consumption, or infrastructure requirements.

Gateways

With lots of wirelessly transmitting devices or sensors, typical Wireless Sensor Networks (WSNs) [25] are created. The sensors and other devices [46] contained in these networks are often not homogeneous, using different communication protocols or even technologies. The sensors [25] or other IoT devices ought to have very low power consumption as they often run off of batteries or similar energy sources, and their data transmission technologies and respective communication protocols follow that principle as well. Following these principles outright disregards many usual transmission technologies, such as Wi-Fi, due to their relatively high power consumption [27]. That is where gateways come in handy. They serve as a critical midpoint [46, 25] in a WSN and address the heterogeneity of different sensor networks and the diversity of protocols.

Gateways function as bridges, translating data between various short- and long-range communication technologies or different protocols into platform-compatible formats, such as the TCP/IP protocol of the Internet, defined by RFC 793 [1]. This process enables easy integration and centralization of devices within the platform. The general idea of gateways is depicted in Figure 2.1.

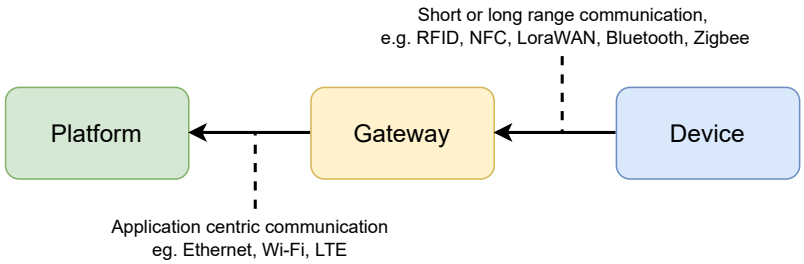


Figure 2.1: General gateway schema based on [11]

When gateways are used in a WSN [25], the architecture allows the end devices to be simplified and focused primarily on data operations. These devices can operate with minimal processing capabilities since the gateways handle the complex logical operations. The end

devices then follow an efficient life cycle that consists mostly of sleeping to conserve energy, and they only wake up to perform essential functions such as data collection, transmission, or reception. However, accurately representing the status, properties, and behaviors of these physical devices in a digital environment, such as a centralized platform, presents challenges. To address these challenges, the concept of Digital Twins can be used, which is further described in Section 2.4.

Furthermore, gateways [25] can be used in three different variations: active, passive, and hybrid. Active gateways are the most used, allowing the nodes to send their data to the gateway server. On the other hand, passive gateways request the nodes to send their data, and hybrid gateways combine these two mechanisms.

In the case of LoRaWAN, devices send data to a gateway, which then decodes the transmitted information and forwards it for further processing. One of the great representatives is The Things Network[3]. This network is a global collaborative ecosystem focused on IoT, utilizing the LoRaWAN communication technology, enabling low-power, long-range communication for end devices using gateways that are publicly accessible.

2.4 Digital Twins

A digital twin's [24, 44, 26] main goal is to continuously reflect or mirror a physical object or system into a virtual space of digital twin models. This allows the analysis of the physical system using its digital twins throughout its entire life cycle. This concept was first introduced by Michael Grieves and presented in 2003 at the University of Michigan. Since then, digital twins have gained much more traction both in academia and industry fields.

A digital model represents a specific physical twin by mirroring both its structure and behavior, including its geometric structure and real-time status, as well as background information and interaction interfaces, making the digital twin model a live replica of the physical twin. A vital feature in the relationship between a physical and a digital twin is close interconnection. This results in a need for the data to flow bidirectionally so that the digital twin can detect changes in the physical twin at all times and perform its actions.

Digital twins and the IoT are, by nature, closely related. Digital twins are one of the most promising technologies for enabling physical components to be connected using their virtual twins in the digital space as well. For instance, when designing parts, their lifespan can be fully simulated using a virtual model of the physical parts, allowing design defects to be found in advance. Or even whole production lines can be simulated to be more efficient.

The life cycle [26] of a digital twin can be subdivided into four primary phases, shown in Figure 2.2: creation/design, production, operations, and disposal. These phases closely represent the relationships between a physical object and its corresponding digital object as they evolve into an entirely identical digital twin.

- Creation/Design phase starts the life cycle of a digital twin as a logical object, which is essentially a blueprint of the physical object. This phase relies heavily on simulation tools to catch any flaws in the physical object prior to production.
- The Production phase is where the logical object is enhanced, and it begins to integrate with the physical object deeply. IoT sensors help establish a close connection between physical and logical objects.

[3]The Things Network website – https://www.thethingsnetwork.org/

- The operational phase marks the moment when a digital twin becomes fully functional, including bidirectional communication with the physical object. This phase oftentimes uses artificial intelligence or machine learning to monitor performance and predict failures of the physical object.
- The last phase of the life cycle is called the disposal phase. All the data collected from a physical twin is archived, and the digital twin serves as a repository for all relevant data collected during its lifespan, allowing for further analysis.

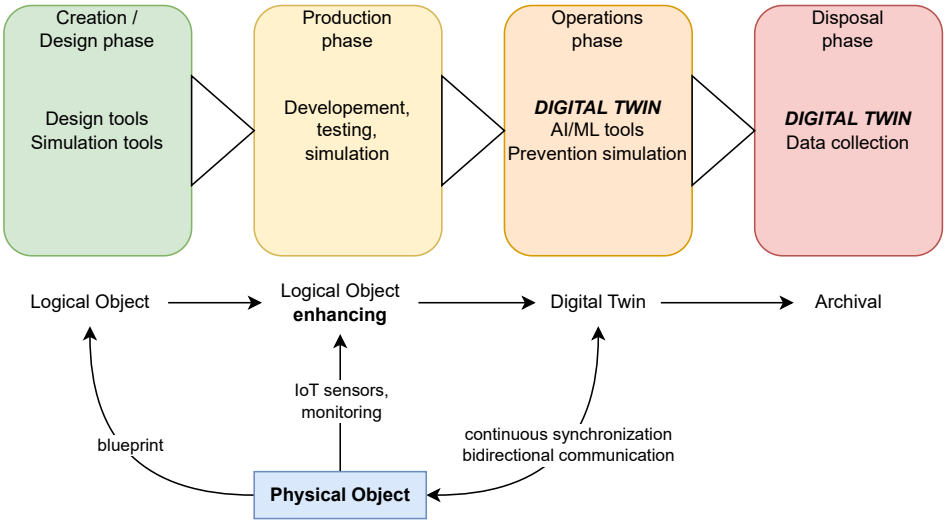


Figure 2.2: Life cycle phases of a digital twin based on article [26]

Data flows predominantly from the physical object to the digital twin, reflecting changes, states, and real-time updates. However, in some cases, the digital twin can send data to the physical object, for example, to correct errors or to synchronize.

Digital twins are used extensively across various fields, predominantly manufacturing, but also in smart cities, for instance. In smart cities, digital twins can help simulate multiple urban systems, such as transportation, energy, or water supply.

2.5 Smart Cities

Smart Cities [3] represent a development strategy that aims to improve residents' well-being through the integration of IoT technologies and the optimization of city services' efficiency using smart devices and sensors embedded in infrastructure systems to analyze extensive data sets effectively to address many common issues. The concept of Smart Cities is closely tied to urban IoT systems and e-governance. The urban IoT systems provide communication technologies designed to support added-value services for both the administration of the city and the citizens [43].

Through e-governance [7] integration, governments can enhance public services or improve interactions with citizens. E-Government connects to the government's use of information and communication technologies and promotes more efficient governance, creating a more technology-oriented relationship between governments and citizens. Integrating

e-governance into a Smart City is not a straightforward task, and this process has several requirements as described in the book [40], such as e-literacy programs, e-democracy, e-participation, e-voting, and more.

Usages

For a long time, typical energy grids only had a one-way energy flow, from a generating source to a customer [35]. Today, this model is evolving as more customers install their own renewable power sources and smart electric meters, generating their own energy, and having the ability to supply energy to public grids as well. In response, smart cities are creating smart grids and developing systems to better analyze the real-time situation of electric grids. These grids use sensors to make electric grids more observable and allow distributed energy generation, and even add self-healing capabilities. As a result, power companies gain access to more accurate data, which can help in preventing power outages. Incorporating information from solar farms has also proven to be a strategy for maximizing the use of sustainable energy sources within smart grids.

Smart transportation [35] is also a big part of smart cities, including smart parking, traffic analysis, public transport planning, and much more. Smart parking aims to help citizens find free parking spaces in parking lots equipped with sensors. Using these sensors, free and occupied parking spots are detected, saving citizens' time and reducing emissions.

Traffic congestion is another major concern of urban transportation systems [43]. Though many cities already use cameras to track traffic, using other low-power and widely used communication technologies could offer more thorough information. This method makes use of various sensors, such as acoustic or air quality sensors, and even connected vehicles equipped with GPS. City officials use these insights to monitor and examine driver behavior or traffic trends. Citizens can also benefit from these data [35, 43], as they allow them to schedule and plan their trips more efficiently, resulting in reduced travel times and even their carbon footprint. In fact, modern navigation apps, such as Waze and Google Maps, already use user-generated data in real-time to recommend optimal routes [35].

Integrating IoT into waste management [35, 43] is another big leap ahead in smart cities. Sensors tracking waste levels in bins or sewers help waste management companies switch from scheduled to on-demand pickups and optimize collection routes, saving resources while still ensuring timely disposal. In combination with rain and weather sensors, sewer overflow monitoring is performed. Preventing overflows by anticipating ahead of time.

Chapter 3

Dashboards

In the early 2000s, dashboards started to gain a lot of attention [15, 14], and with this attention came opportunities that marketers quickly took advantage of. As dashboards gained their honorable spot within business intelligence and data visualization, their definition became vague and often ambiguous. By 2004, the term had grown into a buzzword, and some vendors were tailoring its meaning to suit their offerings, creating more confusion in the industry. For instance, an IBM account manager once said, "By data warehousing, we mean whatever the customer thinks it is" [15]. The same ambiguity problem also took over the dashboards.

Despite confusion in the early 2000s, most vendors seemed to agree that dashboards must include a graphical display mechanism such as traffic lights, gauges, or meters, many similar to those commonly found in automobiles, as people were already familiar with these graphical displays. Stephen Few [13, 15] then defined the dashboard as follows:

> "A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance."
>
> ---
>
> *Stephen Few*

This definition emphasizes the clarity and purpose of dashboards while distinguishing them from other forms of data presentation. Dashboards are not only collections of graphical elements, but purpose-built custom tools to efficiently communicate information that may not be seen easily. They typically consist of a mixture of text and graphical elements to represent key performance indicators (KPIs), discussed in Section 3.4, and other critical information.

In addition, dashboards have several key attributes [13], starting with their visual presentation style. Visual representation does not use graphical elements for aesthetic purposes; instead, if their graphical representation is done correctly, it enables much faster rates of comprehension and richer meaning than text alone. Of course, there are cases where textual representation is more efficient, accurate, or even necessary.

Dashboards provide objective-oriented information, meaning they consolidate different types of information that may not have been otherwise related and often come from various business functions. The information required is, in many cases, a set of KPIs. However, dashboards can accommodate other data types as needed to help one's job.

A key feature of dashboards is the ability to monitor information at a glance, with a primary focus on summaries or exceptions that highlight important points that need attention.

Their main feature is not providing extensive details needed to take action, but instead serving as a tool to quickly alert users to actionable insights. Getting into the details may require shifting to other displays beyond the dashboard itself, using navigational methods through techniques like drill-down interactions.

To support quick monitoring, dashboards must avoid scattering information across multiple locations. Instead, a dashboard should present all critical information on a single screen or within the viewer's immediate field of vision. The main objective of a dashboard is to have the most important information readily available and effortlessly absorbable.

3.1 Types of Dashboards

Nowadays, dashboards have become essential to many businesses, and, of course, they differ dramatically in many aspects, such as their primary roles, types of data displayed, and data domains. Dashboards, in general, can be categorized in numerous ways [13]. The categorization itself helps to understand and leverage their potential in different contexts. For example, categorization helps identify whether a dashboard is best suited for strategic decision-making, operational monitoring, or analytical deep dives, each requiring a different layout and data prioritization. As with IoT, there are a great many taxonomies for their classification based on one or more variables.

One of the most important and easily imaginable classifications is based on the role of the dashboard [13], which is determined by the type of business activity it supports. This resulted in the creation of three main categories: strategic, analytical, and operational. W. Eckerson [12] divides dashboards similarly to S. Few; however, he includes analytical dashboards in a bigger domain, called tactical dashboards. Tactical dashboards are then composed of another three types: enterprise dashboards, mashboards, and, the already mentioned, analytical dashboards. W. Eckerson also researched the usage of the respective types of dashboards and revealed interesting results shown in Figure 3.1. From his research, it is apparent that almost two-thirds of the surveyed organizations utilize all three types of dashboards. Among these types, tactical dashboards emerged as the most widely adopted.

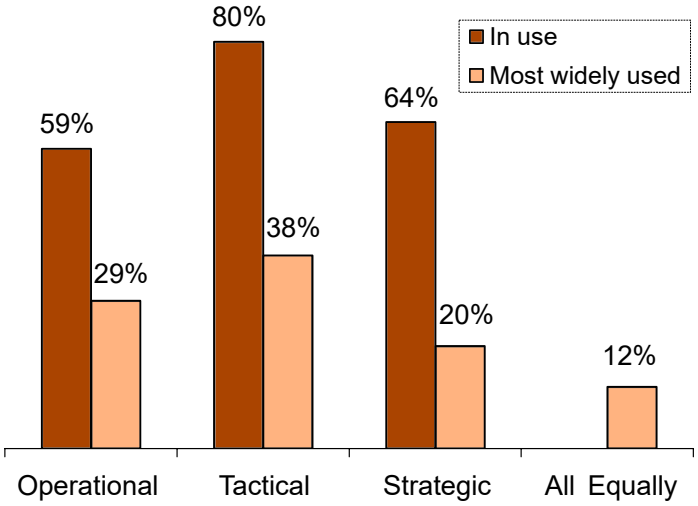


Figure 3.1: Dashboard usage for respective types based on book [12]

Strategic dashboards [13] are primarily designed for high-level decision-making. Most so-called "executive dashboards" and also many dashboards that support managers at any level in an organization, fall into this category. They offer brief summaries to track a company's performance and general health, often including forecasts as well.

One significant aspect of strategic dashboards is that they typically display static data snapshots, usually updated on a daily, weekly, or monthly basis. These dashboards are meant to direct long-term strategies rather than respond to changes in real time. For instance, they could show yearly growth comparisons or quarterly revenue trends.

Simple display mechanisms such as summary charts or basic indicators are the most effective in strategic dashboards. Due to their high-level nature, interactivity is often minimal.

Analytical dashboards [13, 12] are designed for users who want to explore data in-depth to discover insights that might be hidden at first sight or understand the underlying trends. In general, these dashboards demand greater context and include rich comparisons, historical data, and precise performance metrics. In a business setting, these dashboards are mainly used to optimize processes in individual departments, such as finance, sales, or marketing. Managers use them primarily to analyze performance against a set goal using a combination of detailed and summary data.

Similarly to strategic dashboards, analytical ones also benefit from static snapshots of data. On the other hand, opposing strategic dashboards, they should allow for interaction with the data. Interactive elements enable users to drill down into specific data points or examine detailed views. This interactivity supports complex analyses, enabling users to answer questions such as *why* sales are declining or *why* the number of customers is greater, and thus examine the causes, not only to see what is happening.

Operational dashboards [13, 12] are designed to monitor operational processes, events, or activities as they occur. Thus, they must be designed differently from strategic or analytical ones. Operational dashboards are designed to monitor real-time activities and provide immediate awareness of issues that require swift action. Their primary role is to alert users to changes or anomalies in metrics that fall outside acceptable thresholds, such as machines running out of supplies or significant drops in website traffic. In a business environment, these users are mostly the "front-line" personnel.

As in strategic dashboards, the display features must be kept very simple, clear, and action-oriented. Their straightforward looks help prevent mistakes in emergencies when rapid decisions are made, and unclear or complex visual displays tend to cause more mistakes. Another important aspect of operational dashboards is their higher level of information detail compared to other types of dashboards. Oftentimes, they are more interactive, and features such as drill-down capabilities or hover-over interactions allow access to deeper levels of details easily if necessary.

There are numerous other taxonomies for dividing dashboards into logical groups. Other examples include divisions based on interactivity or display mechanisms. These and a few more interesting taxonomies can be found in Table 2-1 of the book [13].

3.2 Dashboard Design Principles

Creating the perfect dashboard for a specific use case is not a straightforward task. Unlike straightforward data visualizations, dashboards are sophisticated tools that need to strike a balance between visual clarity, information density, and user-specific functionality. In addition, designers are often provided with a variety of data streams that can be processed,

abstracted, or simplified. Choosing the appropriate combinations can be quite overwhelming due to many data streams, and a wide range of visual representations [4].

The fundamental challenge when it comes to dashboard design is condensing relevant and often dissimilar information into very limited space while still maintaining clarity. This is the main challenge, though there are others, such as choosing the appropriate data or visual display mechanisms. Stephen Few collected the requirements for a well-designed dashboard in his book [13] into four key points.

- Exceptionally well organized
- Condensed, primarily in the form of summaries and exceptions
- Specific to and customized for the dashboard's audience and objectives
- Displayed using concise and often small media that communicate the data and its message in the clearest and most direct way possible

Data and Non-Data Pixels

Data and non-data pixels [13] are a critical concept in visual information design, originally introduced by Edward R. Tufte in 1983 [38]. At the time, Tufte referred to this idea as the *data-ink ratio*, mainly applying it to printed materials. This principle was adopted for dashboards with the same idea, just replacing "ink" with "pixels". The definition of "data-ink ratio" is the proportion of ink that displays data and the ink that presents visual content or not data. The general idea is to minimize non-data pixels to a reasonable minimum. Examples of typical non-data pixels representatives in dashboards [13] are:

- Excessive grid lines in charts
- Color gradients in backgrounds
- Redundant borders
- Decorative graphics

The goal is not to eliminate all non-data pixels, but to keep them to a reasonable minimum. Another key step to display information effectively is to de-emphasize the remaining nonstructural elements. The whole visual design should be minimal, allowing the data to take priority.

