

Chapter 2

Onboarding and Preliminary Functionality Training



2.1 Preliminaries, Registration and Onboarding

Let us start with introducing the definition of a system, which is a part of the universe that can be affected and/or monitored by KIS.ME. Consequently, KIS.ME can be divided into two layers:

- KIS.Device: hardware capable of performing the desired communication tasks within the system,
- KIS.MANAGER: software being a web platform used to affect and/or monitor the system.

Having the system, it is possible to introduce its components, which are defined as assets. They are physical parts of the system and are exemplified by KIS.Devices.

> Unique resource name (URN)

Each KIS.Device is uniquely identified by the URN number, e.g.,

```
urn:rafi:sbox:9c65f93cbf2d.
```

Once assigned, it cannot be further modified.

Thus, an asset group is simply a set of assets. In the current release of KIS.ME, KIS.Devices are divided into

- KIS.BOX: a communication push-button box (see Fig. 2.1),
- KIS.LIGHT: a communication signal lamp (see Fig. 2.2),
- KIS.IO: an input/output communication box (see Fig. 2.3).

Let us start with defining two kinds of LED lights incorporated within each KIS.Device (see Figs. 2.1 and 2.2):

- Status LED: it exhibits the functional state of a KIS.Device and is defined in Table 2.1;

Fig. 2.1 KIS.BOX

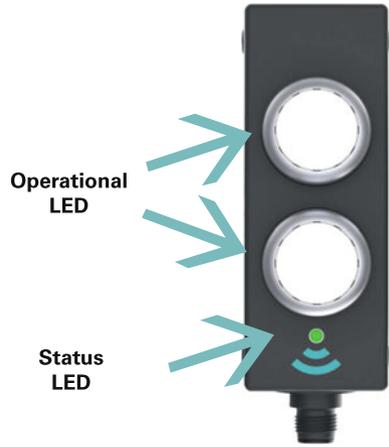


Fig. 2.2 KIS.LIGHT



Fig. 2.3 KIS.IO



Table 2.1 Status LED colors and their meaning

Status LED color	Meaning
Red	Device is booting
Yellow	Device is booted, no WiFi connection
Magenta	Device is connected with WiFi but MQTT Port check Certificate check or NTP time-sync check errors occurred
Blue	Device is connected with WiFi, MQTT Port check and certificate check were successful No connection to KIS.MANAGER
Green	Device is connected to KIS.MANAGER
Turquoise	Update in progress
Flashing Magenta (2Hz)	MQTT Port check failed
Flashing Magenta (1Hz)	NTP time-sync check failed
Flashing Magenta (0.5Hz)	Certificate check failed

Table 2.2 Predefined operational LED colors

Color	RGB HEX code	Integer value
Blue	#0000FF	0
Turquoise	#00FFFF	1
Black	#000000	2
Green	#00FF00	3
Magenta	#FF00FF	4
Red	#FF0000	5
White	#FFFFFF	6
Yellow	#FFFF00	7

Operational LED: it exhibits the individual operational state of a KIS.Device and can be defined by the user using the set of colors provided in Table 2.2.

Note that KIS.IO can be perceived as a simplified version of KIS.BOX without buttons. Subsequently, a general KIS.ME framework overview can be introduced, which is portrayed in Fig. 2.4. Subsequently let us proceed to introduce KIS.Device functionalities. As can be observed in Fig. 2.4, each KIS.BOX can be perceived as a human-machine interface (HMI) with two push-buttons, linked with the KIS.MANAGER via WiFi. Each pushbutton contains an operational LED, which illuminates in an RGB color. However, for the sake of simplicity, the list of colors is limited in KIS.MANAGER to those presented in Table 2.2. Thus, they can be identified by either the RGB HEX code or an integer value. It should be also noted that the black color is used to signify the fact that a corresponding operational LED is not lit. Compared to the KIS.BOX, KIS.LIGHT has limited functionalities as its pri-

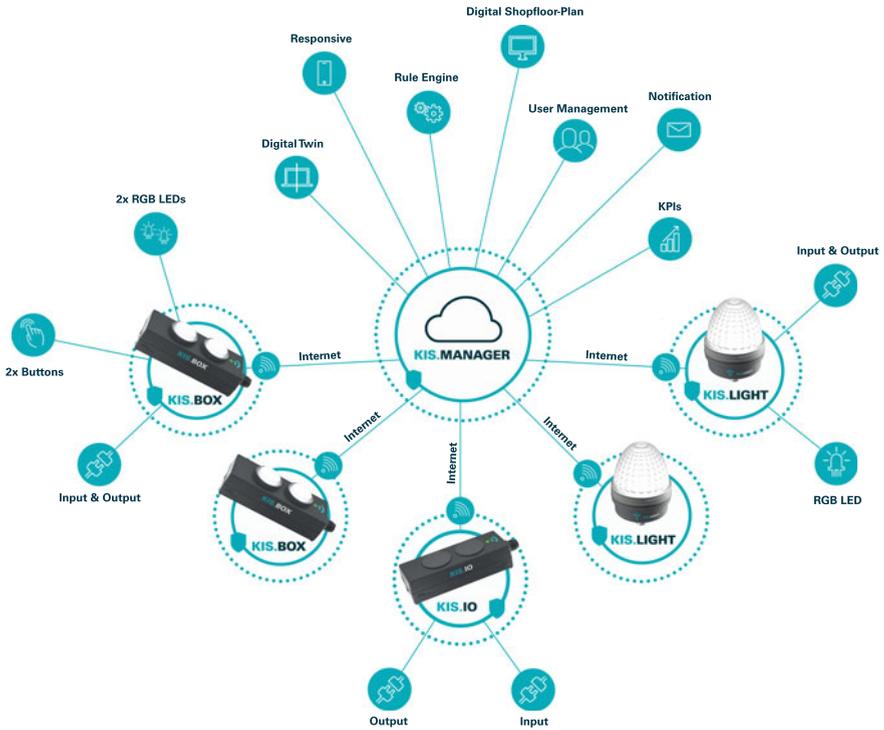


Fig. 2.4 General KIS.ME overview

Table 2.3 KIS.Device parameters

Name	Value
Luminous element color	RGB
Degree of protection	IP65
WLAN standard	IEEE 802.11 b/g/n 2.4 GHz
Connection terminal	M12 8-pin A-coded
Operating voltage	5 ± 10% V, 24 ± 20% V
GPIO	2 inputs/ 2 outputs