Enhancing data pixels is also important, and it follows a similar path as non-data pixels. The rule "Less is more" applies directly to this process. Elimination of unnecessary data pixels is more challenging, but should be given the same weight as the removal of non-data pixels. Including careful consideration of what information is truly needed. Other elimination methods include creating summaries, condensing data, and focusing on exceptions.

Exceptions provide an especially useful means of reducing the data on a dashboard. Many times, there is no need to display the state of something if everything is well, but the user should still be alerted if it is not. When highlighting important data pixels, we should consider categorizing the most important information into two categories:

- Information that is always important
- Information that is only important at the moment

These two categories of important information on a dashboard require different visual strategies. The consistently important information can be statically addressed and will always be highlighted. Momentarily important information may need dynamic highlighting.

Information Layout

Data placement is another key factor when considering the design of a dashboard and the order of its elements. It directly influences how users perceive and categorize visual content by priority.

The main principle of perception is the regions of emphasis. Spatial placement of important information can naturally draw the user's attention to where it should be drawn. According to Few [13], the areas of greatest emphasis on a dashboard are the top-left and center sections. This emphasis on the top-left is based on the principles of Western languages, which are read from left to right and top to bottom. The center one comes from natural visual perception inclination. Article [6] confirms the same principles and describes this phenomenon as the "Z" layout pattern. Figure 3.2 shows all areas of emphasis with their respective weights.

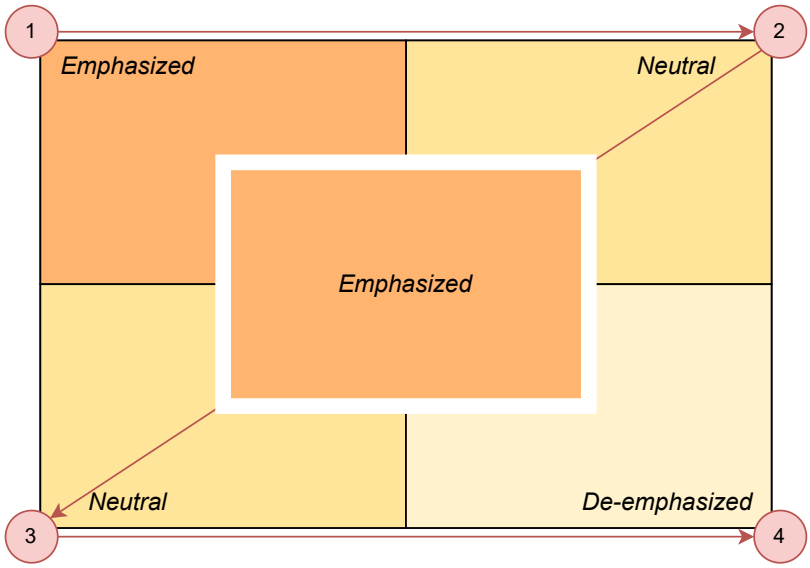


Figure 3.2: Illustration of emphasized areas based on book [13] and the "Z" layout pattern based on article [6]

Colors and Color-Blindness

When choosing a color scheme for a dashboard, we need to keep in mind that quite a portion of users can have some form of color-vision deficiency [42]. To be specific, color-vision deficiency affects approximately 1 in 12 men (8%)[1] and 1 in 200 women (0.5%).

People with impaired color vision are *not* unable to see colors. Instead, they typically have difficulty distinguishing between certain types of color, such as red and green or blue and green [42]. The correct terms for these deficiencies are *deuteranomaly/deuteranopia* and *protanomaly/protanopia* for the red-green insensitivity and *tritanomaly/tritanopia* for the blue-yellow. The terms with suffixes "anomaly" refer to impairment in

[1]Color Blind Awareness site – https://www.colourblindawareness.org/

the perception of respective colors, while "anopia" refers to the complete absence of perception. Figure 3.3 shows an example of these impairments and their effect on people's perception.

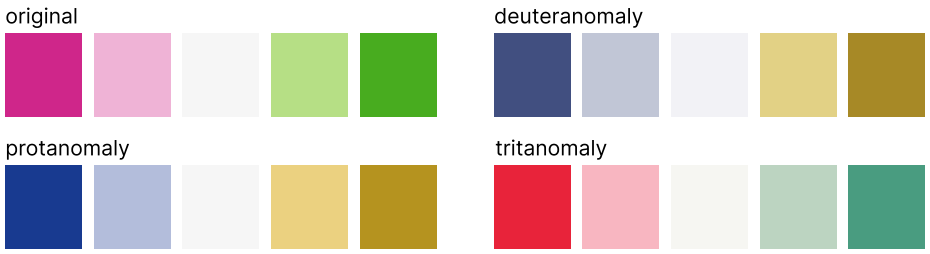


Figure 3.3: Color perception impairments examples based on book [42]

The referred book [42] dives deeper and later proposes a color palette for most color-vision deficiencies. This color palette is shown in Figure 3.4 below.

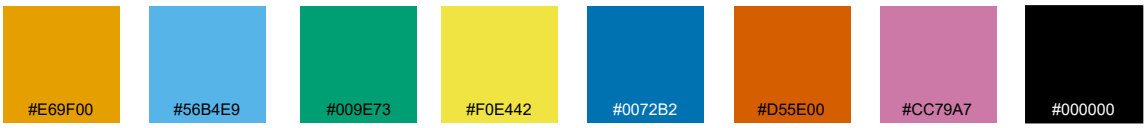


Figure 3.4: Colorblind-friendly color palette based on book [42]

Colors, in general, can be divided into vivid and subtle, and each category has its place. Subtle colors are rather soothing, while vivid ones can truly grab the viewer's attention [13], but only if using colors for this purpose is done sparingly. Using colors to differentiate sections or using colored backgrounds is not generally recommended, as it creates unnecessary non-data pixels and should generally be avoided. Instead, using white space as a divider based on Gestalt principles described in the section below is often a better option.

When using colors, we must consider limits to perceptual distinctness or other visual attributes to differentiate elements. For instance, when using varying intensities [13], or luminance [28] of gray in a line graph, it is essential to keep them distinct enough from each other to be easily recognized. The practical limit for discernible intensities is about five or even less if the background is not uniform [28]. This principle also extends to color hues. Although most people can easily differentiate more than five hues of a single color, our short-term memory cannot retain the meanings of more than nine in total [13]. What should be noted is that the ability to differentiate these colors easily decreases significantly if the colored regions are not contiguous. When hues are applied to small, separated regions, the number of easily distinguishable hues drops to around six or seven [28].

Gestalt Principles

Gestalt principles are based on psychological research conducted in Germany in the early 1900s [12]. These principles primarily focus on understanding how people connect and categorize objects based on their placements and proximity to each other. Researchers revealed their findings, and those include six key principles of visual perception that are still relevant for design today [13].

- *Proximity* – objects that are close to each other are perceived as the same group and can guide the viewer's vision as shown in Figure 3.5.
- *Similarity* – objects that share visual attributes such as color, shape, or orientation are grouped together.
- *Enclosure* – perception of objects enclosed by a visual border or colored background is perceived as a distinct group.
- *Closure* – even unclosed borders or circles are perceived as fully closed, complete structures, as shown in Figure 3.6. Often, a full enclosure is unnecessary, and only two axes are sufficient.
- *Continuity* – objects aligned or appearing to continue one another are perceived as a single group.
- *Connection* – objects connected by lines are strongly perceived as belonging to the same group. This principle is more powerful and outweighs proximity or similarity.

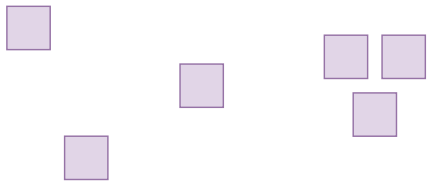


Figure 3.5: Principle of proximity

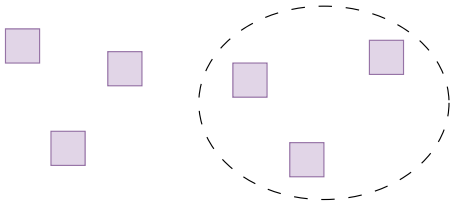


Figure 3.6: Principle of closure

3.3 Data Visualizations Suitable for IoT

Visualizing the vast and diverse data produced by IoT devices requires several visualization mechanisms. The data itself can be divided into two types: quantitative and non-quantitative. Quantitative data is predominant in the IoT and consist of numerical values such as temperature, humidity, or energy consumption. However, non-quantitative data such as device status, environmental conditions, or geographical labels require different visualizations.

Textual Values

Detailed graphs or comprehensive visualizations may be unnecessary and space-consuming in various IoT applications. Textual values [13] may be a better choice for communicating some information. They are particularly suitable for reporting the state of a single measurement or data point, such as the current outside temperature. With their compact nature, they are commonly incorporated into dashboards.

This visualization is not suitable for any data that requires context, for example, in retail, where just one value is often insufficient. Without displaying trends over time or a percentage indicator, presenting the current number of units sold as an isolated figure would not provide actionable insights.

Tables

Tables, created by organizing textual values in a structured format, are a powerful but often underappreciated tool in data visualization [42]. Tables provide a clear and efficient way to display a set of data in great detail. Unlike graphs, which are less effective for retrieving individual data points, tables allow users to quickly and accurately find specific information [13]. When properly formatted, they make searching for relevant information easy and intuitive.

Bar Graphs

Bar graphs are very easy for viewers to interpret and analyze. Compared to other visualization types, such as a pie chart, they are far more efficient and accurate [13]. Bar graphs [42, 12] are highly effective for comparing values across one or more categories along a single measure, quickly showcasing relationships such as the relative sizes of values from largest to smallest. An example bar graph is shown in Figure 3.7. Furthermore, when incorporated with legends and colors, it enhances their ability to display multiple categorical attributes.

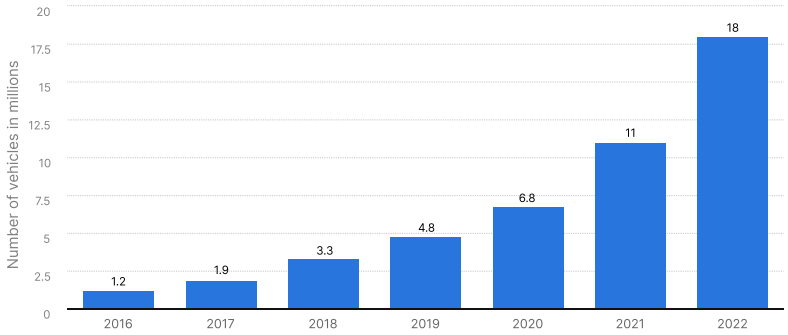


Figure 3.7: Bar graph depicting number of electric vehicles based on website[2]

Box Plots

Data from IoT devices can vary significantly, and bar graphs can sometimes be misleading. Unlike bar graphs, box plot uses statistical summaries such as median and interquartile range that are robust in the presence of skewness and outliers [29]. Moreover, displaying a single measure from a set of values with aggregation functions, such as sum or average, may not be adequate [13]. Instead, box plots display how these values are distributed, while also providing the shape of the data. Box plots were first invented by John W. Tukey in the 1970s, and since then, many more variants have been introduced. Figure 3.8 illustrates an example.

[2]Electric vehicles statistics – https://www.statista.com/statistics/270603/worldwide-number-of-hybrid-and-electric-vehicles-since-2009/

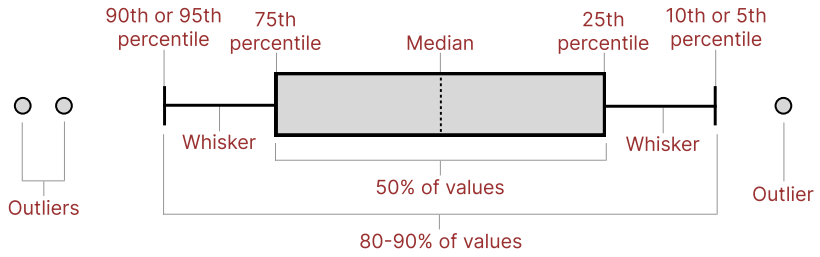


Figure 3.8: Box plot with labeled components based on book [13]

Line Graphs

Line charts are widely used in dashboards, and they have the ability to highlight different properties of the data compared to bar charts. While bar charts emphasize individual values and allow for quick comparisons between adjacent values [13], line graphs excel at displaying the shape, trends, changes, or patterns of the data over time [42].

Some line graphs can even include a so-called confidence band to better reflect the reality of the uncertainty of the data analyzed [42], such as in the example shown in Figure 3.9.

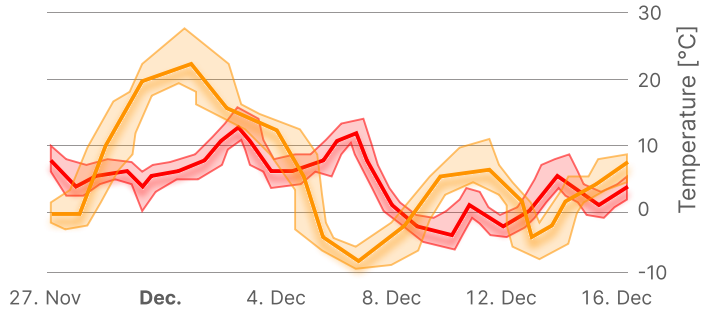


Figure 3.9: Line chart with confidence intervals for respective data sources

Sparklines

Sparklines were first introduced by Edward R. Tufte, and he published this idea in a dedicated chapter of a book called Beautiful Evidence in 2006 [37]. He described the idea of sparklines as data graphic elements that work at the resolution of routine typography with their attributes of being small, data-intense, and word-size graphics.

Sparklines are very clear and space-efficient, providing just one piece of information to the viewer, and that is the overall trend of the data, thus providing historical context [37, 13]. To this date, sparklines are very popular in the financial sphere, as shown in Figure 3.10 below.

Symbol	Name	Price	Change	Change %	Volume	Avg Vol (3M)	Market Cap	P/E Ratio (TTM)
NVDA	NVIDIA Corporation	137.49	+0.48	+0.35%	166.551M	223.106M	3.367T	-
TSLA	Tesla, Inc.	417.41	-14.25	-3.30%	62.625M	92.519M	1.34T	114.99

Figure 3.10: Sparklines in stock markets[3]

[3]Picture is based on Yahoo finance stocks page – https://finance.yahoo.com/markets/stocks/

Bullet Graphs

To address common issues found in traditional visualizations like gauges or meters, Stephen Few [13] introduced a new type of graph. People are used to seeing gauges and meters in their daily lives, however, they often occupy too much space on a dashboard, where space is limited. The primary purpose of these traditional quantitative data visualizations is to display a single data point, which is sometimes compared to a target value or evaluated against qualitative labels that indicate whether the data point is good or bad.

Bullet graphs, illustrated in Figure 3.11, achieve the same goals as the mentioned traditional visualizations, but take up significantly less space. They come from the general idea of bar graphs that emphasize an individual value. Additionally, they incorporate qualitative ranges encoded with background color fills and a symbol marker for comparative measure, which equates to "Target" in the mentioned Figure 3.11 below.

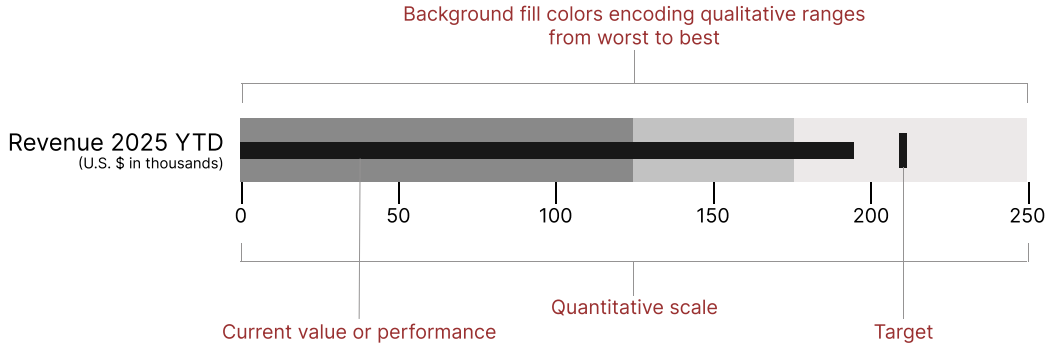


Figure 3.11: Bullet graph with labeled components based on book [13]

3.4 Key Performance Indicators

Not every state has to be displayed in detail on a dashboard, and many times, higher-level overviews are more practical and sufficient. This is where key performance indicators (KPIs) play a vital role. W. Eckerson defines KPIs as a crucial metric that significantly influences business performance [12]. Data collection is crucial for business analysis, but it often requires further transformation or interpretation to extract meaningful insights. That is where KPIs come in handy, and a well-designed one can trigger a chain reaction of process improvements throughout the whole business.

D. Parmenter, in his book [30], says that KPIs, in general, should be monitored always, daily, or weekly in some cases. If a KPI were to be monitored monthly, quarterly, or annually, it cannot be a *key* indicator to a business, but rather a performance indicator reflecting past outcomes.

In general, there are two major types of performance indicators: outcome metrics and driver metrics [12]. Outcome metrics, also known as *lagging indicators*, measure the results of business activities based on a strategy. These metrics are typically backward-looking and provide insights into the fulfillment of goals, such as revenue growth or customer satisfaction. On the other hand, driver metrics or *leading indicators* measure business activities that have the ability to influence the results of the outcome KPIs. These metrics are tactical and allow businesses to adjust their strategies to meet or exceed outcome goals for some period.

For instance, in a business setting, a KPI can measure the total number of weekly sales to monitor the effectiveness of a strategy. In the context of IoT, KPIs can be used to monitor important specific metrics of devices. For example, a temperature threshold KPI inside a greenhouse can monitor the ideal set temperature.

3.5 User Experience

In recent years, the term *user experience* (UX) has gained a lot of traction as an important aspect of technology design and interaction. Although UX is used mainly as the "usability" paradigm [21], it is much more complex. This paradigm is composed of many variables, and generally, these variables can be broadly divided into three groups: the user's internal state (e.g., expectations, motivation), characteristics of the designed system (e.g., complexity, purpose), and the context or environment within which the user interactions occur. In the context of dashboards, we can only focus on the category of the characteristics of the designed system, as we cannot alter the user's internal state, nor their context or environment.

Aesthetics

Unattractive visual displays can negatively affect the user's mindset, making them less susceptible to their use and also decreasing the dashboard's ability to communicate [13]. This certainly does not mean that some plain decorations should be used on a dashboard, but rather, the data should be displayed attractively, without distractions.

Aesthetics in data visualization are essential to provide graphical representations of data that are both informative and visually pleasing. Claus O. Wilke [42] describes aesthetics in data visualization as follows:

> "All data visualizations map data values into quantifiable features of the resulting graphic. We refer to these features as aesthetics."
>
> Claus O. Wilke

These aesthetics, as Wilke describes, are commonly used attributes such as positions, shape, size, color, line width, or line type when using visual displays to interpret data.

Responsive Design

Today's variety of devices, from smartphones to large desktop monitors, makes responsive design [17] another essential factor for dashboards. Responsive design ensures that a page or a singular element can adapt its layout and content to viewing contexts across a spectrum of digital devices. The main goal of responsive design is to make the result a more satisfying experience for any user, regardless of the device used. Creating pages or elements with fixed width, such as a minimum target resolution in web pages, creates both short- and long-term problems. Short-term problems limit users with more capable hardware and limit their user experience. In the long term, future redesigns are almost guaranteed to accommodate changes in the target resolution. Responsive design in web pages relies on three key principles:

- *Fluid layouts* allow scalable layouts by using relative units, such as percentages, instead of fixed ones. Fluid layouts are dynamic and automatically adjust to the available real estate on the user interface.
- *Flexible media* ensures that the content within a layout resizes as well within its container, avoiding overflow issues on devices with smaller screens.
- *Media queries* are a feature of modern cascading style sheets (CSS). They allow designers to apply specific styles based on the user's device properties, such as screen width or orientation.

Applying these principles to dashboards is not an easy task. Doing so might lead to breaking Stephen Few's principle [13], of a single-screen dashboard, where all the information should be available at a glance. This principle is difficult to maintain, mainly on smaller devices and especially when dealing with numerous smart devices on a single dashboard. In such cases, vertical scrolling may be necessary to ensure that all data is easily accessible.

Chapter 4

Data Visualization Requirements Analysis

Before digging into the analysis of requirements for IoT data visualizations, we shall clarify our user domain. Dashboards are used in almost any sphere imaginable, and so is IoT. Visualizations can differ dramatically, and they also should, as a data analyst for a power plant, will require highly detailed, precise, and customizable visualizations for in-depth analysis, while other users may prioritize simplicity with immediate values. Presenting users with non-suitable visualizations can discourage their engagement or even lead to misinterpretation of the data.

The user domain that is the primary focus in this thesis is mainly smart home users. The needs of smart home users are mainly intuitive and actionable data displays, and unlike data analysts, smart home users benefit mostly from clear and straightforward visualizations. In an operational dashboard, as discussed in Section 3.1, its data visualization elements can include summaries like total energy usage or production from solar panels, or the current temperature in a room.

4.1 Groups of Devices

As the number of connected devices in smart homes increases, it is common, and many times unavoidable, to organize them into logical groups, such as by rooms, device types, or based on their function. Doing so allows easier management, monitoring, and oftentimes makes it possible to take control of their collective status, like shutting off all the lights.

There are numerous ways of group visualization, but card-based summaries are among the most used. Cards are common even for other purposes on dashboards, as they create a clear differentiation between visual displays. When used for groups, key metrics are displayed, such as total power usage, number of active devices, or average temperature throughout the house.

Among other visualizations are interactive maps and floor plans, which allow users to create maps of their homes and then place individual devices into their corresponding locations. These maps can be great for beginners, as they are intuitive to use and provide a clear visual representation of where each device is located within their homes.

Nevertheless, this visualization comes with several significant limitations. It takes a lot of time to create such a map, as each device must be manually placed within the home layout, which also has to be defined. They also occupy a large amount of space on a dashboard,

often taking up an entire page and limiting the visibility of other important information. Even though their visual overview can be useful, interactive maps usually lack thorough information without further action. To access a device's current state, users are usually expected to use hover or click actions on individual icons or similar elements. Controlling an entire group, such as a whole room, is not common in map-based visualizations and requires users to control devices individually.

As covered in Section 3.4, key performance indicators can be quite effective in tracking groups of devices. These indicators allow for setting threshold values that can alert to critical situations, for instance, a discharged battery. Doing so notifies users without requiring them to physically check every device for its state.

4.2 Historical Data of Individual Devices

Historical data is as important as current data. Having historical data at hand enables users to analyze trends and patterns and make decisions based on previous results. The data itself can be categorized into quantitative and non-quantitative types, and each of them requires different visualization approaches. Quantitative data, such as temperature readings, energy consumption, or luminance levels, are typically represented numerically. A single reading then creates a specific data point at a given time.

Among the most commonly used visualization techniques for quantitative data are line charts. Line charts excel at showing trends and the overall shape of readings. With their abilities, they can be found on almost any dashboard and are very well-known among people. Some dashboards display them in general view, while others only show them after clicking on a device that is outputting quantitative data based on the user's preference.

In addition to line charts, other visualizations, such as bar charts, sparklines, or even simple textual values, are suitable depending on the requirements. These charts, among others, were further described in Section 3.3. These visualizations can also be accompanied by percentage changes as trend indicators, alert symbols, or icons, which are more suitable for color-blind people [13], or even changes in colors based on values.

The other type of data, non-quantitative, such as device status or Wi-Fi network name that is being used by the device, requires a different approach. These values are usually not represented numerically, but rather categorically, often using textual strings. A common choice for the visualization of non-quantitative data is timelines or icons. Timelines present events in a clear chronological order, making it easy to track changes over time. They're simple but effective, allowing users to quickly see how the state or behavior of a device has evolved over time.

A more immediate and intuitive way of telling the current state of a device is through the use of icons and coloring. Dashboards frequently use icons, sometimes just for aesthetic reasons, but when combined with a device identifier, such as a name, they offer a concise and understandable visual status of a device.

4.3 Existing Solutions

In order to design effective and performative data visualizations for smart home dashboards, it is essential to analyze current platforms that are focused on a similar user domain.

This chapter examines four widely used solutions: Home Assistant[1], OpenHAB[2], Google Home[3] and Logimic ACADA[4]. While the first three platforms are clearly oriented toward residential smart home users, Logimic ACADA operates in a different domain, focusing more on industrial and commercial applications. Nevertheless, its alternative interface strategies and visualization techniques still offer valuable insights and a foundation for scalable design solutions.

Each of the mentioned platforms offers unique visualizations of real-time and historical data. Moreover, all the solutions and their designs are responsive to an extent. Home Assistant and OpenHAB are open-source systems that provide extensive customization options, allowing users to create tailored dashboards for monitoring and controlling devices based on their wants and needs. Both of these platforms may not be suitable for all users, and some may find these solutions overwhelming. Google Home, on the other hand, focuses on a user-friendly and streamlined interface that provides minimal customization. Google Home is especially designed for seamless integration with Google's ecosystem and emphasizes ease of use. Lastly, Logimic ACADA specializes mainly in industrial and enterprise-level monitoring, offering robust and scalable solutions for maintaining whole sensor networks.

Home Assistant

As discussed, Home Assistant allows for almost any thinkable customization. There are countless official and third-party add-ons for device integrations, as well as for customization of the visualizations and the overall theme of the system. The flexibility of this system makes it a popular choice among advanced users and smart home enthusiasts who want to centralize all their devices.

Home Assistant comes with a set of visualizations that covers both quantitative and non-quantitative data. Line charts, as shown in Figure 4.1, are the most predominant and allow users to track trends, such as temperature or humidity. Furthermore, when hovering or clicking on a data point in the graph, the exact value is displayed along with the reference date. Line charts are displayed upon clicking on an entity with quantitative data.

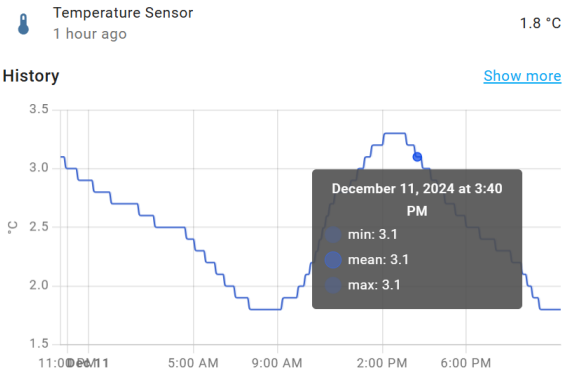


Figure 4.1: Example line chart depicting temperature changes

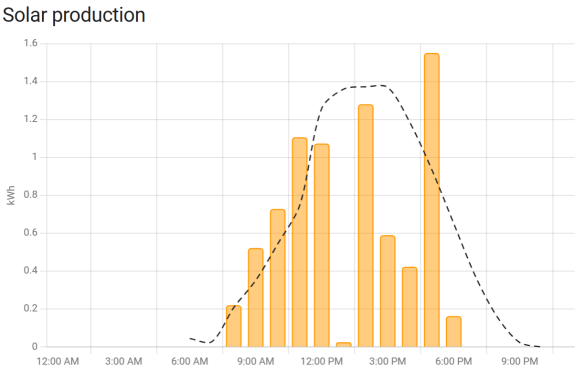


Figure 4.2: Hourly solar production yields shown using bar graph

[1]Home Assistant homepage – https://www.home-assistant.io/
[2]OpenHAB homepage – https://www.openhab.org/
[3]Google Home homepage – https://home.google.com/welcome/
[4]Logimic homepage – https://www.logimic.com/

Bar graphs, as shown in Figure 4.2, are also available, and users can customize some of their aspects, such as the time range, the type of statistics displayed (e.g., averages, totals, or maximum values), or the use of logarithmic scales.

Card elements are heavily used within the dashboard to visually separate groups of devices that belong together or just individual entities. Together with sparklines, as shown in Figure 4.3, they create a compact and visually pleasing way to present the data while still giving historical context and real-time value.

When historical context is unnecessary, immediate values can be displayed to provide a concise snapshot of the current state. As shown in Figure 4.4, this visualization combines the entity name, an optional icon, and the current value of the entity. This format is versatile, supporting both quantitative data and non-quantitative data.

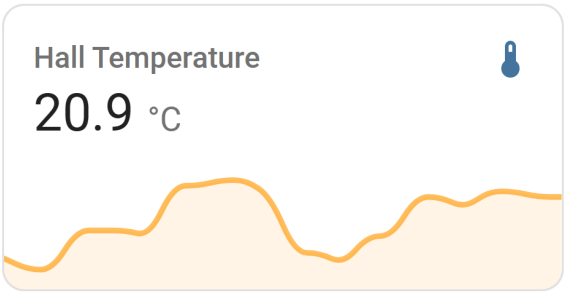


Figure 4.3: Example sparkline card

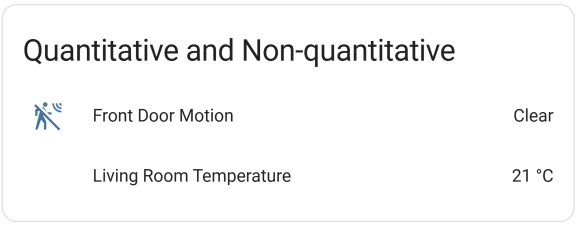


Figure 4.4: Example card of immediate values

Moreover, a detailed short-term history of sequential state changes, shown in Figure 4.5, is displayed upon clicking on a device with non-quantitative data. This visualization again leverages hover or click actions, and the exact time frame of a state is displayed upon interaction. A simplified version of the history is also shown in a log book, which does not provide the exact times and durations of individual states.

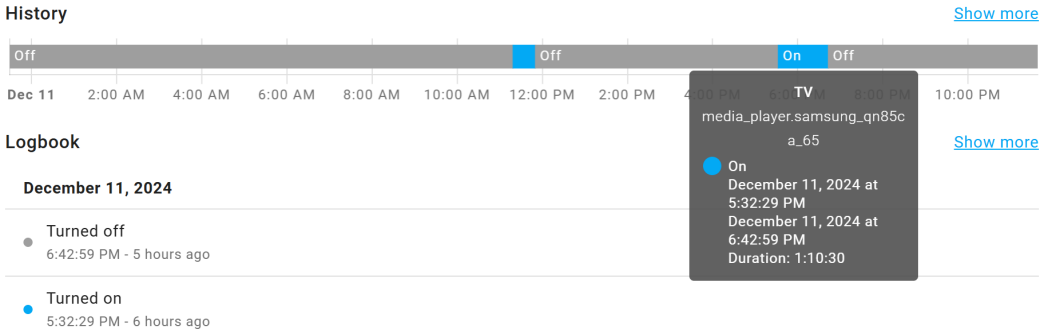


Figure 4.5: Sequential states history of a media player

Groups of Devices

Groups in Home Assistant allow users to join multiple devices or entities into a single entity. However, this approach of grouping devices carries significant drawbacks as well. Creating a group, such as one for all lights within a room, as shown in Figure 4.6, allows users to turn them on or off and adjust brightness levels simultaneously using just one group entity.

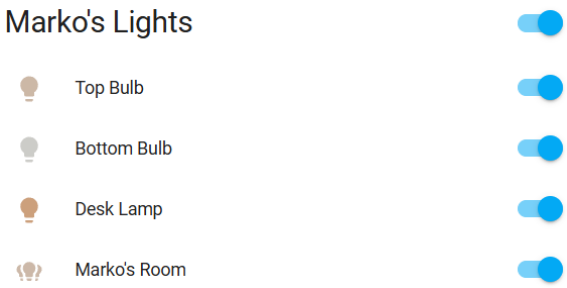


Figure 4.6: Example of a group, where the bottom entity is a group of lights in a room

In automation, similar benefits occur, as there is no need to list all the entities but rather just one group entity, thus reducing redundancy. For example, setting a rule to turn off "All Lights" when leaving home is easier when lights are grouped. In visualizations, groups reduce the number of entities by consolidating devices into logical units, making the display clearer.

One of the primary drawbacks of grouping devices into a single entity is the loss of granularity. For example, a group of lights will display an aggregate state such as "On", even if only one light is actually turned on. Additionally, if one of the lights becomes unavailable, the group entity remains unaffected and does not notify the user of the issue. The group entity will only display as unavailable if all devices in the group are unavailable, which can obscure potential problems with individual devices. Furthermore, the visualization of groups lacks detailed information about individual devices. Even after interacting with the group entity, users are not presented with specific states or conditions of the devices within the group, limiting the utility of this feature for troubleshooting or monitoring.

OpenHAB

OpenHAB and Home Assistant share many similarities, and both can be extensively customized. However, OpenHAB places a greater emphasis on configuration through textual files, whereas Home Assistant primarily uses a more user-friendly, UI-driven configuration. Resulting in a steeper learning curve, but allowing finer levels of control.

OpenHAB, just like Home Assistant, supports a wide range of *things*[5], through its *bindings* feature, which is equivalent to integrations in Home Assistant. However, once again, the process often requires more manual configuration than Home Assistant. When creating a dashboard, users are presented with two separate concepts: *sitemaps*[6] or *HABPanel*[7].

Sitemaps are set up using text files in which users specify hierarchical layouts according to a particular syntax. This method is lightweight, however, its lack of flexibility in design and absence of any GUI editor can feel outdated.

On the other hand, *HABPanel* is a graphical interface that lets users create customizable dashboards. OpenHAB designed HABPanel specifically for creating dashboards that are meant to be used for wall-mounted tablets. The interface includes a variety of built-in widgets, and users also have the option to use third-party widgets or create their own.

[5]Term *things* refers to physical devices in OpenHAB

[6]Sitemaps configuration – https://master--openhab-docs-preview.netlify.app/docs/configuration/sitemaps.html

[7]HABPanel configuration – https://master--openhab-docs-preview.netlify.app/docs/configuration/habpanel.html

Once a widget is created, it can be easily moved around the screen using drag-and-drop controls or resized.

OpenHAB's visualization techniques are also comparable to those in Home Assistant, with line charts being the primary method for displaying historical data. An example is shown in Figure 4.7. Hover and click actions are available to interact with data points. However, OpenHAB includes a unique feature: the ability to select periods and time ranges right next to the displayed graph. These features allow users to adjust the displayed time frame, enabling quicker and more efficient analysis of historical data compared to Home Assistant.

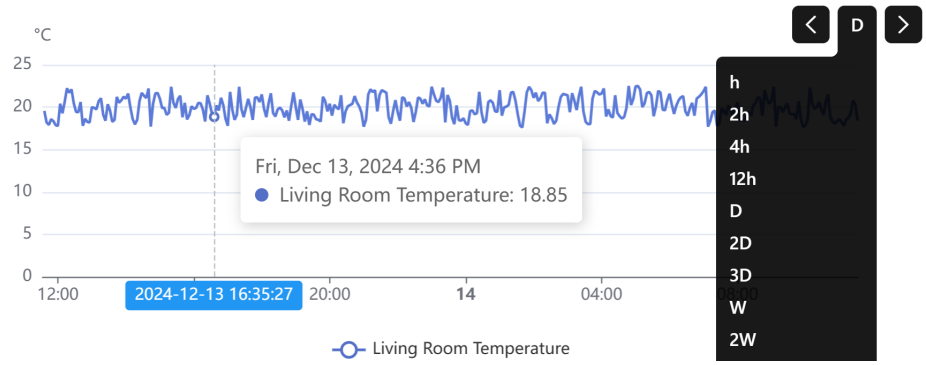


Figure 4.7: Example line chart of temperature data

In addition to line charts, OpenHAB offers a variety of other visualization types, including bar graphs, heatmaps, pie charts, scatter plots, and gauges[8]. Users can also overlay multiple graphs, combining them into a single chart with shared axes.

What should be noted is that OpenHAB does not store historical data for all entities by default. RRD4J[9] service is used for persistence, and it is installed out of the box. Users need to configure which items[10] they want to persist. Once configured, RRD4J allows OpenHAB to record and store the state changes of those selected items, enabling historical data analysis and visualization. Without such a configuration, OpenHAB only retains the current state of the items.

Visualizations of state changes of individual items, as shown in Figure 4.5 for Home Assistant, are not a dedicated feature in OpenHAB. Instead, regular graphs are used, which might not be the best choice and can feel rather cluttered for items with binary states.

Groups of Devices

Groups of items in OpenHAB aim for the same goals as Home Assistant, primarily streamlining visualizations and automations. However, in OpenHAB, groups must be created and configured through textual files; nevertheless, when compared to Home Assistant, OpenHAB provides much greater control over the groups and their behaviors.