Table 2.4 M12 PIN specification

PIN 1	PIN 2	PIN 3	PIN 4	PIN 5	PIN 6	PIN 7	PIN 8
VCC voltage	In 1	GND	In 2	Out 1	Out 2	USB D+	USB D-

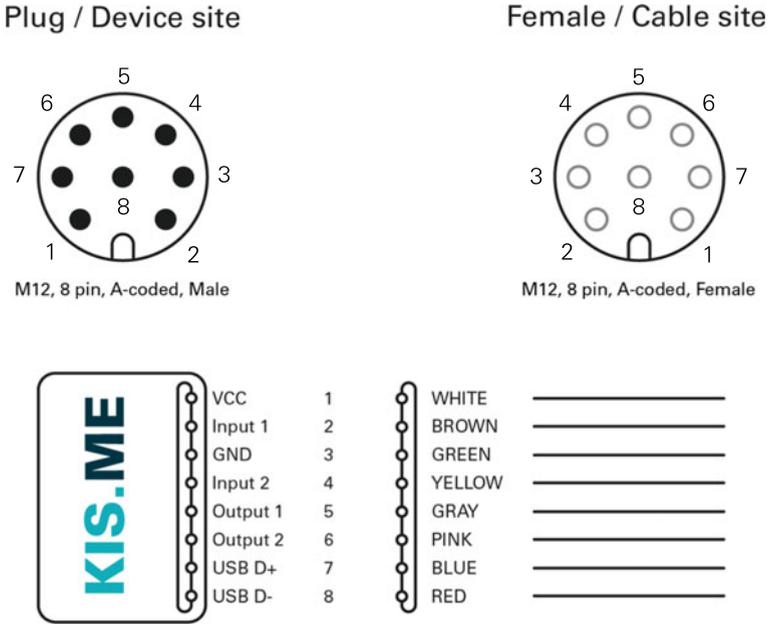


Fig. 2.5 KIS.Device M12 connections

mary purpose is to be a signalling lamp with one operational LED only. Finally, the essential technical parameters of KIS.Devices are given in Table 2.3. It can be noted that each KIS.Device is fed with an M12 8-pin A-coded connection terminal, while the purpose of particular PINs is given in Table 2.4. The operating voltages can be either 5 V or 24 V. In the first case, a KIS.Devices are fed through USB, while in the second one, the respective 24 V voltage is fed through PINs 1 and 3. Each KIS.Device possesses a general purpose input output (GPIO) interface, which is available while using a 24 V power supply only. As can be observed in Table 2.4, GPIO is composed of two digital inputs and two outputs. Thus, each KIS.Device enables attaining the functionalities portrayed in Fig. 2.5, which can be further exploited in a large number of practical applications. Finally, it should be noted that a full technical documentation of KIS.Devices is available at <https://kisme.rafi.de/en/>. Having the hardware layer provided, let us proceed to the software one. As can be observed in Fig. 2.4, the main KIS.MANAGER features can be summarized as follows:

- System assets can be easily digitalized through digital twins of KIS.Devices.
- The system is divided into workspaces, i.e., selected parts of the system, which inherit its desired set of assets.
- A graphical representation of the workspaces can be introduced using floorplans.
- An asset’s behaviour can be affected and monitored using the Rule engine functionality, which makes it possible to implement functional IF-THEN rules governing interactions between assets.

- The behaviour of the system and the associated assets can be instantly monitored using Datapoints, which correspond to possibly time-varying properties of KIS.Devices. They can be also defined as exchanged variables between a KIS.Device and KIS.MANAGER.
- Performance of the system and the associated assets can be periodically determined using Datapoint-based key performance indicators (KPIs).
- It allows defining users as human beings with granted access determined by membership to a given user group.
- System behaviour and performance can be visualized using a set of time-driven plots and aggregated charts.
- E-mail notifications pertaining to system behaviour can be predefined and automatically distributed.
- A high security level is attained with Message Queuing Telemetry Transport (MQTT), which is a standard messaging protocol for the IoT. Moreover, designated certificates are used for secure authentication.

Having a general overview of both hardware and software functionalities, let us proceed to onboarding, i.e., the process of linking KIS.Devices with the KIS.MANAGER. However, a preliminary step towards onboarding is to register at the KIS.MANAGER, which can be easily realized with <https://kismanager.rafi.de>. Once a user company account is created, a compulsory checklist should be verified:

- a KIS.Device,
- an M12-to-USB cable (see Table 2.4),
- a computer/tablet equipped with a web browser and a USB port,
- a company account with admin user rights (see Sect. 2.3),
- WLAN access with permission credentials.

After completing the compulsory checklist, one can perform the onboarding procedure:

Step 1. Connection:

1. Plug-in KIS.Device to a PC/tablet using an M12-to-USB cable.
2. The KIS.Device status LED color should change from red into yellow (see Table 2.1).
3. The KIS.Device is available in the PC/tablet as a mass storage device.

Step 2. Authentication:

1. Open <https://kisme.rafi.de/en/> and go to the Onboarding link.
2. Enter your login and password pertaining to KIS.MANAGER admin rights.
3. Enter your designated WiFi parameters: SSID, password and WLAN encryption mode.
4. Click the save button to generate onboarding.zip containing WiFi parameters and store it onto the PC/tablet.

Step 3. Upload onboarding.zip onto the KIS.Device visible as a mass storage device.

Step 4. Processing onboarding.zip:

1. In progress: the status LED starts to flash in yellow;
2. Completed: the status LED lights constantly in yellow.

Step 5: Onboarding completion:

1. Under an available WiFi connection, the KIS.Device status LED should perform the following cycle: Yellow → Magenta → Blue → Green (see Table 2.1).
2. After refreshing the KIS.MANAGER, the new KIS.Device is available in Main menu → Assets.

> Changing WLAN

The WLAN data, and hence the WiFi network, can be changed on demand by simply repeating the above Step 1–Step 5 onboarding procedure.