In OpenHAB, groups support multiple state aggregation functions, including arithmetic operations like *SUM* or *AVG*, as well as boolean operations such as *AND*, *OR*,

[8] OpenHAB charts documentation – https://www.openhab.org/docs/ui/chart-pages.html
[9] OpenHAB – RRD4J documentation – https://www.openhab.org/addons/persistence/rrd4j/
[10] The term *item* is a virtual representation of a physical *thing* in OpenHAB

or *NOR*, and others[11]. Nested groups are also supported, making it easier to organize devices in a structured way. For example, a "Power Usage" group can aggregate energy consumption data from multiple smart plugs that can be grouped by individual rooms. This principle presents the total consumption as a single value, while subgroups can display data for specific rooms or zones. With proper customizations, the overall state of a group can be plotted into a graph, showing the results of a selected aggregation function over time.

Google Home

Google Home[12] focuses on ease of use and accessibility to cover a much broader customer base. To achieve these goals, the Google Home app is available on all devices imaginable, including tablets, smartphones, and desktops, with its responsive web user interface. Creating simple smart homes is effortless with Google Home, mainly due to its setup process, which requires minimal technical knowledge or experience. Integrations are widely supported among popular services and devices, such as lighting, thermostats, and cameras. Google Assistant further solidifies Google Home by enabling voice command support, thus eliminating the need to always pick up a phone or approach a wall-mounted tablet to control devices. With its ease of use, accessibility, and seamless integration with smart devices, it dominates the market among the other mentioned platforms. The app itself has over 500 million downloads solely in the Google Play[13] store.

Unlike the two previously mentioned platforms, Google Home does not provide any advanced visualization features. Moreover, the historical data is not directly available to the user, and can only be retained through third-party integrations, like Google Nest, or other workarounds, such as using the IFTTT[14] platform to log data into a spreadsheet.

Real-time control and status monitoring are the primary functions of Google Home. These functions are carried out using simple dashboards, shown in Figure 4.8, primarily consisting of card elements displaying device status and allowing quick interactions.

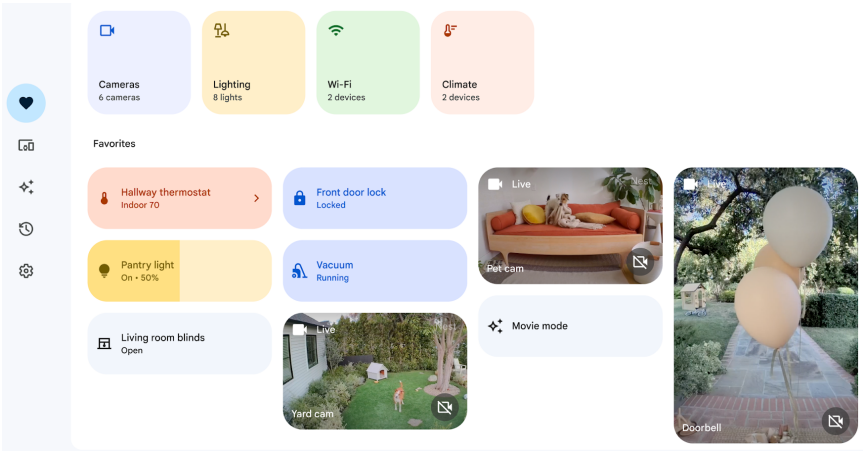


Figure 4.8: Example Google Home dashboard layout on a tablet, taken from gallery[15]

[11] OpenHAB groups and state aggregation functions – https://www.openhab.org/docs/configuration/items.html#groups

[12] Google Home homepage – https://home.google.com/welcome/

[13] Google Play – Google Home app – https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app

[14] If This Then That (IFTTT) homepage – https://ifttt.com/

Groups of Devices

Groups are also rather simplified in Google Home, and devices can be organized into rooms based on their physical location. Users can then use commands, such as "Turn off the lights in the kitchen", and Google Assistant will execute corresponding actions. This grouping principle is great for simple scenarios, but it lacks advanced features such as aggregation functions. Groups in Google Home are primarily used as logical and visual dividers between individual clusters of devices within a smart home.

Logimic

Logimic provides a specialized platform for asset control and data acquisition (ACADA), focusing primarily on scalable and robust solutions. Unlike the previously mentioned platforms that cater mainly to the consumer smart home market, Logimic is mostly focused on industrial applications and large-scale deployments. Their projects target other areas, including street lighting control systems, medical monitoring, and air quality monitoring. To support these large-scale deployments, they use visualizations and techniques that differ greatly.

ACADA dashboard solutions, as mentioned, focus on large numbers of smart devices, making it nearly impossible to display the statuses of individual devices in such a valuable space. Instead, groups are heavily used to simplify the visualization process. For example, street lights can be organized into groups based on the streets or electrical cabinets, reducing clutter and allowing operators to quickly identify exact areas in need of attention.

Furthermore, in order to alert the operators, many types of KPIs, as described in Section 3.4, are set up and leveraged to display critical values while omitting less relevant data. These KPIs and their definitions range based on the specified use case, but often include metrics such as battery levels, signal quality, or temperatures.

The resulting dashboard, as illustrated in Figure 4.9, is specifically designed to handle large volumes of devices, featuring a layout and visualizations that deliver clear and actionable insights, distinguishing it from the previously analyzed solutions.

Figure 4.9: Logimic dashboard

[15]Google Play – Google Home app preview photos – https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app

Upon clicking on any of the groups, further details about the individual KPIs can be displayed in the KPIs tab, providing operators with more granular information. This functionality is demonstrated in Figure 4.10, where a detailed view of a selected group's KPIs is shown.

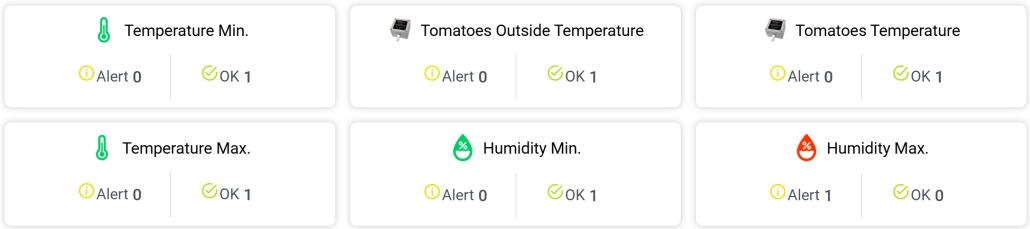


Figure 4.10: Detailed view of KPIs for a selected group in the Logimic dashboard

If needed, individual devices within groups can be previewed, with visualizations offering comprehensive information about active alerts and device profiles. These include parameters, attributes, model details, and additional transmitted data, presented using multiple gauges.

Historical data statistics for each device and its transmitted metrics are presented using customizable line charts. Users can adjust the timeframe, select specific input data, and display multiple datasets simultaneously. This interface is shown in Figure 4.11.

Figure 4.11: Device statistics shown using a line chart with multiple data inputs

4.4 Interaction Analysis

The analyzed solutions offer varying levels of interactivity, with hover and click actions being the most predominant in all of them. However, it is important to note that hover actions are naturally not available on mobile devices. Hover actions are used mainly in graphs to display detailed information about a single data point, such as its precise value or timestamp. Beyond graphs, hover actions are often used to provide tooltips or help messages, providing additional context to users.

On mobile devices, click actions replace hover functionality. Although this approach ensures compatibility with touchscreens, it can make it more challenging to interact precisely with individual points on a graph. Otherwise, click actions are the most used across all solutions, whether it is for navigation, toggling an entity's state, or opening detailed views.

Drag-and-drop actions have also started to emerge, and since March 2024, they have been part of Home Assistant[16]. Similarly, OpenHAB, more specifically its HABPanel dashboard interface[17] also supports them. Their primary use in both mentioned platforms is the rearrangement of widgets or cards, with HABPanel also allowing resizing.

Long-press actions are very limited in the analyzed solutions, with Google Home being the only one taking advantage of this type of interaction. In its mobile application, a standard click action on a light card toggles the light's state. Further information or configuration can be accessed by long-pressing on the card. By doing so, additional details, such as the brightness or hue of a light bulb, can be accessed and changed.

4.5 Derived Requirements

From the analysis of existing solutions, it is apparent that in order to cater to the whole user domain, the setup or customization of the dashboard must be simple and intuitive. Thus, the whole process should take place within the graphical user interface (GUI) and not within the textual files, such as in the configuration process of OpenHAB, as discussed in Section 4.3. The initial setup is a crucial part of the whole configuration and must be kept straightforward and user-friendly, with live previews of the resulting visualization.

Data visualizations should be kept to a reasonable information density as well. For instance, presenting users with overlaying graphs out of the box may be overwhelming. The visualizations should be simple and customizable, such as by changing the time frame or changing the axis names. Hover and click actions are also really powerful in data visualizations and can provide detailed information about a single data point in a graph, for example. That way, users are provided with the data's historical trends, but also precise values when needed. Multiple data sources and their data set overlaying in a single graph can also be helpful, as unknown correlations and trends can emerge many times.

Groups are another important element when dealing with smart devices and dashboards. With many devices onboard, it is oftentimes even unavoidable to reduce redundancy and clutter. In the analysis, their extents and usability vary greatly from simpler ones, like in Home Assistant or Google Home, without any fancier functions, to groups using KPIs in Logimic's dashboards. KPIs are powerful tools and can make groups "smarter", such as by setting a critical or warning threshold for individual devices in a group and then seeing if the whole group is stable or not. These indicators provide users with a monitoring tool that monitors the performance or status of grouped devices more effectively and alerts if any actions are needed.

Responsive design, as discussed in Section 3.5, was also shown to be necessary in the world today. Smart homes are designed to make one's life easier by streamlining daily routines or by providing seamless control over various devices, regardless of the user's location or device being used. All analyzed solutions incorporate responsive web design to some extent, although many provide dedicated applications for tablets or smartphones.

Another key requirement is the ability for users to customize and even completely build the dashboard layout. In smart homes, the diversity of devices and varying priority levels of different users mean that a fixed layout cannot serve everyone's needs. Custom dashboard layouts allow users to prioritize the information and controls that are most relevant to them.

[16] HomeAssistant – Drag&Drop blog post – https://www.home-assistant.io/blog/2024/03/04/dashboard-chapter-1/

[17] OpenHAB – HABPanel documentation – https://www.openhab.org/addons/ui/habpanel/

Chapter 5

Dashboard Proposal

The analysis of existing solutions in Chapter 4 highlighted differences in design choices, greatly influenced by the target audience and their technical proficiency. OpenHAB and Home Assistant cater more towards advanced users, providing a wide range of customizable visualizations that require technical configuration. In contrast, Google Home emphasizes simplicity and accessibility, making it more appealing to a broader audience with limited technical expertise. Building on these observations, the proposed dashboard and its elements aim to balance these contrasting approaches, providing a solution that is both intuitive and informative. The goal is to ensure that the solution is neither overwhelming nor lacking in contextual depth, making it suitable for a diverse range of users.

Before proposing a solution, one of the key factors to consider in design decisions is the target devices. In smart homes, once users have set up their system, including the dashboard, they should not be required to rely on a desktop computer or laptop at all times. Instead, using portable devices such as mobile phones or wall-mounted tablets offers a more convenient way to quickly assess the situation at a glance. The analyzed existing solutions, particularly Home Assistant and OpenHAB, are primarily designed for desktop use. Although they offer a mobile interface, it largely mirrors the desktop version, which can make it more difficult to navigate. This thesis will primarily focus on responsive design and mobile phones as target devices.

However, mobile devices present several challenges when it comes to dashboards, the primary being limited screen space, and others, such as the lack of hover actions, which often provide additional detailed information on desktops. These limitations will require different design solutions to ensure that essential information is still accessible and to make sure that interactions remain intuitive on smaller screens.

5.1 Dashboard Layout

The dashboard layout should be designed with careful consideration of the information hierarchy, as introduced in Section 3.2. The topmost section of the dashboard, which has the greatest amount of natural emphasis, is static and displays groups of aggregated devices with at least one failing KPI and requiring immediate attention. Furthermore, the displayed cards in this static section should be sorted based on their importance as well, such as by the number of failing devices in each card, arranged in descending order from left to right.

Below this static section, the dashboard continues to be user-customizable. Users can freely configure the visualizations in this area according to their preference and judgment of importance. The hierarchy of the dashboard layout, along with the natural emphasis direction, is shown in Figure 5.1, and its principle applies not only to desktops but also to smaller devices.

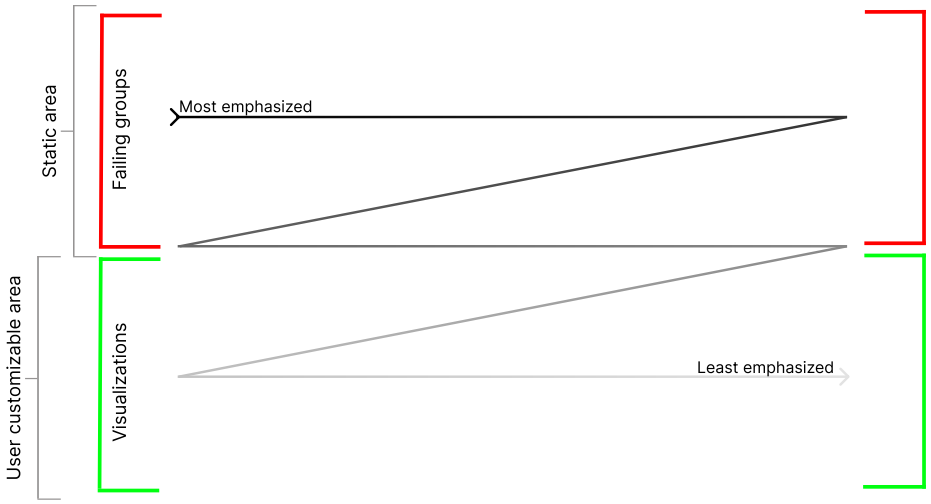


Figure 5.1: Dashboard layout areas with natural emphasis direction

Smaller devices can only display a limited amount of information on a single screen without scrolling, and when dealing with numerous devices, it can be insufficient to evaluate the state of the entire smart home. On smaller devices, vertical scrolling is crucial to make the most of the available space and ensure that users can quickly access all the information they need. Since vertical scrolling is deeply integrated with users' daily interactions with mobile devices, whether it is browsing the internet or social media, it should be the predominant navigation method. Additionally, horizontal scrolling can serve a complementary function by, for instance, enabling users to switch between different dashboard views, such as navigating between rooms or categories.

5.2 Visualization Elements

The usage of KPIs can streamline the dashboard and enhance overall information density by highlighting only critical values, such as low battery level. Devices that are not as essential or used solely for automation often do not require full-fledged visualizations in the primary dashboard view. Instead, being alerted about their critical statuses is enough in many cases and saves a lot of valuable dashboard space, especially on mobile devices.

When this approach is combined with device aggregation in the form of groups, users can still be easily alerted, and the design remains compact in the main dashboard view. These visualizations, as illustrated in Figure 5.2, are meant to be displayed in the static section of the dashboard. The left variant is explicitly optimized for small-screen devices, minimizing vertical space usage. Data on each card follows the natural reading flow from left to right or top to bottom. Muted background colors provide additional context by visually indicating the proportion of failing devices within a group. For example, approximately half of the background is colored according to the triggered thresholds in "Temps" group, where three out of six devices exceeded their thresholds.

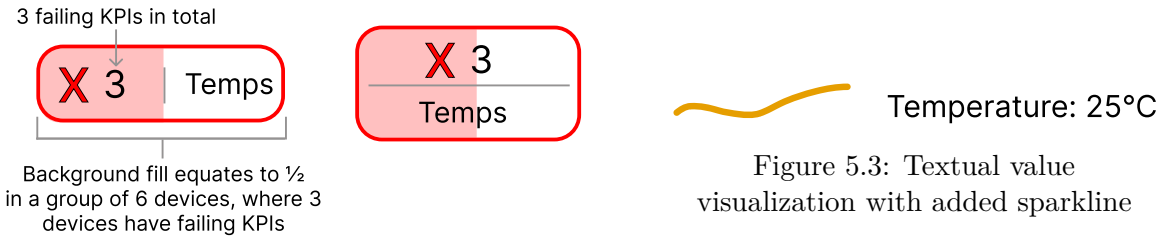


Figure 5.2: Groups visualization with KPIs

Figure 5.3: Textual value visualization with added sparkline

As discussed in Section 3.3, textual values can often be superior to other visualizations, due to their ability to provide precise information that users can easily interpret. On the other hand, sometimes it is vital to be aware of the historical context and current value. Combining a sparkline with textual value achieves just that, while taking up minimal space, resulting in a simple, yet informative visualization as shown in Figure 5.3.

Tables can be the ideal solution to display precise values from multiple entities. The proposed formatting of a table is shown in Figure 5.4 and minimizes the presence of non-data pixels, previously discussed in Section 3.2. Incorporating user-defined thresholds or KPIs can further enhance the functionality of tables by drawing attention to critical values. Values that either exceed or fall below a predefined threshold, based on desired behavior, can be visually highlighted to alert the user.

Sensor	Min.	Max.	Avg.
Kitchen	19.3°C	**23.4°C**	21.1°C
Hall	**18.1°C**	21.9°C	20.3°C
Bathroom	21.4°C	**23.7°C**	22.6°C

Figure 5.4: Table with highlighted values based on set KPI thresholds

For historical data, line charts are often the superior choice, as they provide more detailed insights than sparklines. As noted in Chapter 4, they are widely used in existing solutions and particularly popular for showing trends over time and often support additional interactions, such as viewing a particular data point.

In this proposal, important values of the x-axis are highlighted based on the selected time frame to provide users with clear visual cues. For example, if the time frame spans a month, the start of each new week might be highlighted, whereas, for a weekly view, each new day could be emphasized. As discussed in Section 3.2, excessive grid lines can be distracting and create non-data pixels, and therefore, users should have the option to toggle grid lines on both axes according to their preferences.

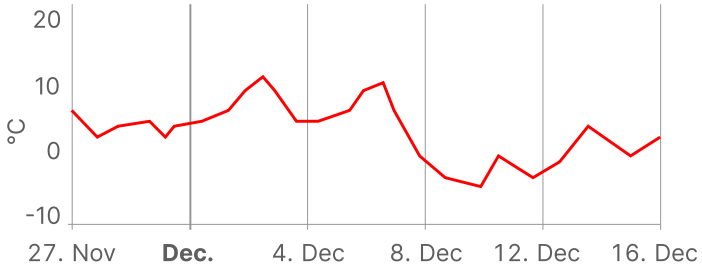


Figure 5.5: Line chart depicting temperature, with only x-axis grid lines

Simpler visualizations are frequently more appropriate when working with non-quantitative data or data that might not be visually appealing when graphed, such as binary states. Displaying how states sequentially change over time provides a clear and intuitive overview of state transitions. This method works particularly well for tracking changes in devices where binary or categorical states dominate, such as device statuses (e.g., *on/off*) or process stages (e.g., *heating, starting, printing, cooling* in 3D printing). By arranging these states chronologically, users can spot trends, irregularities, or recurring patterns with ease. In addition, interactive features, such as hovering or dragging across the visualization, can provide further detailed information about a particular segment, including properties such as its duration or other important associated metadata.

This visualization also has certain limitations, mainly due to the use of colors, as discussed in Section 3.2. The visualization, shown in Figure 5.6a, is efficient when dealing with binary or a few categorical states. However, if a device does not fall into these two categories, this visualization, as shown in the mentioned Figure 5.6a, may not be suitable. Differentiating between many distinct colors while still being able to recognize their corresponding states is challenging, and for color-blind people, it may even be impossible.

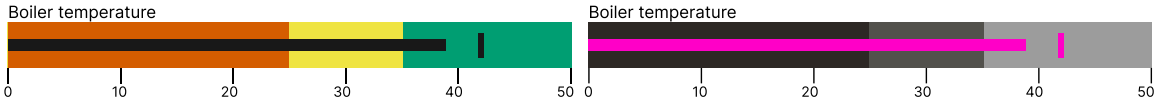
(a) Colored variant (b) Black-and-white variant

Figure 5.6: Two styles of proposed sequential state visualization.

To address this issue, distinguishing only adjacent states with distinct colors and replacing full state names with simple numerical values or even completely omitting the value can provide a more effective solution, as illustrated in Figure 5.6b. Users can access additional details about the individual segments through the mentioned interactions, which reveal detailed information when needed. This approach reduces visual clutter and helps when working with large time frames or devices that frequently change states.

Gauges are well-known and widely used to date. However, gauges often occupy significant space while delivering limited information. Bullet graphs, as described in Section 3.3 and illustrated in two variants in Figure 5.7 below, offer an interesting replacement, with their compact footprint and efficient display of performance metrics. They display a clear visual representation of the latest measurement compared to predefined qualitative ranges, represented by background colors from worst to best in left-to-right orientation, as per the referenced Figure 5.7.

Although S. Few [13] uses only gray hues, the colors in Figure 5.7a were carefully chosen to be accessible to most individuals with color vision deficiencies, in accordance with the guidelines in [42]. Nevertheless, users should not be limited to the colored variant and should have the option to choose between color and gray hues, as shown in Figure 5.7b according to their preference.



(a) Colored variant (b) Gray hues variant

Figure 5.7: Both color variants of the proposed bullet graph.

Sliders, clickable cards, and simple switches are ideal for interacting with actuators. These visualizations can also effectively display the current state. In Figure 5.8, actuators or groups of actuators can be displayed alongside other entities on a single card, with their current states highlighted through the colors of the icons and the positions of the toggle switch. Figure 5.9, on the other hand, focuses on a single entity, using background color to indicate its current state. For example, on the bottom card, the background fill is equivalent to 50% of the brightness. This behavior could be mapped to different properties of the entity, such as motor speed or other metrics with specified minimum and maximum values.

Figure 5.8: Card displaying groups of lights with switches for individual groups

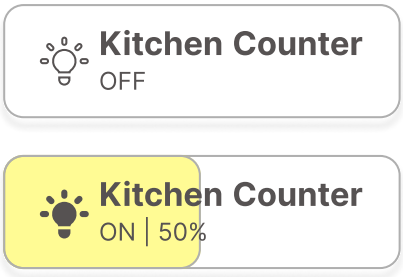


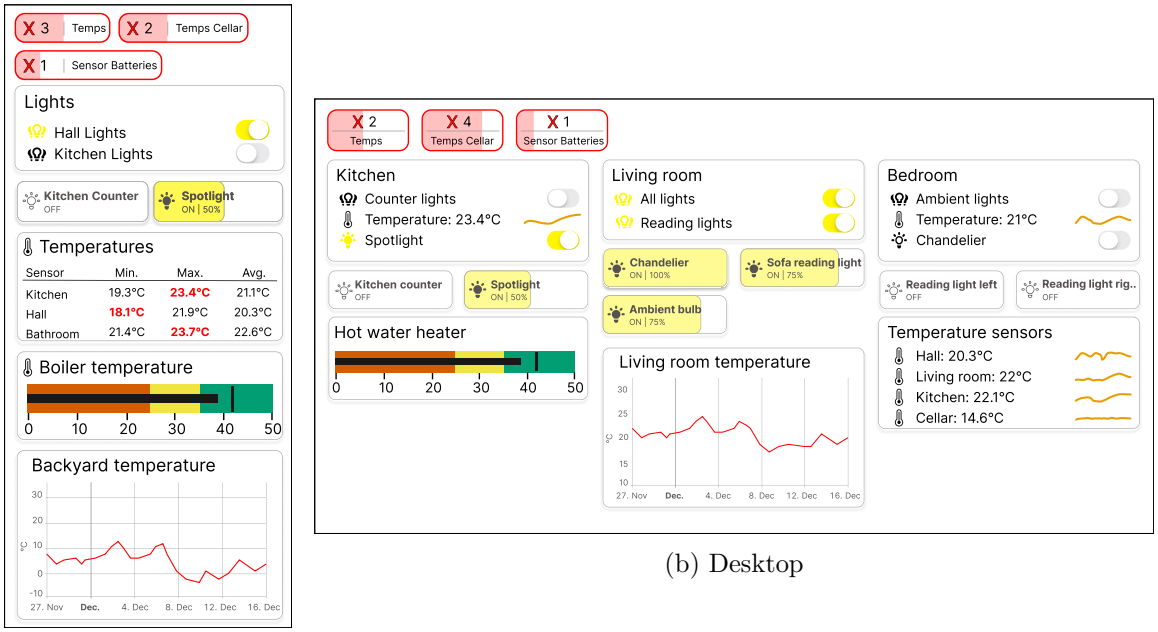
Figure 5.9: Cards displaying a single light entity

5.3 Proposed Views

Building on the visualization elements presented previously, this section presents the proposed views for detailed screens and the dashboard. These views are meant to provide users with a comprehensive and intuitive interface for monitoring and controlling their smart devices. Starting with the general dashboard views for both mobile devices, shown in Figure 5.10a, and for larger screen devices, such as desktop computers, illustrated in Figure 5.10b.

The dashboard is meant to serve as an overview of a smart home that can be viewed at a glance. The proposed dashboard follows the principles described in Section 5.1, and emphasizes KPIs as the main focus. Thus, the groups that leverage KPIs are prioritized and displayed at the very top of the dashboard, where the natural emphasis is the largest. This approach ensures that critical issues are immediately visible, in accordance with the layout principles discussed in the mentioned Section 5.1.

In addition to dashboards, detailed modal views are used to provide in-depth information about individual devices or groups. These modals display only key device properties and historical data using an appropriate visualization based on the data. For categorical data, the sequential states visualization is shown, as in the case of Figure 5.11. On the other hand, a line chart is used instead for quantitative data. The modal view serves only as a quick overview and displays a button to redirect the user to a dedicated detail page if more information is needed. This approach addresses navigation issues by allowing users to quickly access detailed information without always being redirected.



(a) Mobile

(b) Desktop

Figure 5.10: Examples of the proposed dashboard layout on different screen sizes.

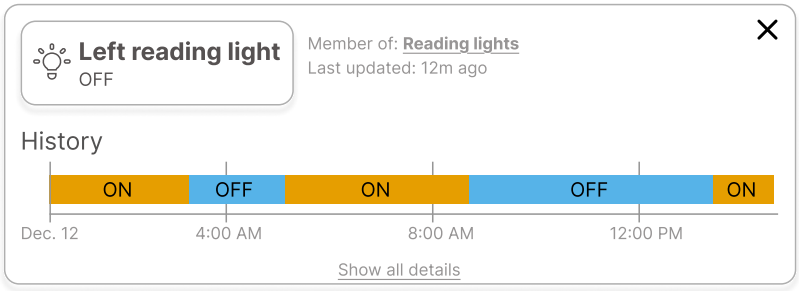


Figure 5.11: Detailed modal view of a device with categorical states

When a group of devices contains devices with failing KPIs, it is crucial to identify the specific devices causing the issues. The detailed group modal view, shown in Figure 5.12, presents a brief overview of the total number of devices in the group, the number of failed devices, and their percentage relative to the group as a whole. Below this concise overview, a detailed list of the failing devices is presented, with each device's failing KPIs clearly indicated. Other devices that are not failing are not shown by default, as this modal view is only meant for quick overviews. To access complete group details, users can click the redirection button at the bottom of the modal view.

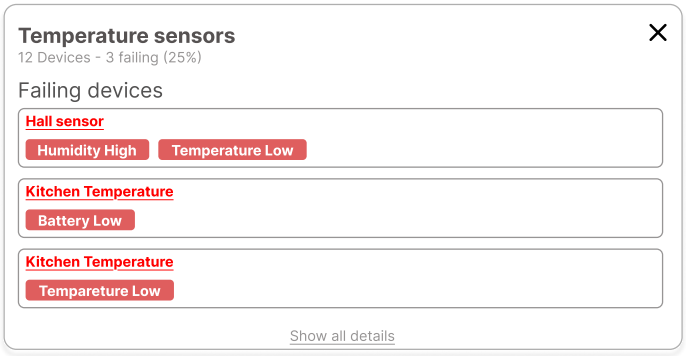
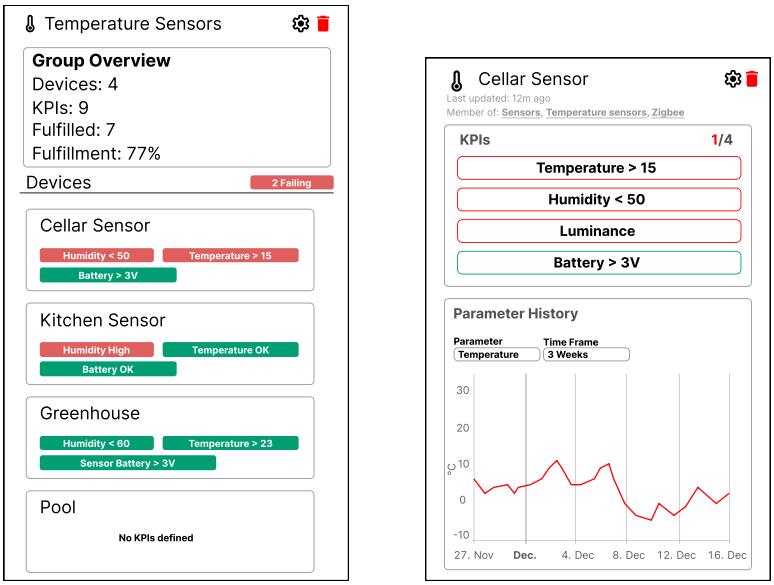


Figure 5.12: Detailed modal view of a group

Modal views provide brief summaries, but they are not always enough for in-depth analysis. In these situations, specific detail pages may be required to dig deeper or gather in-depth information about a particular group or device. Unlike modal views, these pages present as much relevant information as possible and include additional controls, such as settings and delete options.

The dedicated group detail page, as shown in Figure 5.13a, displays all devices within the group along with their KPIs, including both fulfilled and failing. The individual devices listed in this view are also meant to be clickable and display further information about a particular device.

Similarly, the device detail page, shown in Figure 5.13b, presents a complete overview of the device, prioritizing KPIs immediately after a brief summary section that includes the time of last update and its associated groups. This is followed by an interactive section with parameter and time frame selectors, allowing users to preview historical data.



(a) Dedicated group detail view

(b) Dedicated device detail view

Figure 5.13: Mobile views of dedicated detail pages.

5.4 Interactions

Interactions greatly affect the user experience and the overall usability of the dashboard. As discussed in Section 4.4, the analyzed solutions often lack in terms of interaction design, especially on touch devices, apart from Google Home, which stands out.

The proposed dashboard should implement intuitive interactions for both touch- and mouse-based devices. These actions are carefully chosen based on common patterns that most users are familiar with to ensure their intuitiveness.

- **Clicking:** On both desktop and touch-based devices, clicking or tapping on elements will open detailed views or additional information.
- **Hover, Long-press and Drag:** On desktop computers, hovering over elements on the dashboard, such as line or bullet charts, will display tooltips displaying detailed information about a single measurement. On touch devices, where hovering is not available, a long-press gesture will serve the same purpose. Furthermore, dragging inside a line chart could display an interactive tooltip, so that users can locate the precise data point more easily compared to solely tapping.
- **Drag and Drop:** Users should be able to rearrange dashboard components using drag-and-drop on desktop devices, as this technique is quick and intuitive with a mouse, and has also started to emerge in other existing solutions. However, because using drag-and-drop can be rather challenging on mobile devices, alternative methods such as dedicated move buttons should be provided for touch interfaces.
- **Horizontal Swipes:** On touch devices, horizontal swipe gestures can be used to switch between different dashboard views, such as changing rooms or device groups.

Chapter 6

Implementation

The implementation into the already existing Real-time IoT (RIoT)[1] architecture, originally developed as part of Michal Bureš's thesis [9], roots in the requirements analysis, already presented in Chapter 4. The primary objective was to design and develop a responsive, user-customizable dashboard component that uses visualizations from a graphical library, but also to provide detailed views for both devices and groups for drilling down, while ensuring a consistent user experience and the following principles discussed in Chapter 3. The implementation was carried out in collaboration with Marek Joukl [23], who was responsible for other system modules, such as navigation, integration of a visual programming language editor, or user management. My work concentrated on the responsive dashboard, its visualization builders and editors, detailed views for groups and devices, in both modal and dedicated page forms, and the groups overview page.

6.1 Technologies

The selection of technologies was guided by the requirements for a modern, maintainable, and performant web application. The frontend application is built using React, which is a declarative, component-based JavaScript library. Its component-based approach, along with its ability to re-render only parts of the Document Object Model, made it a good choice for this system with loads of interactive and dynamic user interfaces (UIs) or for IoT monitoring itself, as its data is not static. *Vite*[2] was chosen as the local development server due to its features, such as instant hot module replacements or optimized production builds. *Vite* also supports TypeScript, which was also utilized in this project, and all source files have been written using it. React also supports its TSX type or typed JavaScript XML, which allows to combine JavaScript and HTML into a single file, or TypeScript in this case.

Styling is managed primarily using Tailwind CSS, which is a framework that provides a list of utility CSS classes that can be applied directly to HTML elements via classes. For more complex UI elements, custom React components were developed and, where needed, created and styled using *Styled Components*[3].

For advanced UI components, the project utilizes *shadcn/ui*[4], which is a set of designed, accessible components, based on Radix UI[5] primitives, and a code distribution platform.

[1] Real-time IoT (RIoT) homepage – https://github.com/pocketix/riot
[2] Vite homepage – https://vite.dev/
[3] Styled Components library homepage – https://styled-components.com/
[4] shadcn/ui homepage – https://ui.shadcn.com/docs
[5] Radix UI homepage – https://www.radix-ui.com/primitives

This allowed for the incremental adoption of components that are highly accessible by default, but also completely customizable. Using *Tailwind* alongside *shadcn/ui* also helps with "bundle"[6] size as only the necessary utility CSS classes and components are included in the final build.

For some data visualizations, including line and bullet charts, the *Nivo*[7] charting library was used. *Nivo* is built on top of *D3*[8] and React, and provides interactive and highly customizable chart components rendered in SVG. This ensures that charts remain sharp and clear at any resolution or zoom level, unlike raster-based graphics, which can become pixelated. Furthermore, *Nivo* allows dynamic updates and real-time data rendering.

Data communication with the backend is handled by the Apollo GraphQL[9] client, which provides a solution for GraphQL queries, mutations, and subscriptions. This client also includes advanced features such as caching or real-time data synchronization.

For runtime validation, further described in Section 6.3, the solution makes use of the *zod*[10] library. *Zod* is a TypeScript-first schema declaration and validation library that enables the creation of strict schemas, which are suitable even for complex data structures, with nesting support. In this project, *zod* plays a crucial role and is used to validate the dashboard configuration and the individual visualization card's configurations, as well as the builder's form values before submission.

6.2 Architecture

The created components and pages are based on the Model-View-Controller (MVC) architectural pattern, the main goal of which is to separate concerns into multiple components, thus also enhancing the general maintainability. In the implementation of my part of the system, each of the main components is divided into two main parts: the *controller* and the *view*.

The *controller* is responsible for handling data logic and communicating with external services, such as the GraphQL API in this case. It manages the execution of generated GraphQL queries, mutations, and subscriptions, which can be referred to as the *model*, based on the defined GraphQL schema. The *controller* then processes the results and informs the view of the result. It acts as a "middleman" and ensures that the view receives the data in a format specified by the view's arbitrary inputs, often called "props". These formatting operations are often not the simplest, and the view would become too complex without this separation.

The *view* is focused on rendering the user interface based on its "props" and presenting data to the user, with little to no data manipulations. Additionally, the view can also initiate requests to the controller, such as fetching new data or triggering a database object's update or creation. For instance, in the visualization builders on the dashboard, upon changing relevant fields in the builder's form, new data is fetched in order for the preview to correspond with the final visualization. The MVC architecture used in the system is depicted in Figure 6.1 below. For simplicity, the backend is shown as a single element, although in reality it consists of a GraphQL API that handles requests and returns responses

[6] A *bundle* refers to the collection of JavaScript, CSS, and other assets sent to the browser
[7] Nivo charts homepage – https://nivo.rocks/
[8] D3 data visualization library homepage – https://d3js.org/
[9] Apollo GraphQL homepage – https://www.apollographql.com/
[10] Zod library homepage – https://zod.dev/

to the *model* layer, as well as two databases: PostgreSQL for relational data and InfluxDB for time-series data.