> Accessing the KIS.ME demo

It is possible to preview KIS.ME performance without having your own KIS.Devices. For that purpose, a KIS.ME demo platform was designed, which can be accessed by performing the following steps:

1. Go to <https://kisme.rafi.de/en/#demo>.
 2. Access the KIS.ME demo with
 - Username: `demo.kisme@rafi.de`.
 - Password: `Demo1234!`.
-

2.2 Hierarchical Structure: From Assets and Users to Workspaces

The objective of this section is to provide a general KIS.ME-based system structure overview. For that purpose, let us introduce a suitable nomenclature:

User group: a set of users with a predefined KIS.MANAGER rights level,

Asset group: a set of assets,

Workspace: a selected part of the system, which inherits its desired set of assets.

From the above definitions, it is evident that Workspace and Asset group can be somehow perceived as synonyms. Indeed, while going to Main menu → Asset groups, one can see the view which is portrayed in Fig. 2.6. Thus, all system assets associated

Asset Group ▲	Definition
<u>My Devices</u>	Inventory
<u>Workspace 1</u>	Workspace
<u>Workspace 2</u>	Workspace
<u>Workspace 3</u>	Workspace
<u>Workspace 4</u>	Workspace
<u>Workspace 5</u>	Workspace

Fig. 2.6 Asset groups

with all KIS.Devices are contained in the inventory asset group My devices. KIS.ME, which makes it possible to arrange five workspaces (see the column Definition in Fig. 2.6) may inherit the assets contained in the inventory asset group.

> Adding asset groups

The current licence model allows six asset groups, i.e., My inventory, Workspace 1–Workspace 5. In the prospective licence models, it will be possible to add new groups.

Once Asset groups are defined, it is possible to proceed to User groups, which can be reached through Main menu → User groups. As can be seen in the column Description in Fig. 2.7, there are four predefined user groups: Admin, Installer, Operator and Observer, possessing various rights and permissions (see Sect. 2.3 for details). Thus, a single user may belong to these groups, which strictly defines rights and permissions. On the other hand, all five workspaces can also be perceived as user groups.

> User group membership

A single user may belong to multiple user groups, which define rights and permissions. This also means that he can belong to multiple user groups associated with workspaces, which clearly determine access to the desired asset groups.

Fig. 2.7 User groups

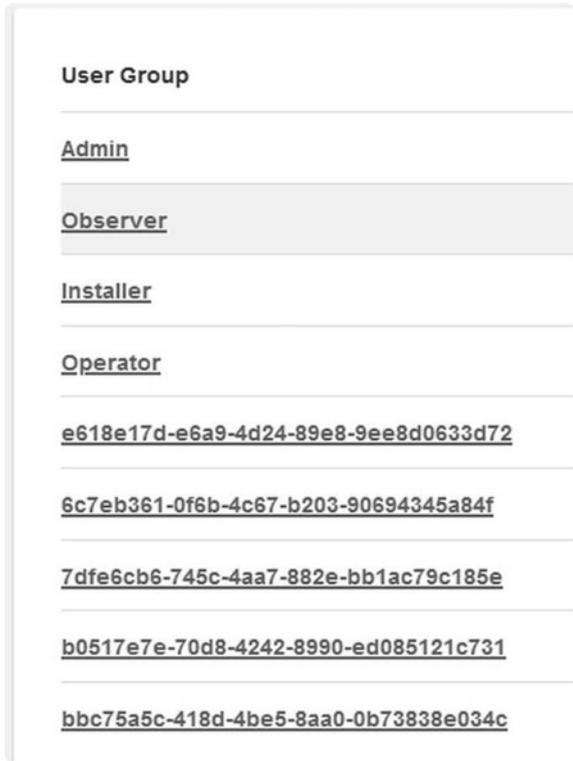
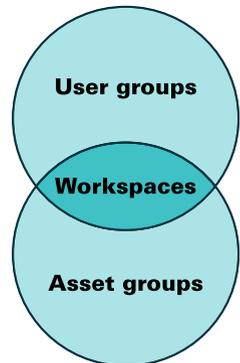


Fig. 2.8 KIS.ME hierarchical structure



Finally, the KIS.ME hierarchical structure can be expressed using Fig. 2.8. Thus, the objective of the subsequent section is to provide a concise overview pertaining to rights and permissions acting inside this structure.

2.3 Rights and Permissions

Let us start with four predefined user groups (see Fig. 2.7), which are initially called Admin, Installer, Operator and Observer. Their concise overview is given in Table 2.5, which uses the following nomenclature:

- Access: the user has access to a given feature;
- Add/delete: the user is able to add and/or deleted a given feature;
- Manage: the user is able to manage the properties of a given feature;
- View: the user is able to view a given feature.

➤ Names and permissions of predefined user groups

It should be noted that the initial names of predefined user groups can be modified. Alterations can be also performed with respect to their permissions (see Fig. 2.14).

2.3.1 User Management

The process of adding a user is very intuitive but requires access with admin rights (see Table 2.5). Under such a condition, adding a new users and assigning them the desired rights boils down to the following steps:

1. Go to Main menu → User management → Users.
2. Push the Create new user button. 
3. In the Master data tab (see Fig. 2.9), provide e-mail, language (locale), time zone, full name, and initial state.

➤ Initial state

The Initial state field determines whether the new user must agree (terms of acceptance pending) to an end user license agreement or whether this can be omitted and the user can be active immediately (Active).

4. In the User groups tab (see Fig. 2.10), push the Assign to User Groups button.
5. By selecting the desired checkboxes, assign a user to a predefined group (Admin, Installer, Operator, Observer; cf. Table 2.5) and to desired workspaces (Fig. 2.11).

Table 2.5 Rights of predefined user groups

Rights	Admin	Installer	Operator	Observer
Add/delete assets				
Manage permitted assets	x	x		
Access to all asset groups	x			x
Access to permitted asset groups	x	x	x	x
Add/delete users	x			
Add/delete dashboards	x	x		
Manage permitted dashboards	x	x	x	
View permitted dashboards	x	x	x	x
Add/delete digital twins	x	x		
Manage permitted digital twins	x	x	x	
Add/delete CDPs and/or KPIs	x	x		
View CDPs and/or KPIs	x	x	x	x
Add/delete e-mail templates	x			
Manage permitted e-mail templates	x			
Add/delete rules in rule engine	x	x		
Manage rules in rule engine	x	x		
View rules in rule engine	x	x	x	x

Fig. 2.9 Creating a new user: master data

Create User

Master Data User Groups

E-mail *

Locale *

Time Zone *

Full Name

Initial state

Terms Acceptance Pending Active

* required

Save **Cancel** **Toggle hidden properties**

> **Admin and workspaces**

Irrespective of the workspaces being assigned, a user with admin rights has access to all of them (see Table 2.5).