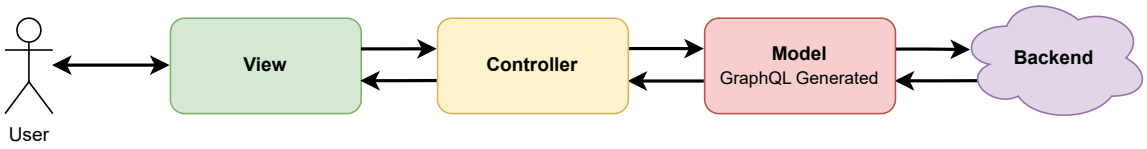


Figure 6.1: Used MVC schema

Context Management

The resulting application is not static, and the system allows for adding, deleting, or updating the device's properties, such as its name. These changes need to be reflected and become challenging with many components. In order to prevent the so-called *prop drilling* and to enable efficient global state management, the application uses React Contexts. This feature allows data and functions to be shared across the component tree without the need to explicitly pass props through every level or repetitive data fetches before presenting the views.

Context Providers are special React components that supply the context value to all their child components. In order for a child tree to have access to the context values, it must be encapsulated with a provider. By doing so, the children gain access to the context's values and functions. In this application, context providers are used to manage global states and also provide convenient utility functions related to devices, groups, and real-time data updates.

Instances Context

The first of the implemented contexts is the instances' context. Instances are used throughout the application, from builders, where the user selects an instance and its parameters to be visualized, to the visualizations themselves. The entire instance object is not being stored in the database after a visualization configuration is submitted, only its unique identifier. As a result, each time the instance is visualized, the complete object must be retrieved to ensure that the latest properties are loaded. The same goes for parameters; for instance, when building a visualization, it is always necessary to display the selected instance's parameters. Instead of alerting the controller and fetching new data using the model, the instance's context utility function is invoked directly to retrieve its parameters.

Subscription Contexts

GraphQL and the backend of the system offer subscriptions, a substitute for webhooks, and provide real-time updates, which are, in this case, KPI updates and parameter snapshot updates. Rather than establishing multiple subscriptions within the application, this context uses a single subscription, which is then delegated to the subscribed components. Moreover, the currently implemented backend subscriptions do not allow subscribing to a specific KPI or instance.

In order to compensate for the lack of precision of the subscriptions, their updates are only propagated to the interested components using the *useSyncExternalStore* hook, which allows for setting listeners, who can then be notified about changes. Without using this

hook, the components would re-render each time new data came, even if they would not be affected by the data change. With many sensors, each sending multiple data points per minute, this would be a serious drawback and performance issue.

Having implemented the smart subscription contexts, a new hook was made that benefits from both the instances' context and the KPI subscription context. A hook that provides instances with KPIs and their statistics, such as the number of fulfilled KPIs or the total number of KPIs, while also providing real-time updates on new subscription data. This hook was used within multiple components, including the detailed view for both a singular device and the group detailed views, which update without the need for refreshing.

6.3 Dashboard

The dashboard component, essentially the central feature of the application, is implemented as a dynamic, responsive, grid-based layout using the *react-grid-layout*[11] library. Allowing users to rearrange, resize, and configure individual visualization cards within the dashboard in a respective section based on their preferences.

Information Layout

As proposed in Section 5.1, and illustrated in Figure 5.1, the implemented dashboard adheres to the same hierarchical information layout. While users are provided with extensive customization options and can create a dashboard that suits them, the topmost section remains static. It is reserved for critical system information based on the user-defined KPIs and device groups.

The top area of the dashboard consistently displays group cards that summarize the state of individual groups and highlight the number of failed KPIs in contrast to all KPIs for the given group. However, this section shows only aggregate visualizations for groups with at least one failing KPI. This ensures that the dashboard only displays relevant information and is immediately visible to the user. The positioning of these cards is intentional, as the area is viewed as one of the most natural emphases by users and should draw the user's attention to issues that might require immediate action.

Below the static topmost section, the remainder of the dashboard is fully customizable. Users can add, remove, and position the added visualization cards according to their personal preference and judgment of importance. The dashboard's ability to be customized almost freely is meant to accommodate a diverse user base, as different users may have varying priorities and needs.

Editing the Dashboard

To support dashboard customization, a toggleable "edit mode" was implemented, accessible from the top-right corner of the dashboard page. By default, the *react-grid-layout* library renders cards as draggable at any given time. However, this behavior is not ideal, as dashboards are typically configured once and rarely modified, and constantly enabled dragging interfered with scrolling on mobile devices. To address this behavior, users must first enable the edit mode to make changes. In edit mode, the cards are overlaid, and tooltips are disabled to prevent any interference.

[11] react-grid-layout library homepage – https://github.com/react-grid-layout/react-grid-layout

Upon entering edit mode, the current dashboard configuration is saved and can be restored at any time to discard changes. In this mode, users can reposition cards by using drag-and-drop gestures via the "four crossed arrows icon" or resize them by dragging the bottom-right "handle". However, as dragging gestures are often impractical on mobile devices, additional dedicated controls appear to improve usability, allowing users to move or resize cards using these buttons, as shown in Figure 6.2.

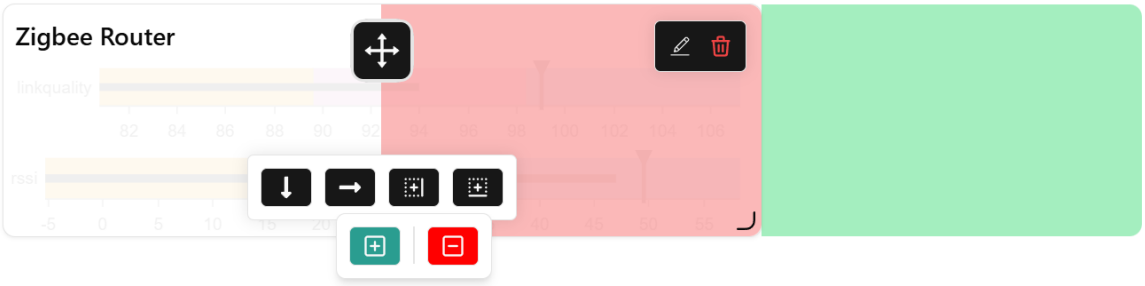


Figure 6.2: Position and size controls of an individual card

These controls only display valid directions based on the card's current layout position and constraints. For example, a visualization can have a minimal height constraint; if so, decreasing its height might not be possible, and the corresponding buttons are hidden to avoid confusion. As illustrated in Figure 6.2, when either of the size adjustment buttons is selected, additional "+" and "-" buttons appear, and the affected card sections are highlighted to preview the changes. These dedicated controls are also accessible via keyboard navigation and switch focus upon opening.

Responsiveness

The grid system enables the setting of multiple breakpoints, effectively changing the number of columns based on the width available for the component. The current definition of the number of columns based on the current breakpoint is a constant in the format "`[key: breakpoint]: columns`" that is defined as follows:

```
const COLS_CONST = { lg: 6, md: 4, xs: 3, xxs: 2 }
```

Breakpoints themselves correspond to CSS media queries, described in Section 3.5, and ensure that the dashboard remains responsive and usable across a wide range of devices and screen sizes. This breakpoint configuration approach, along with its data model, further discussed in Section 6.3 below, makes it possible to define a customized dashboard for each screen or device size. Meaning that the visualizations' sizing properties can be adjusted or the visualization can even be completely omitted for a specific screen size, allowing even more customizability for the users. Row height is also customizable, and the card's default sizing values are calculated and applied to the new item when any builder is finalized. It is important to note that while the grid ensures the layout's responsiveness, the responsiveness of the individual card contents must be handled separately.

Data Model

Dashboard configurations are stored as structured JSON objects in the backend database, inserted through GraphQL mutations, and retrieved via GraphQL queries upon dashboard

initialization. To support the already mentioned responsive design, a separate layout configuration is maintained for each breakpoint. Furthermore, the dashboard is split into tabs, which help with organization, each with separate breakpoint-specific layouts, making them completely independent.

The data model is designed to minimize redundancy by decoupling the layout information from the individual card's visualization details, as it can appear in multiple breakpoints. More specifically, the layout section stores only the positional, sizing information, and other metadata, if needed, such as minimum width or height constraints, about individual cards. The details section contains the configuration and properties of the card itself for each added card. The details are then interconnected with a layout entry at a breakpoint by a unique identifier. Thus, a typical dashboard configuration within a dashboard category then consists of:

- **Layouts:** An object mapping each breakpoint to an array of layout items, where each item specifies the card's unique identifier, position, and dimensions, and other metadata within the grid.
- **Details:** A collection of objects, each representing the configuration and properties of a specific card, such as its title, and a thorough visualization configuration that includes information such as data sources, aggregate functions, and any customization settings.

Zod Validation

Since the dashboard configuration is stored as a JSON object in the database, it does not benefit from any safety or integrity constraints typically provided by the relational database schemas. This creates a potential threat of malformed, incomplete, or generally inconsistent data entering the system.

To compensate for the lack of these features, both the overall dashboard and each individual card's configuration are validated at runtime using the *Zod* library. *Zod* schemas precisely describe the general expected structure, types, and constraints for both the dashboard layout and the card configurations.

The dashboard configuration is obtained immediately upon mounting the component and validated using *Zod's* `safeParse`[12] functionality, which safely parses the data and captures any validation errors. The validation process is modular and takes place across multiple components. First of all, the dashboard's tabs and their respective layouts are validated while the details of individual cards are initially parsed as generic types. The specific configuration of each card is then validated by its respective controller when the card is rendered.

This separation of concerns in the validation process allows the dashboard to remain partially functional even if some cards contain invalid configurations, instead of hiding or resetting the entire dashboard due to a single malformed card. The affected cards are marked and displayed with an error state. The complete validation workflow is illustrated in the flow chart in Figure 6.3 below.

[12] *Zod* – `safeParse` documentation – https://zod.dev/?id=safeparse

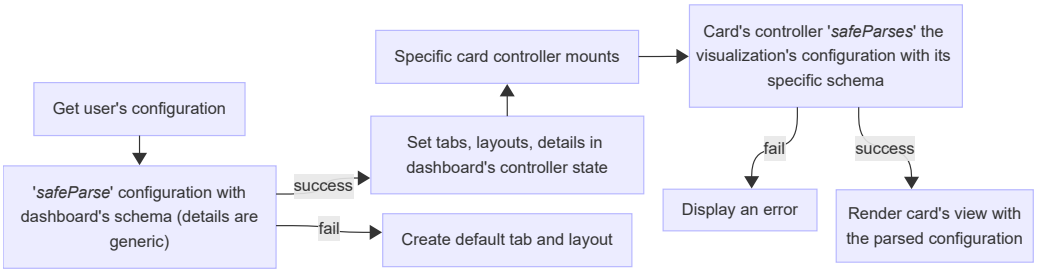


Figure 6.3: Flow diagram describing dashboard's configuration loading

By integrating *zod* schema validations into both loading and saving JSON data, the application maintains a high level of data integrity. This approach is especially important within the dashboard component as it allows for loads of customization.

Efficient Data Fetching

While the entire dashboard configuration is loaded at once, the actual data required for visualizations within each card is retrieved on demand. To optimize performance and reduce unnecessary network requests, the application utilizes *InView* components from the `react-intersection-observer`[13] library. These components detect when a card becomes visible within the viewport, or an offset from the viewport, and only then trigger the card's rendering, thus the data fetching logic for that specific card.

Visualization Builders and Editors

The dashboard includes a set of builder and editor tools that allow users to customize the dashboard and add new visualization cards or edit existing ones. The builders and editor are responsive modal windows that appear as a dialog on large screens or a *drawer* on devices with smaller screens, providing a more native feeling and enhancing the user experience. These builders are available anywhere on the dashboard using the "+" icon in the bottom right corner.

When a user initiates the builder, they are presented with an intuitive selection of available visualization types, each displayed as a clickable preview, as shown in Figure 6.4 below. This gallery of visualizations is sourced from the custom graphical visualization library, presented in Section 6.4, but uses static mocked data in this preview. As the dashboard allows independent layouts for different breakpoints, existing visualizations are also listed below the gallery preview, allowing users to easily add them to the current screen size.

After selecting a visualization type, the user is guided through a configuration process where key properties, such as data sources, based on the selected instance and parameter, aggregation functions, and even appearance properties, such as color schemes, are configured based on the selected visualization. Throughout these builders, tooltips are provided to assist users and clarify the meaning of various inputs.

These builder are built using forms and controlled by their respective *zod* schemas that validate the form's values before submission. If any errors occur, the user is notified and the submission is prevented. Upon adjusting the fields with errors and success-

[13] react-intersection-observer library homepage – https://github.com/thebuilder/react-intersection-observer

fully submitting the builder's form, the new card with configured visualization is added to the dashboard layout at the very bottom.

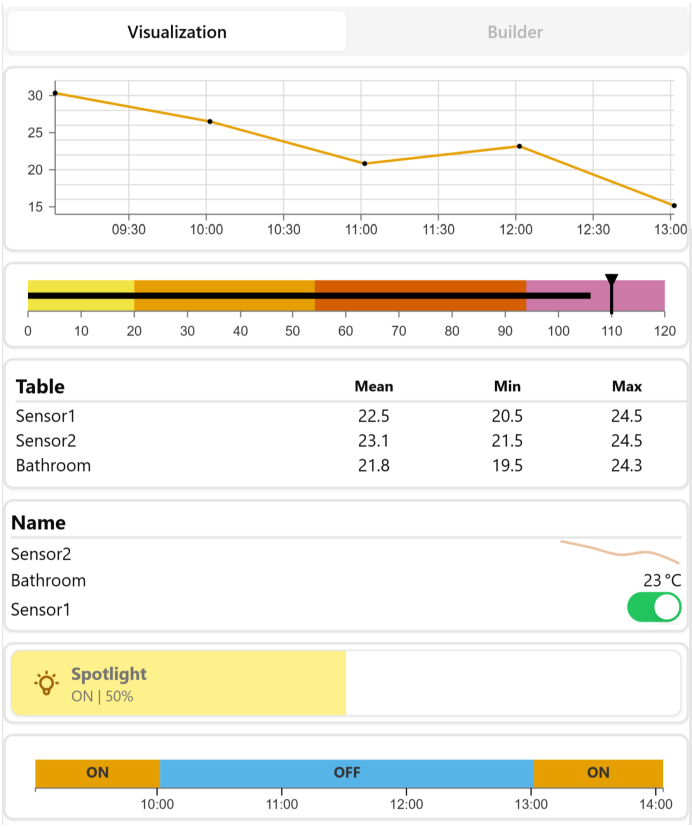


Figure 6.4: Visualization selection in the visualization builder

The builders also handle the cards' default sizing and ensure that the card's content fits when added to the dashboard. The users are then free to adjust their dimensions or position on the dashboard by entering the "edit mode" and either dragging the card or using the dedicated movement and sizing controls.

Builders are also repurposed as editors and accept an initial configuration as an optional "prop". After saving changes in the respective editor dedicated to the visualization being edited, the sizing is recalculated while keeping its original position. The sizing recalculation when editing a card respects the user's existing configuration and only changes it if necessary.

Automatic Configuration and Smart Defaults

The builders incorporate several automatic actions and adjustments to further streamline the process. Notably, in the line chart builder, margins are calculated automatically by generating a mock axis to determine the necessary width and thus adjust margins, avoiding overlapping of the axis legend and tick values. Likewise, in the bullet chart builder, a mock label is used to adjust the left margin based on the width of the "name" field, and the builder can also suggest automatic ranges and targets. If the user provides sufficient information for a single chart, the builder prompts them with a dialog to automatically generate these

values. Upon confirming this dialog, the builder's controller fetches statistics and uses them to calculate the ranges and an approximate target.

A similar approach is applied in the switch visualization builder, where the lower and upper percentage "bounds" can be automatically determined. All the calculated values can also be easily further adjusted by users in all builders.

6.4 Graphical Visualization Library

The graphical visualization library is the foundation for all data visualizations within the dashboard. Given the requirement for a fully responsive dashboard, the library ensures that every visualization is responsive based on available width and height. All charts are rendered using SVG, and plain HTML and CSS are used for other components, such as tables.

A key consideration in the design of the graphics library is accessibility. As discussed in Section 3.2, color-vision deficiency affects a significant portion of the population. Thus, the color palette has been carefully chosen to be color-blind friendly, following the color-blind friendly color palette shown in Figure 3.4 in Section 3.2. This ensures that critical information, such as lines inside a line chart, remains distinguishable for most users. In addition, all visualizations support dark mode while ensuring the dashboard remains visually consistent and comfortable.

Line and Bullet Chart

For advanced charts, such as line and bullet charts, the graphical library leverages the *Nivo* charting library that provides a robust foundation and extensive customization. Starting with the line chart, *Nivo* supports thorough customization, including custom layers. This feature was utilized to implement a specialized x-axis tick layer that highlights transitions between days, weeks, or months, following the approach proposed in Section 5.2.

Nivo also offers interactive tooltips with crosshair support for touch- and mouse-based devices that enable precise data point selection. However, these interactive tooltips have some caveats. On touch-based devices, using the crosshair causes the screen to move when adjusting the crosshair's vertical position, rendering it almost useless. To address this, a custom long-press hook, further described in Section 6.6, was implemented, allowing users to lock the screen by long pressing inside the chart.

The legends for line charts are also custom-built, as *Nivo*'s legends are rendered as SVGs and hence do not support dynamic item widths by default. The custom solution sorts legends by instance and parameter, displaying the full name only for the first occurrence of an instance and initials for subsequent parameters, to optimize space usage. The visualization of the line chart inside a card that appears on the dashboard is shown in Figure 6.5.

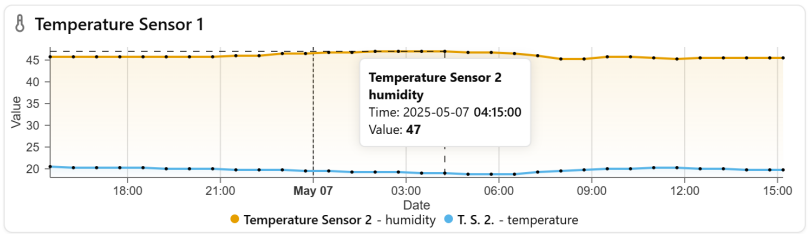


Figure 6.5: Line chart visualization inside a dashboard card

The *Nivo's* bullet chart implementation, while functional, is less customizable than the line chart. Notably, it does not support custom layers or x-axis customization, which can lead to issues such as tick values overlapping in specific environments. The Figure 6.6 below displays an example bullet chart card.

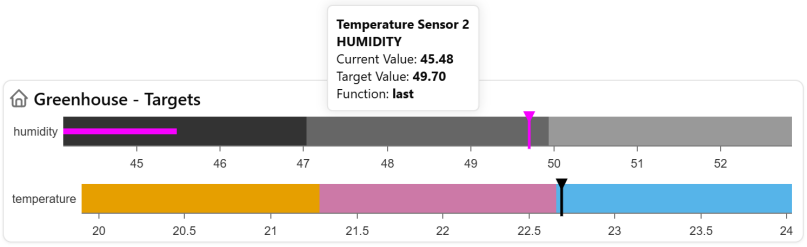


Figure 6.6: Bullet chart visualization inside a dashboard card

Both line and bullet charts utilize long-press detection on touch-based devices to distinguish between short taps and long presses. Long-pressing is dedicated to "getting more information", thus displaying the tooltip. On the other hand, short taps open up a modal device detail view, which will be presented in Section 6.5.

Sequential States

The sequential states visualization is directly implemented with *D3* since *Nivo* does not provide this visualization type. The visualization follows the same pattern as the line chart, with the x-axis highlighting important dates, and is fully responsive, redrawing itself based on the available width. Furthermore, touch actions and draggable tooltip, whose position is shown using the dashed vertical line, are also available for both touch- and mouse-based devices. To improve visibility, states with short durations are visually extended. Furthermore, this visualization automatically switches color pallets based on the number of unique states, as described in Section 5.2. An example is shown in Figure 6.7.

Figure 6.7: Sequential states visualization

Table and Entity Table

Tabular visualizations, including both regular and entity tables, are mainly built using HTML and CSS. The entity table also benefits from the line chart visualization and uses it in a *sparkline* mode, which disables the axis, grid, and tooltips, creating a compact chart that is used inline. Example visualizations are shown in Figures 6.8 and 6.9.

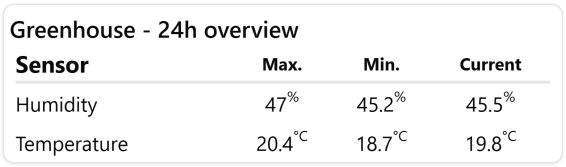


Figure 6.8: Table card visualization

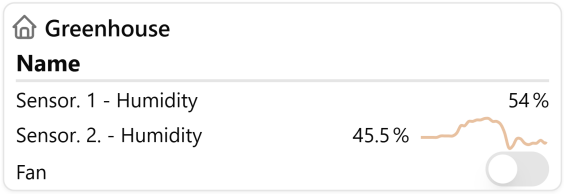


Figure 6.9: Entity table visualization

Switch

The switch visualization closely follows the proposed concept and is implemented solely with HTML and CSS, with its appearance determined by the provided configuration. If a secondary "percentual" configuration is present, the background fill visually represents this value, such as fan speed or light bulb's brightness. This visualization also benefits from long-press actions and displays a slider where the secondary parameter's value can be adjusted. These adjustments and boolean state toggles by clicking the card are purely visual, since their backend integration has not yet been implemented in RIoT. However, once implemented, the component is fully prepared to trigger the corresponding actions. These visualizations and the responsive detail modal view are shown in Figures 6.10 and 6.11 below.

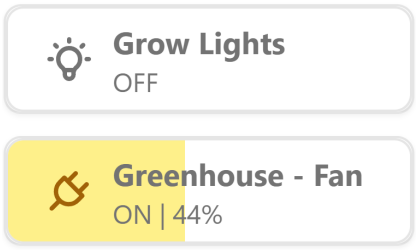


Figure 6.10: Switch card visualization

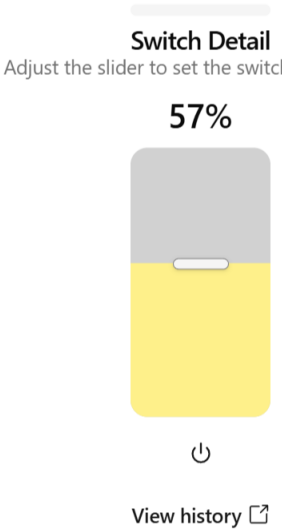


Figure 6.11: Switch's percentual slider

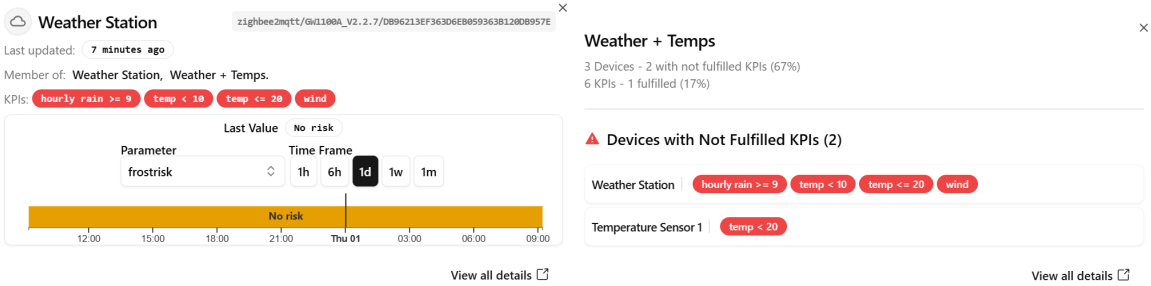
6.5 Detailed Views

The implemented detailed views closely follow the proposed principles in Section 5.3, distinguishing between quick overview modal dialogs and comprehensive dedicated pages.

For devices, the modal view presents essential real-time information, including the time when the device last sent any updates, associated groups, and any failing KPIs, which also update in real-time. Below these key attributes, users can preview the history of a selected parameter and also adjust the time frame. The selected parameter can also be passed into

this modal; for instance, upon clicking on a line in a line chart, the exact parameter is immediately displayed. The historical data is visualized using either a line chart or a visualization of sequential states, as shown in Figure 6.12a.

Group detail modals are accessible by clicking a failing group in the dashboard's static top section, and provide a brief overview of the group. At the top, users are presented with percentages of failing KPIs and devices, followed by a list of devices and their specific failing KPIs. Each device in this list is also clickable and presents its detailed modal view if clicked. This modal view is shown in Figure 6.12b.



(a) Device detail modal view (b) Group detail modal view

Figure 6.12: Detail modals used to display detailed views.

Both device and group modals feature a "View all details" button that navigates users to the dedicated detail pages. These pages are especially important for drill-down operations, presenting a complete overview of all KPIs and settings for the selected device or group. The dedicated and modal views are designed to be fully responsive and accessible. The dedicated device detail page, as shown in Figure 6.13 below, offers means for comparative analysis as well. Users can choose to view a specific parameter of a device, compare the parameter across different time ranges, or contrast it with the parameter from another device. However, due to allowing the comparison across different time ranges, the tooltips in this mode had to be disabled, as *Nivo* does not support such features, and in this case, two separate charts are overlaid.

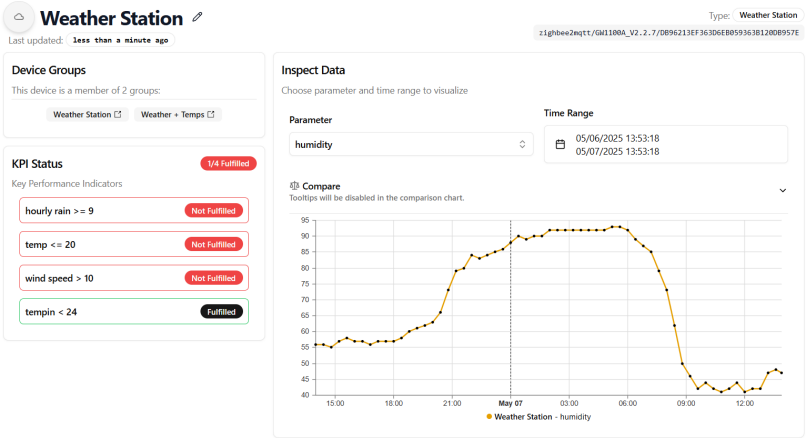


Figure 6.13: Device details dedicated page view

6.6 Responsivity and User Experience

The responsivity of all implemented components was one of the primary focuses, from the dashboard, whose responsivity was already discussed in Section 6.3, through to the builders, modal windows, and dedicated detail pages.

Mobile devices pose a serious challenge for dashboards, which generally benefit from space and a single-page overview. With limited screen real estate, vertical scrolling is unavoidable when dealing with larger dashboards. The dashboard aims to deliver as much information as possible, while mitigating non-data pixels, previously discussed in Section 3.2.

User interactions are also adapted for touch interfaces, supporting gestures such as long-press and horizontal swipes on the dashboard. When using line charts, a key usability issue emerged on touch devices. The crosshair tool for identifying data points was challenging to use because the screen would move during the interaction. To address this, a more advanced interaction mechanism was implemented, which involves long-pressing on the chart to lock the screen position, thus allowing free movement across the chart without any interruptions. This behavior, described earlier in Section 6.4, is guided by the logic described in Figure 6.14 by the flow diagram. The use of a long-press gesture was necessary because, without incorporating a delay and calculating positional deltas, false positives occurred, causing the screen to be locked even during regular scrolling.

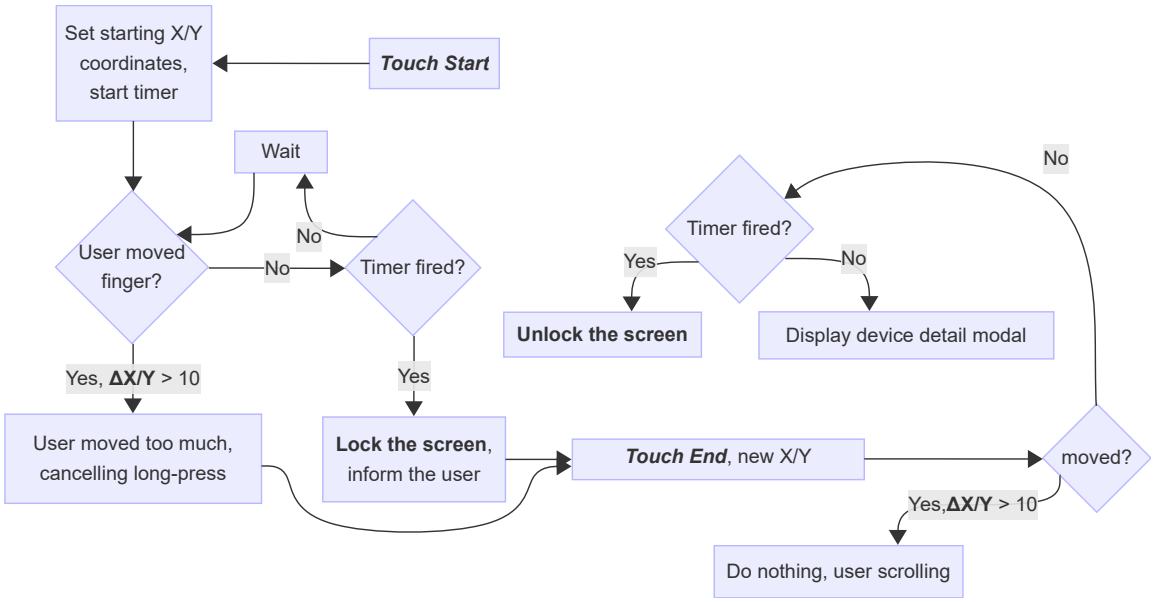


Figure 6.14: Line chart long/short press flow diagram

Modal windows, which appear over rendered content, are used throughout the application and display in two forms: dialog and "drawer". This conditional rendering is determined by the available width and returns a drawer for small-screen devices, which is a more native and user-friendly interface.

Chapter 7

Testing

The objective of the testing phase was to evaluate the functionality, usability, and overall clarity of the dashboard and its associated components under realistic usage conditions. The testing was carried out according to predefined testing scenarios, listed in Appendix B, that were carefully designed to reflect real-world tasks that a typical user might encounter regularly when using the system. Although these scenarios do not cover the entirety of the implemented parts, the primary focus was put on the dashboard itself, both because it represents the system's core functionality and to prevent the scenarios from being too broad or time-consuming. Furthermore, feedback was collected during an open discussion in a group meeting, where technically proficient participants, including two experts in the IoT domain, one of whom works for Logimic and had experience working on a similar dashboard solution, were guided through the system.

The testing environment for the application also followed real-world conditions. The application was deployed on a server, where it was remotely accessed primarily via mobile phones using browsers, such as Safari or Chrome.

7.1 In-Person Testing with Users

The testing sessions were conducted in person, where the participants were directly observed as they navigated the interface. Moments of hesitation or confusion were given particular attention because they often pointed to areas where the user experience could be improved. In order to gain a better understanding of their expectations and thought processes, participants were also asked to think aloud. However, because the respondents had never seen this system before, it was difficult to target and locate these moments.

Using the described qualitative approach, recurring patterns began to emerge, which highlighted the most critical areas that needed adjustment. One of the most frequently reported usability issues concerned the dedicated move buttons for adjusting the dashboard's layout. These buttons were intentionally designed to be small, as the cards are resizable, and some of them can be reduced to very small sizes. Without making them smaller, occasionally, overlaps with other elements occurred, such as the edit or delete icons. Users, however, only noted this and automatically proceeded to use the pinch-to-zoom gesture instead.

Another significant issue emerged during the visualization building process, where, after choosing a specific visualization type, users seemed to expect an automatic transition to its configuration interface. Instead, respondents were required to click the "Next" but-

ton at the bottom or the "Builder" tab at the top of the dialog. This extra step was not immediately obvious, leading to hesitation and confusion. In some cases, respondents instinctively attempted to use swipe gestures to navigate forward, but these gestures were not supported within this modal view.

While in edit mode, the dashboard initially synchronized changes after each tile movement or resizing. This approach had two major drawbacks: an excessive amount of GraphQL mutations and the repetitive appearance of pop-up notifications that confirm that the changes have been saved. Causing unnecessary and unwanted distractions in the user interface.

7.2 Open Discussion with Colleagues and Logimic Expert

In addition to individual testing, feedback was collected during a group discussion with colleagues who had higher technical expertise, including two domain experts in the IoT field. One concern involved the padding of the visualization cards, which appeared slightly off. This, however, as even noted by the respondent, balances the information density and visual aspects of the dashboard, where the initial goal was to minimize these paddings and thus reduce non-data pixels. Another user interface problem was highlighted in the visualization builders, where multiple cards were nested, resulting in excessive borders and cluttering the view.

One of the experts initiated a discussion on performance, concerning the data fetching logic of the dashboard. As described in Section 6.3, the gradual retrieval mechanism of data was recognized as a crucial feature. The expert further noted its positive impact on reducing required computational power and improving overall efficiency.

After noting down these issues and valuable feedback, most of the issues were addressed through iterative improvements, and if possible, also consulted back with the participant. The move buttons were adjusted to be larger if possible, based on the available width and height of the card.

In the visualization builder, the user is now automatically redirected after clicking on a visualization for the second time, which seemed to be the respondents' intuitive behavior, without the need to press the "Next" button. Swipe gestures were also added that navigate the user to the next tab if a visualization is selected, as they proved to be carried out intuitively by the users. Furthermore, the swipe gestures are enabled even inside the builder to navigate back; however, a user is first prompted, informing them that their changes will be discarded.

The intermediate configuration synchronizations with the database were prevented by accumulating the changes and then sending them once the user leaves edit mode. Thus, solving both mentioned issues by dramatically reducing the number of API requests and preventing unnecessary pop-ups.

In summary, ten participants, composed of eight *(80%)* men and two women *(20%)*, with ages ranging from 22 to 53, were involved in the testing phase. The selected group of participants was intentionally diverse, and the same applies to their technical expertise, ranging from beginners to advanced users. This diversity revealed interesting differences with beginners often highlighting issues related to interface clarity or intuitive navigation, while more experienced users pointed out problems with efficiency, workflow optimizations, and feature expectations.

7.3 Proposed Extensions and Future Improvements

In addition to addressing the identified usability issues, the testing phase also revealed multiple opportunities and ideas for further development. Several participants also directly provided suggestions for enhancing the functionality beyond its current capabilities.

One commonly proposed feature was the ability to favorize a group card with KPIs, or the ability to insert it as a card into the user-customizable area of the dashboard. Implementing such behavior should not be too challenging and would require creating a new visualization builder with a group picker. Another notable suggestion was focused on user access and feature control, where administrators could completely disable or restrict access to specific features, such as allowed visualizations, or lock a user's dashboard from further adjustments.

From my point of view and beyond the participants' suggestions, potential extensions of this solution could include real-time notifications concerning failing KPIs, customizable themes of the visualizations, or better line chart visualization with the ability to preview one series of data at a time and fading others. Lastly, more visualizations could be added to the system, such as box plots, as described in Section 3.3, which provide an interesting alternative to bar and line charts, although their complexity may limit their usability in some cases.

Chapter 8

Conclusion

The primary objective of this thesis was to address the growing complexity of IoT ecosystems by designing and implementing a responsive, intuitive, and user-friendly dashboard solution for IoT devices, especially within smart homes.

The result is a dashboard and visualizations that are both responsive and meet the objectives. By combining KPIs and visualizations such as tables, timelines, line charts, or bullet graphs, the dashboard offers users an intuitive tool for managing their smart home systems. The proposed dashboard view and its elements are based on established dashboard research principles, such as information layout and density, ensuring that users are naturally drawn to critical elements.

The dashboard was designed with scalability in mind and supports categorization through tabs to improve organization. Users are also allowed to configure entirely independent layouts per device, based on its screen size (e.g., laptop, tablet, phone). This design choice allows users to include only essential information on mobile devices, while using more detailed layouts on large-screen devices. Structurally, the dashboard is divided into two main sections. The topmost section is static and reserved for highlighting groups with failing KPIs that may require immediate attention. Following this section lies the user-customizable area, which supports scheduled data re-fetching for historical data and live updates for displaying current values.