6. After pushing the Save button, the user will receive an e-mail that contains a link which enables providing a password.

Finally, any modification pertaining to an existing user can be realized with the above procedure. However, instead of Step 2, an existing user has to be selected from the available user list.

Fig. 2.10 Creating a new user: user groups

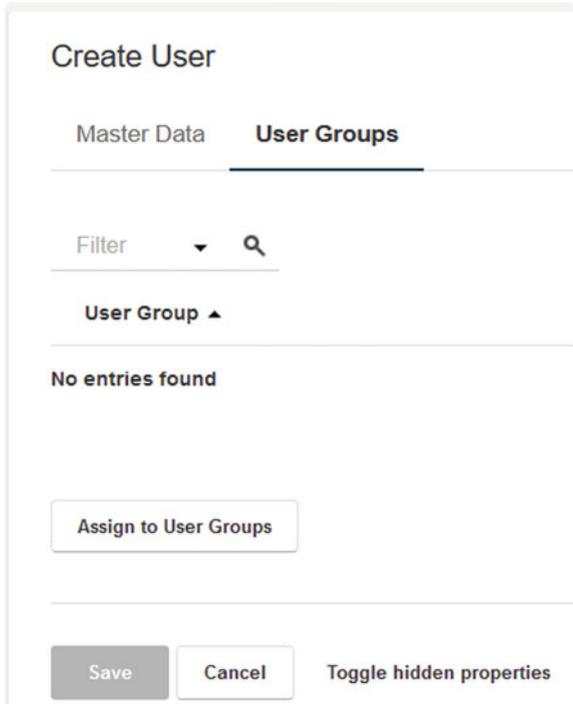


Fig. 2.11 Assigning a user to user groups

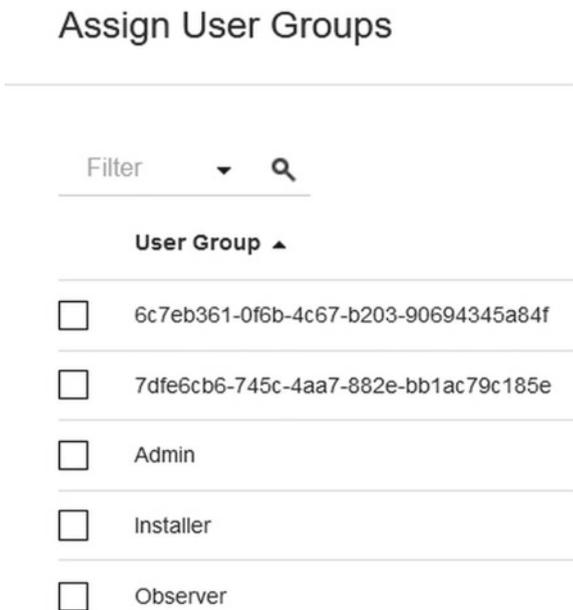
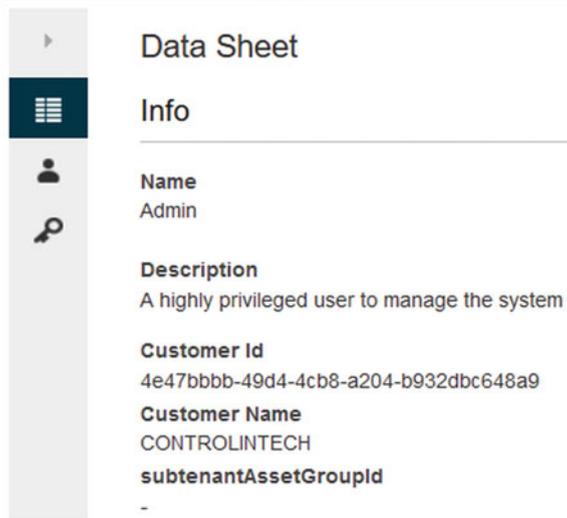


Fig. 2.12 User group data sheet



2.3.2 User Groups and Workspace Management

As mentioned in the preceding section, each user can belong to multiple user groups. The permissions of a given user group to another one can also be freely defined. Indeed, while going to Main menu → User management → User groups, one can see the view portrayed in Fig. 2.7. Subsequently, by selecting any group, e.g., admin, one can see the view shown in Fig. 2.12, which contains essential details about this group. By selecting the Edit button , one can see the view presented in Fig. 2.13. As can be observed, there are two tabs:

- Info: it contains the name, description and other descriptive parameters of a group;
- User Groups with Access Permissions: it pertains to the list of user groups for which a given access group has access permission (see Fig. 2.14).

> Assigning users to a group

It should be mentioned that by proceeding to Main menu → User management → User Groups and then selecting a desired group, one can see the view portrayed in Fig. 2.12. By pushing the Assigned Users button , it is also possible to edit the user assignment.

Fig. 2.13 Editing a user group: info

Edit User Group

Info User Groups with Access Permission

User Group

Description

subtenantAssetGroupId

Not changeable once set and saved

Customer Id

Not changeable once set and saved

Customer Name

Not changeable once set and saved

Advanced settings

Technical name (Autogenerated) *

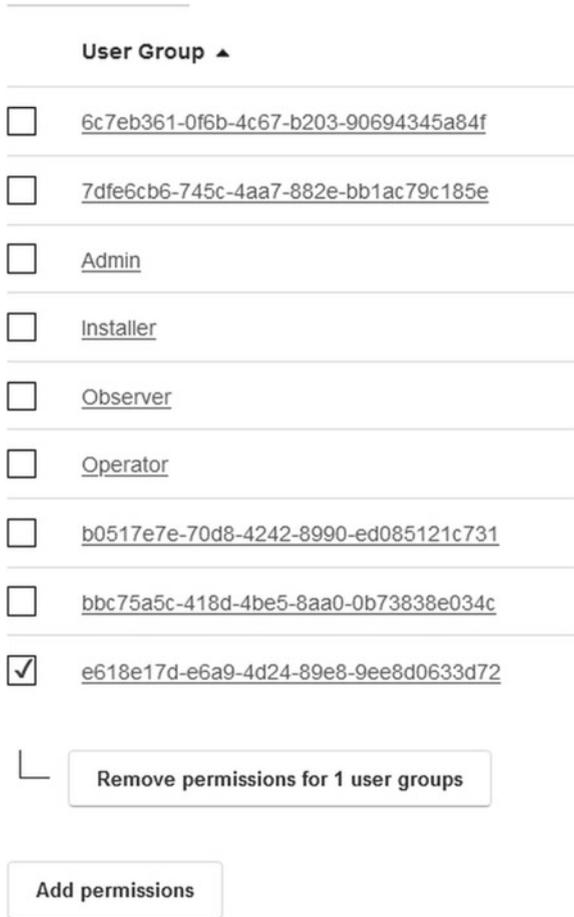
This field is tenant wide unique

*** required**

2.4 Asset Management

The objective of the preceding sections was to perform a suitable introduction to the hierarchical KIS.ME structure along with the KIS.Device onboarding procedure. As a result of performing onboarding on a set of assets, one can see a view similar to that presented in Fig. 2.15. Indeed, it can be easily accessed with Main menu → Assets. This sample view indicates that there are eight KIS.Devices, i.e., four KIS.BOXes and

Fig. 2.14 Editing a user group: user groups with access permissions



four KIS.LIGHTs. It can be also immediately deduced (see the Connection column) that, except for KIS.BOX 0 and KIS.LIGHT 3, all KIS.Devices are connected with KIS.MANAGER via dedicated WiFi(s). Subsequently, by selecting a sample asset, e.g., KIS.BOX 1, one can see the view presented in Fig. 2.16. The objective of the subsequent part of this section is to perform essential asset management concerning

- changing the name of the asset,
- assigning the asset to asset groups,
- obtaining information about current status of an asset.

Let us start with the first task by pushing Data Sheet button . As a result, the view presented in Fig. 2.17 is obtained. Subsequently, the Master Data edit button  can be used to change the name of the asset as shown in Fig. 2.18. Let us proceed to the second task, which pertains to assigning KIS.BOX 1 to desired asset groups. For that purpose, the Asset groups tab should be selected as shown in Fig. 2.19. Finally,

Asset Name ▲	Connection	Asset Groups
KIS.BOX 0	Offline	My Devices
KIS.BOX 1	Online	My Devices
KIS.BOX 2	Online	My Devices
KIS.BOX 3	Online	My Devices
KIS.LIGHT 0	Online	My Devices
KIS.LIGHT 1	Online	My Devices
KIS.LIGHT 2	Online	My Devices
KIS.LIGHT 3	Offline	My Devices

Fig. 2.15 Asset view

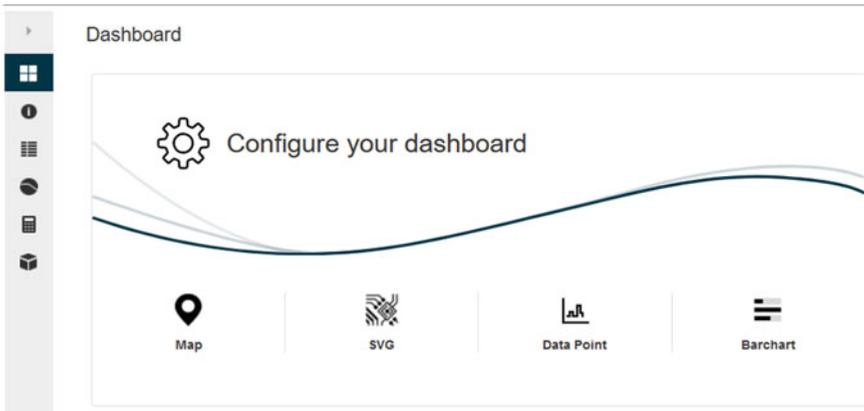


Fig. 2.16 Asset view: KIS.BOX 1

the assignment process reduces to pushing the Assign to Asset Groups button and selecting the desired asset groups as depicted in Fig. 2.20. The process of unassigning an asset from the asset group can be performed in a similar fashion. Indeed, it is enough to check a desired checkbox and push the Unassign button.

Fig. 2.17 Data sheet of KIS.BOX 1

The screenshot displays a mobile application interface for viewing asset data. On the left is a vertical sidebar with several icons: a right-pointing arrow, a 2x2 grid, an information icon, a list icon (highlighted in dark blue), a pie chart, a calculator, and a cube. The main content area is titled 'Data Sheet' and is divided into several sections:

- Master Data**:
 - Asset Name**: KIS.BOX 1
 - URN**: urn:rafi:sbox:9c65f93cc3eb
 - A button labeled 'Show QR Code'.
 - Device Type**: KIS.BOX
- Asset Groups**: Indicated by a badge with the number '1'. Below this is a 'Filter' dropdown menu with a search icon.
- Asset Group**: A section header with an upward-pointing triangle.
- My Devices**: A link to view the user's devices.
- Asset Shadow Datapoints**:
 - isOnline**: true
 - lastUpdate**: 06/20/2021 14:00:08 (+02:00)
 - appVersion**: r32
- Asset Shadow Values**:
 - connection.last**: 06/20/2021 14:00:05 (+02:00)

Fig. 2.18 Changing the asset name

Edit Gateway

Data Sheet Asset Groups Onboarding

Asset Name *

URN *

Transport channel *

Use certificate onboarding

Device Type *

Not changeable once set and saved

*** required**

> Group relationship graph

The relationship between an asset and the associated asset groups can be easily visualized. Indeed, in Fig. 2.17, one can find the Group relationship graph button , which can be used to visualize an associated graph. Such a sample graph is portrayed in Fig. 2.21.