Detailed views also form a valuable part of this implementation, as they accompany the dashboard and help reduce unnecessary visualizations in the main view. These are particularly useful for occasional monitoring of device parameters that are not frequently observed. Furthermore, the builders, which are also repurposed as editors, serve critical functionality, allowing users to add new or edit existing visualizations, and help guide users through the process with their form validations.

The implemented solution was presented at the Excel@FIT 2025[1] conference, where attendees had the opportunity to try the application themselves. The solution was also tested for its usability, which revealed issues mainly from the user interface standpoint. Many of which were addressed, with some respondents leaving suggestions for further features or improvements.

[1] Excel@FIT Conference homepage – https://excel.fit.vutbr.cz/

Bibliography

[1] *Transmission Control Protocol* [online]. RFC Editor, september 1981. Available at: https://doi.org/10.17487/RFC0793. [cit. 2025-04-24].

[2] Al Sarawi, S.; Anbar, M.; Alieyan, K. and Alzubaidi, M. Internet of Things (IoT) communication protocols: Review. In: IEEE. *2017 8th International Conference on Information Technology (ICIT)* [online]. July 2017, p. 685–690. ISBN 978-1-5090-6332-1. Available at: https://doi.org/10.1109/ICITECH.2017.8079928. [cit. 2024-09-06].

[3] Alahi, M. E. E.; Sukkuea, A.; Tina, F. W.; Nag, A.; Kurdthongmee, W. et al. Integration of IoT-Enabled Technologies and Artificial Intelligence (AI) for Smart City Scenario: Recent Advancements and Future Trends. *Sensors* [online], 2023, vol. 23, no. 11. ISSN 1424-8220. Available at: https://doi.org/10.3390/s23115206. [cit. 2024-10-01].

[4] Bach, B.; Freeman, E.; Abdul Rahman, A.; Turkay, C.; Khan, S. et al. Dashboard design patterns. *IEEE Transactions on Visualization and Computer Graphics* [online]. IEEE, 2022, vol. 29, no. 1, p. 342–352. Available at: https://doi.org/10.48550/arXiv.2205.00757. [cit. 2024-11-20].

[5] Bahashwan, A. A.; Anbar, M.; Abdullah, N.; Al Hadhrami, T. and Hanshi, S. M. Review on Common IoT Communication Technologies for Both Long-Range Network (LPWAN) and Short-Range Network. In: Saeed, F.; Al Hadhrami, T.; Mohammed, F. and Mohammed, E., ed. *Advances on Smart and Soft Computing* [online]. Singapore: Springer Singapore, 2021, p. 341–353. ISBN 978-981-15-6048-4. Available at: https://doi.org/10.1007/978-981-15-6048-4_30. [cit. 2024-10-10].

[6] Bradley, S. Design Principles: Compositional Flow And Rhythm. *Design principles: Compositional flow and rhythm* [online]. April 2015. Available at: https://www.smashingmagazine.com/2015/04/design-principles-compositional-flow-and-rhythm/. [cit. 2024-11-12].

[7] Brous, P. and Janssen, M. Advancing e-Government Using the Internet of Things: A Systematic Review of Benefits. In: Tambouris, E.; Janssen, M.; Scholl, H. J.; Wimmer, M. A.; Tarabanis, K. et al., ed. *Electronic Government* [online]. Cham: Springer International Publishing, 2015, p. 156–169. ISBN 9783319224787. Available at: https://doi.org/10.1007/978-3-319-22479-4_12. [cit. 2024-10-19].

[8] Browning, D. *IoT Started with a Vending Machine* [online]. 25. july 2018. Available at: https://www.machinedesign.com/automation-iiot/article/21836968/iot-started-with-a-vending-machine. [cit. 2024-10-03].

[9] Bureš, M. *System for Processing Data from Smart Devices* [online]. Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Available at: https://www.vut.cz/en/students/final-thesis/detail/153937. [cit. 2025-04-18].

[10] Darroudi, S. M. and Gomez, C. Bluetooth low energy mesh networks: A survey. *Sensors* [online]. MDPI, june 2017, vol. 17, no. 7. ISSN 1424-8220. Available at: https://doi.org/10.3390/s17071467. [cit. 2025-04-22].

[11] Davies, R. *Internet of Things, Opportunities and Challenges* [online]. Briefing. European Parliamentary Research Service, European Parliament, may 2015. Available at: https://www.europarl.europa.eu/thinktank/en/document/EPRS_BRI(2015)557012. [cit. 2024-10-14].

[12] Eckerson, W. W. *Performance dashboards: measuring, monitoring, and managing your business*. 2nd ed. John Wiley & Sons, 2010. ISBN 9780470589830.

[13] Few, S. *Information Dashboard Design: The Effective Visual Communication of Data*. 1st ed. O'Reilly Media, Inc., 2006. ISBN 0596100167.

[14] Few, S. Dashboard Confusion Revisited. *Perceptual Edge* [online]. March 2007th ed. Perceptual Edge, March 2007, no. 1, p. 1–6. Available at: https://perceptualedge.com/articles/visual_business_intelligence/dboard_confusion_revisited.pdf. [cit. 2024-11-10].

[15] Few, S. and Edge, P. Dashboard Confusion. *Perceptual Edge* [online], march 2004, no. 1, p. 1–4. Available at: https://www.perceptualedge.com/articles/ie/dashboard_confusion.pdf. [cit. 2024-11-10].

[16] Gama, J.; Pashami, S.; Bifet, A.; Sayed Mouchawe, M.; öning, H. et al. *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning: Second International Workshop, IoT Streams 2020, and First International Workshop, ITEM 2020, Co-located with ECML/PKDD 2020, Ghent, Belgium, September 14-18, 2020, Revised Selected Papers* [online]. 1st ed. Netherlands: Springer Nature, january 2021. Communications in Computer and Information Science. ISBN 3030667707. [cit. 2024-10-03].

[17] Gardner, B. S. Responsive web design: Enriching the user experience. *Sigma Journal: Inside the Digital Ecosystem*, 2011, vol. 11, no. 1, p. 13–19.

[18] González García, C.; Meana Llorián, D.; Pelayo García Bustelo, B. and Cueva Lovelle, J. A review about Smart Objects, Sensors, and Actuators. *International Journal of Interactive Multimedia and Artificial Intelligence* [online], january 2017, vol. 4, p. 7–10. Available at: https://doi.org/10.9781/ijimai.2017.431. [cit. 2024-10-16].

[19] Greengard, S. *The internet of things*. 1st ed. Cambridge: The MIT Press, march 2015. ISBN 9780262527736.

[20] Greengard, S. *Internet of Things* [online]. 14. november 2022. May 08, 2024. Available at: https://www.britannica.com/science/Internet-of-Things. [cit. 2024-10-19].

[21] Hassenzahl, M. and Tractinsky, N. User experience - a research agenda. *Behaviour & Information Technology* [online]. Taylor & Francis, 2006, vol. 25, no. 2, p. 91–97. Available at: https://doi.org/10.1080/01449290500330331. [cit. 2024-12-29].

[22] Haxhibeqiri, J.; De Poorter, E.; Moerman, I. and Hoebeke, J. A Survey of LoRaWAN for IoT: From Technology to Application. *Sensors* [online], 2018, vol. 18, no. 11. ISSN 1424-8220. Available at: https://doi.org/10.3390/s18113995. [cit. 2024-10-02].

[23] Joukl, M. *Web Application for Managing Smart Devices* [online]. Brno, 2025. Bachelor's Thesis. Brno University of Technology, Faculty of Information Technology. Available at: https://www.vut.cz/en/students/final-thesis/detail/161070. [cit. 2025-05-02].

[24] Kamburjan, E.; Din, C. C.; Schlatte, R.; Tarifa, S. L. T. and Johnsen, E. B. Twinning-by-Construction: Ensuring Correctness for Self-adaptive Digital Twins. In: Margaria, T. and Steffen, B., ed. *Lecture notes in computer science* [online]. Cham: Springer International Publishing, October 2022, p. 188–204. Lecture Notes in Computer Science. ISBN 978-3-031-19849-6. Available at: https://doi.org/10.1007/978-3-031-19849-6_12. [cit. 2024-11-04].

[25] Kocakulak, M. and Butun, I. An overview of Wireless Sensor Networks towards internet of things. In: IEEE. *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)* [online]. January 2017, p. 1–6. ISBN 978-1-5090-4228-9. Available at: https://doi.org/10.1109/CCWC.2017.7868374. [cit. 2024-11-26].

[26] Minerva, R.; Lee, G. M. and Crespi, N. Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models. *Proceedings of the IEEE* [online], 2020, vol. 108, no. 10, p. 1785–1824. Available at: https://doi.org/10.1109/JPROC.2020.2998530. [cit. 2024-10-31].

[27] Montori, F.; Contigiani, R. and Bedogni, L. Is WiFi suitable for energy efficient IoT deployments? A performance study. In: IEEE. *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)* online. September 2017, p. 1–5. ISBN 978-1-5386-3906-1. Available at: https://doi.org/10.1109/RTSI.2017.8065943. [cit. 2025-01-27].

[28] Munzner, T. *Visualization Analysis and Design*. 1st ed. CRC Press, 2014. AK Peters Visualization Series. ISBN 9781498759717.

[29] Nuzzo, R. L. The Box Plots Alternative for Visualizing Quantitative Data. *PM&R* [online], 2016, vol. 8, no. 3, p. 268–272. ISSN 1934-1482. Available at: https://doi.org/10.1016/j.pmrj.2016.02.001. [cit. 2025-03-11].

[30] Parmenter, D. *Key performance indicators: developing, implementing, and using winning KPIs*. 2nd ed. John Wiley & Sons Inc, 2010. ISBN 978-0470545157.

[31] Pereira, T.; Impagliazzo, J.; Santos, H. and Chen, J. *Internet of Everything: Second EAI International Conference, IoECon 2023, Guimarães, Portugal, September 28-29, 2023, Proceedings* [online]. 1st ed. Cham: Springer International Publishing AG, 2024. Lecture notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. ISBN 9783031515712. [cit. 2024-10-25].

[32] Sinha, S. *State of IoT 2024: Number of connected IoT devices* [online]. 3. september 2024. Available at: https://iot-analytics.com/number-connected-iot-devices/. [cit. 2024-10-17].

[33] Sisinni, E.; Saifullah, A.; Han, S.; Jennehag, U. and Gidlund, M. Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Transactions on Industrial Informatics* [online], 2018, vol. 14, no. 11, p. 4724–4734. Available at: https://doi.org/10.1109/TII.2018.2852491. [cit. 2025-05-10].

[34] Soori, M.; Arezoo, B. and Dastres, R. Internet of things for smart factories in industry 4.0, a review. *Internet of Things and Cyber-Physical Systems* [online], april 2023, vol. 3, p. 192–204. ISSN 2667-3452. Available at: https://doi.org/10.1016/j.iotcps.2023.04.006. [cit. 2024-11-10].

[35] Syed, A. S.; Sierra Sosa, D.; Kumar, A. and Elmaghraby, A. *Smart Cities* [online], 2021, vol. 4, no. 2, p. 429–475. ISSN 2624-6511. Available at: https://doi.org/10.3390/smartcities4020024. [cit. 2024-10-02].

[36] Tozlu, S.; Senel, M.; Mao, W. and Keshavarzian, A. Wi-Fi enabled sensors for internet of things: A practical approach. *IEEE Communications Magazine* [online], 2012, vol. 50, no. 6, p. 134–143. Available at: https://doi.org/10.1109/MCOM.2012.6211498. [cit. 2024-11-03].

[37] Tufte, E. R. *Beautiful Evidence*. 1st ed. Graphics Press, may 2006. ISBN 0-9613921-7-7.

[38] Tufte, E. R. and Graves Morris, P. R. *The visual display of quantitative information*. Graphics press Cheshire, CT, 1983. ISBN 9780961392147.

[39] Vagdevi, P.; Nagaraj, D. and Prasad, G. V. Home: IOT based home automation using NFC. In: IEEE. *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)* [online]. 2017, p. 861–865. ISBN 978-1-5090-3243-3. Available at: https://doi.org/10.1109/I-SMAC.2017.8058301. [cit. 2024-12-08].

[40] Vinod Kumar, T. M. E-Governance for Smart Cities. In: Vinod Kumar, T. M., ed. *E-Governance for Smart Cities* [online]. Singapore: Springer Singapore, 2015, p. 1–43. ISBN 978-981-287-287-6. Available at: https://doi.org/10.1007/978-981-287-287-6_1. [cit. 2024-10-27].

[41] Want, R. An introduction to RFID technology. *IEEE Pervasive Computing* [online], 2006, vol. 5, no. 1, p. 25–33. Available at: https://doi.org/10.1109/MPRV.2006.2. [cit. 2024-10-9].

[42] Wilke, C. O. *Fundamentals of Data Visualization*. O'Reilly Media, Inc., Sebastopol, CA, 2019, 2019. ISBN 9781492031055.

[43] Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L. and Zorzi, M. Internet of Things for Smart Cities. *IEEE Internet of Things Journal* [online], 2014, vol. 1, no. 1, p. 22–32. Available at: https://doi.org/10.1109/JIOT.2014.2306328. [cit. 2024-10-01].

[44] Zhang, Y. *Digital Twin: Architectures, Networks, and Applications* [online]. 1st ed. Cham: Springer Nature, 2024. Simula SpringerBriefs on Computing. ISBN 9783031518195. [cit. 2024-11-04].

[45] Zhou, B.; Li, W.; Chan, K. W.; Cao, Y.; Kuang, Y. et al. Smart home energy management systems: Concept, configurations, and scheduling strategies. *Renewable and Sustainable Energy Reviews* [online], 2016, vol. 61, p. 30–40. ISSN 1364-0321. Available at: https://doi.org/10.1016/j.rser.2016.03.047. [cit. 2024-10-23].

[46] Zhu, Q.; Wang, R.; Chen, Q.; Liu, Y. and Qin, W. IOT Gateway: BridgingWireless Sensor Networks into Internet of Things. In: IEEE. *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing* [online]. December 2010, p. 347–352. ISBN 978-1-4244-9719-5. Available at: https://doi.org/10.1109/EUC.2010.58. [cit. 2024-11-28].

Appendix A

Contents of the included storage media

```
xolesa00/ - Root directory
├── implementation/ - Source code of implementation
│   ├── diff.txt - Generated file highlighting authorship of each line
│   │   of code, distinguishing between this thesis's work and my col-
│   │   league Marek Joukl, as the application was developed together
│   │   in two separate theses
│   └── src
│       ├── components/
│       │   └── ui/ - Primarily reserved for shadcn/ui components
│       ├── context/
│       ├── features/
│       │   └── dashboard/ - Most of the components used by the dashboard
│       ├── generated/ - Generated GraphQL models, derived from defini-
│       │   tions in the graphql/ directory
│       ├── graphql/ - GraphQL queries, mutations, and subscriptions
│       ├── hooks/
│       ├── lib/
│       ├── pages/
│       │   ├── controllers/
│       │   └── views/
│       ├── schemas/
│       ├── styles/
│       ├── types/
│       └── utils/
├── text/ - Source files used in the technical report of this thesis
├── excel_at_fit_poster.pdf
├── README.md
└── xolesa00-thesis.pdf
```

Appendix B

Testing scenarios

Note: Please only consult hints if you are unable to complete the task. The goal is to assess the usability of the dashboard and other components without any prior knowledge of its functionality.

Test 1: First Impression Assessment

Setup: Navigate to the `Home` page. Take a look at the dashboard for a few seconds, then assess what draws your attention.
Questions:

- What was the first element that caught your attention?
- Was the view clear and easy to understand?

Test 2: General Understanding of the Dashboard

Setup: When on the `Home` page, try using the dashboard for a minute or two.
Questions:

- Would you know how to add a new visualization?
- How would you edit or delete an existing visualization?
- If you wanted to see more details about a specific device included in any of the visualizations, how would you expect to do that?

Test 3: Layout Changes

Setup: When on the `Home` page, try to change the layout of the dashboard by changing the size of the cards or their position.
Questions:

- Did you have to use any hints to complete the task?
- If done on a mobile device, did you use drag and drop or dedicated buttons to change the layout?

Hint 1: To enter the edit mode, click the edit button at the top right corner of the page.
Hint 2: You can change the size of the cards by dragging the bottom right corner of the card or by using the resizing icons at the bottom. You can change the position of the cards by either dragging the four arrows symbol at the center of the card or by clicking the arrows at the bottom section.

Test 4: Complex Line Chart

Setup: When on the `Home` page, navigate to dashboard section named `Testing`. Then look for a graph named `Test 4. - Complex Chart`, and try using the tooltip in the chart.
Questions:

- Did you have to use any hints to complete the task?
- Can you distinguish all the lines from each other?
- How hard was it to find the chart? Did you have any issues with scrolling?
- Did you manage to display the tooltip? If not, what was the issue?
- Did you have trouble reading the chart's axis labels or tick values?

Hint 1: Tooltips are displayed upon long-pressing and then dragging across the chart. Or by hovering over the chart with a mouse.
Hint 2: The dashboard sections are symbolized by tabs. If there are too many, the tabs are horizontally scrollable.

Test 5: Creating a New Dashboard

Setup: When on the `Home` page, enter the edit mode and try creating a new dashboard category in the tabs section.
Questions:

- Was it clear how to create a new dashboard category?
- Did you have to use any hints to complete the task?
- Did you have any issues with the dashboard creation process?

Hint 1: To enter the edit mode, click the edit button at the top right corner of the page.
Hint 2: You can create a new dashboard category by clicking on the `+ Add tab` button which is the last button on the right side of the tabs section. The tabs are scrollable horizontally, so you may need to scroll to the right to see the button.

Test 6: Building a New Visualization

Setup: When inside your newly added dashboard category, click on the `+` button to create a new visualization, then select any visualization type and try to create a new visualization.
Questions:

- Which visualization did you select?

- Was it clear what you needed to do to create a new visualization?
- Did you have any issues with the visualization creation process?
- Are there any features that you would like to see added to the specific visualization?

Test 7: Overall Experience

Questions:

- What device did you use to test the dashboard? (e.g., mobile, tablet, laptop)
- Did you use touch gestures or a mouse to complete the tests?
- How would you rate your overall experience with the dashboard?
- What did you like the most about the dashboard?
- What did you like the least about the dashboard?
- Do you have any suggestions for improvements?

Note: Please provide any additional comments or feedback you may have about the dashboard. Your input is valuable in helping improve the user experience and general usability. Thank you very much for your time and feedback!