The last task pertains to obtaining information about detailed parameters of an asset by pushing the Info button  (see Fig. 2.16). The above parameters cover the device information divided into the following groups:

Fig. 2.19 Asset and the associated asset groups

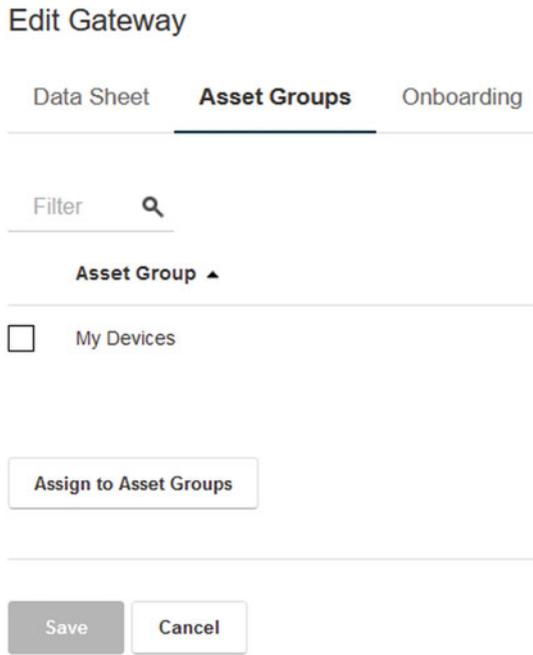


Fig. 2.20 Assigning asset groups

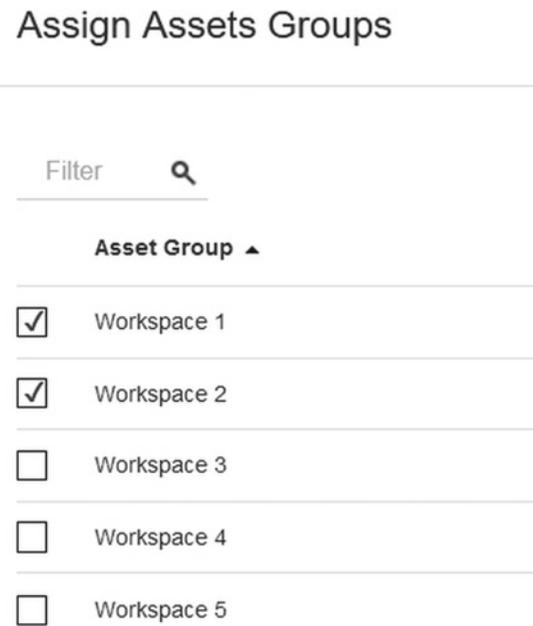
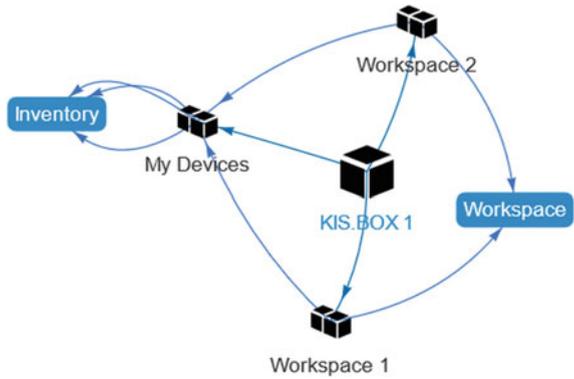


Fig. 2.21 KIS.BOX 1: group relationship graph



Hardware: type (KIS.BOX/KIS.LIGHT), part number, serial number, data matrix code, MAC address, hardware revision,
Software: OS version, application version, microcontroller firmware version,
Network: WiFi SSID, WiFi signal strength, WiFi channel, IP address, subnet, gateway,
Firmware update: a set of detailed parameters including the update status,
Certificate: the certificate expiration date of a KIS.Device.

2.5 Dashboards and Widgets

The view presented in Fig. 2.16 is divided in the so-called Dashboards, which are defined as an overview pages for an asset and/or asset groups. Dashboards can be managed by pushing the Edit dashboard button . After selecting this option, one can perform one of the following tasks (see Fig. 2.22):

- add a new dashboard,
- design or edit an existing dashboard.

The first one is very intuitive and does not need any further explanation as it reduces to providing the name of a new dashboard. As a result, the dashboard is automatically created and displayed as a new tab in the asset view. Thus, let us proceed to designing a dashboard. Each one is composed of widgets. A widget is a component of the interface which makes it possible to perform a desired action. There are nine available widget types, which can be characterized as follows (see Fig. 2.23):

- Digital twin,
- Info,
- Datapoint Chart,
- Data Sheet,

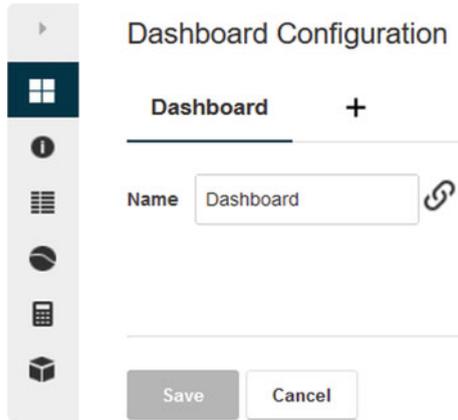


Fig. 2.22 Dashboard design

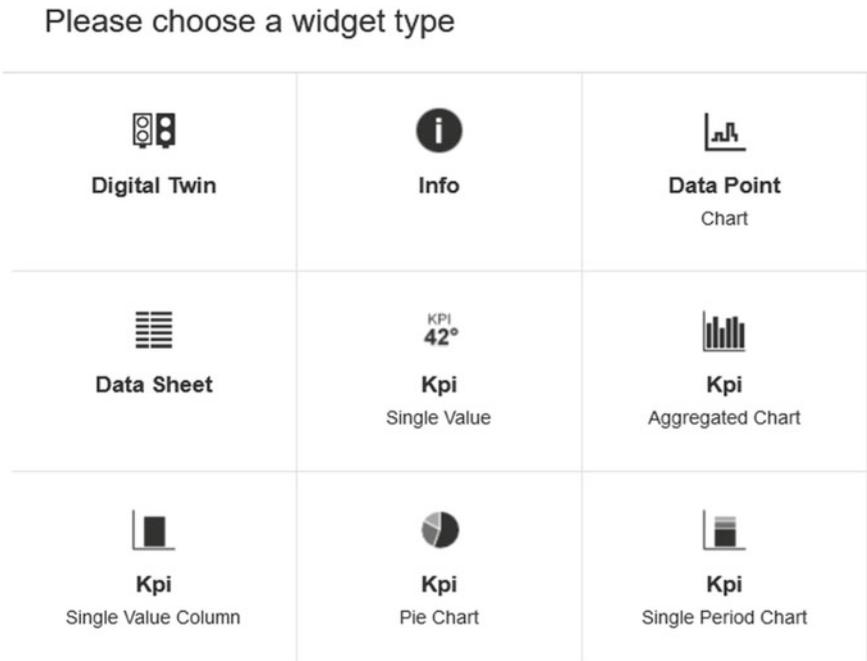


Fig. 2.23 Asset: nine widgets

- KPI Single Value,
- KPI Aggregated Chart,
- KPI Single Value Column,
- KPI Pie Chart,
- KPI Single Period Chart.

The first four widgets can be directly used for digitalization of KIS.Devices as well as the monitoring and analysis of their behaviour. The remaining five widgets require suitable preprocessing using KPIs, which are discussed in Sect. 4.1.2.

2.6 Digital Twin Design

The objective of this section is to introduce the Digital twin widget along with its essential functionalities. The digital twin can be defined as a KIS.MANAGER-based virtual counterpart of a KIS.Device, which is connected to the real one through dedicated WiFi. A graphical representation of both KIS.Device digital twins is presented in Fig. 2.24. Now, let us proceed to digital twin design. For that purpose it is necessary to go to Main menu → Assets and select the desired asset, e.g., KIS.BOX 1. Subsequently, the Edit dashboard button  should be used and then the Add widget button can be employed, resulting in the view portrayed in Fig. 2.23. Finally, the digital twin design boils down to selecting an appropriate widget and applying the resulting dashboard changes. This produces in the dashboard view presented in Fig. 2.25. The same procedure can be performed for any KIS.LIGHT, e.g., KIS.LIGHT 1. As a result, the digital twin presented in Fig. 2.26 is obtained. Irrespective of the KIS.Device being used, it can be noticed that the actual values of both digital inputs (GPIO 3, GPIO 4) and outputs (GPIO 1 and GPIO 2) are given as well. In particular, a 0 binary state is signified by Off while 1 is denoted by On. Moreover, their switching frequency, expressed in mHz, can be observed as well. As detailed in Sect. 2.1, the possible operational LED colors are limited to the ones provided in Table 2.2. As shown in Figs. 2.27–2.28, the digital twins allow changing the color of the operational LEDs by simply selecting the desired one and then pressing the Set button. Note also that a given operational LED may be flashing or blinking, which can be achieved via the Flashing checkbox. It should be also noted that the digital twins display the current state of the operational LEDs, which can evolve in various ways, e.g., due to the appropriate rules implemented within Rule engine (see Sect. 2.9).

Time drive

Let us perform a simple change of the KIS.BOX 1 state pertaining to its second button operational LED color (cf. Button 2 in Fig. 2.28). It can be realized as follows:

1. Select the Button 2 color as blue.
2. Press the Set button.

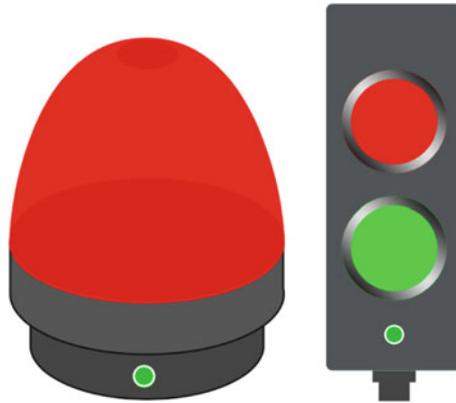


Fig. 2.24 KIS.LIGHT and KIS.BOX digital twins



Fig. 2.25 Digital twin of KIS.BOX 1



Fig. 2.26 Digital twin of KIS.LIGHT 1

- 3. Wait a moment.
- 4. Select the Button 2 color as black.

After this simple procedure, one can press the Start time drive button . This allows monitoring or reconstructing historical states of KIS.Devices, which can be realized according to Fig. 2.29.

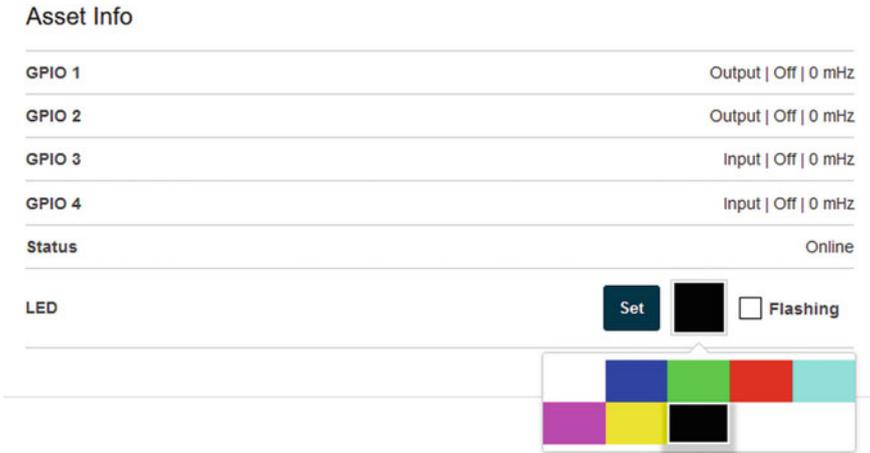


Fig. 2.27 Changing the KIS.LIGHT operational LED color

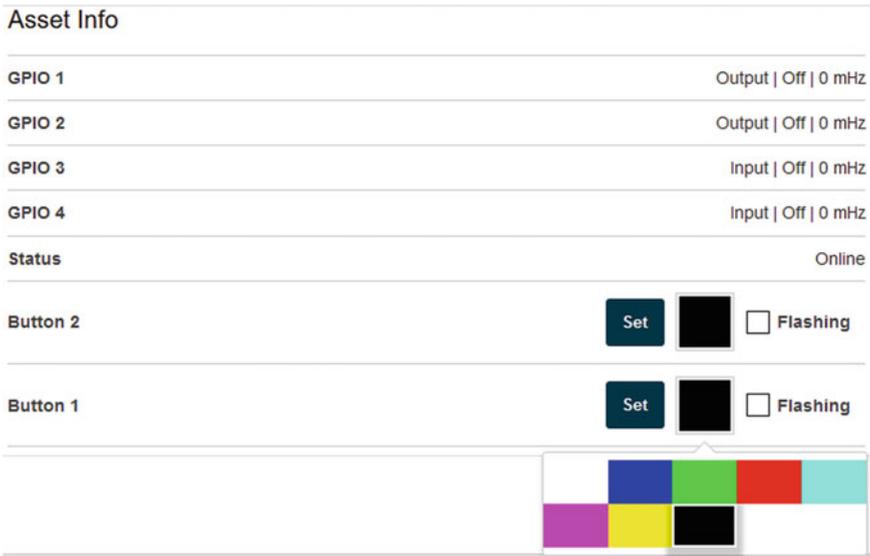


Fig. 2.28 Changing the KIS.BOX operational LED color

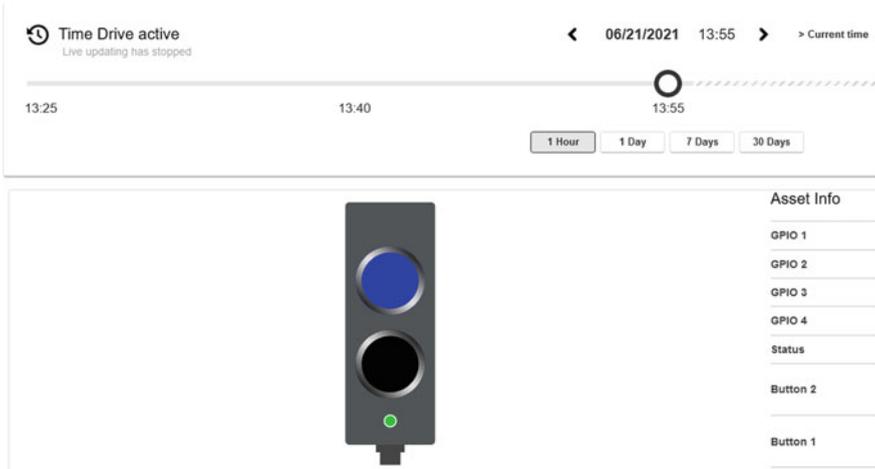


Fig. 2.29 KIS.BOX 1 time drive

The objective of the preceding sections was to introduce the reader into essential subjects related to asset and user management. The subsequent part aims at going into details pertaining to the description of the current asset state using the concept of Datapoints.

2.7 Datapoints: Plotting and Storing Data

Datapoints can be perceived as links between KIS.Devices and KIS.MANAGER. They can be easily accessed through Main Menu → Assets → KIS.Device → Datapoints, and can be of different types, which are listed in Table 2.6.

Table 2.6 Datapoint types

Type	Description
Boolean	A logical value, i.e., either <code>true</code> or <code>false</code>
Long	An integer value, e.g., 24
Double	A double precision floating point value, e.g., 3.14
Text	A character string, e.g., #0000FF

> Important

From the software engineering viewpoint, Datapoints can be perceived as read only variables, which can be further processed and analysed. The only restrictions are the following:

- Neither logical nor numerical operations on the Text type Datapoints are allowed, and hence they can only be stored or visualized.
- No numerical operations can be performed on the Boolean type Datapoints; however, this limitation can be easily tackled with the `IF []` command (see A.16).

Datapoints are processed in real time (limited by the data transfer rate), and hence their values depend on the current state of a KIS.Device. A full list of Datapoints along with their simple sample applications is provided in Appendix B. The evolution of Datapoints can be easily observed by going to Main menu → Assets → KIS.Device and then pressing the Datapoints button . As a result, the view presented in Fig. 2.30 is obtained. Subsequently, by selecting the desired Datapoints, their time evolution can be graphically observed in a dedicated plot. This process is illustrated in Fig. 2.31. A similar functionality can be directly obtained within the KIS.Device dashboard. Indeed, by proceeding to the dashboard, i.e., by pressing the Dashboard button  and then the Edit dashboard one , it is possible to add one of the widgets (Add widget button) presented in Fig. 2.23. Finally, to achieve the desired functionality, a Datapoint Chart is incorporated within the dashboard. Its configuration requires selecting a Datapoint (see Fig. 2.32), providing a headline of the figure as well as the plotting interval. After this preliminary setup, one can proceed to defining the plot options, which can be realized according to Fig. 2.33. Apart from these, one can set the plot color as well as define the axis properties (cf. Fig. 2.34):

Show axis: enable/disable an axis;

Scale axis: an axis may have a limited range. That can be time-varying, which can be realized by assigning a suitable Datapoint;

Show Min/Max: show minimum and maximum values of the data being plotted.

Apart from the above features, it is possible to define Thresholds over/below which the plot color will be changed (see Fig. 2.34).

> Multiple plots

The datapoint Chart widget allows presenting multiple plots, which can be individually managed using options and properties described in this section.

Data Points

Name ▲	Type
<input type="text" value="Filter..."/>	- All - ▼
 appVersion	 Datapoint
 bootloaderVersion	 Datapoint
 button1Color	 Datapoint

Fig. 2.30 KIS.BOX 1 datapoints

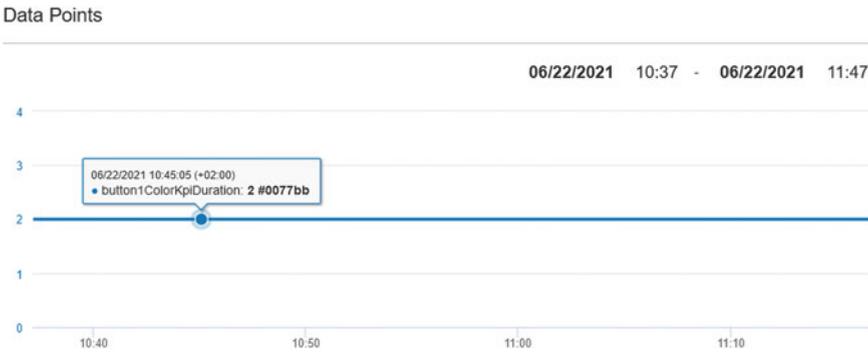


Fig. 2.31 KIS.BOX 1 datapoints' trend

Fig. 2.32 Datapoint chart: selecting a datapoint

Please select a Data point...

Name ▲	Type
<input type="text" value="Filter..."/>	- All - ▼
appVersion	 Datapoint
bootloaderVersion	 Datapoint
button1Color	 Datapoint
button1ColorKpi	 Datapoint
button1ColorKpiDuration	 Datapoint

Plot style	stairs	linear	shaded stairs	shaded linear
Point style	line	circle	square	triangle
Line style	solid	dotted	dash-dot	dashed

Fig. 2.33 Datapoint chart: plot options and their interpretation

button1ColorKpiDuration

Display

Show Axis

Scale Axis

Show Min/Max

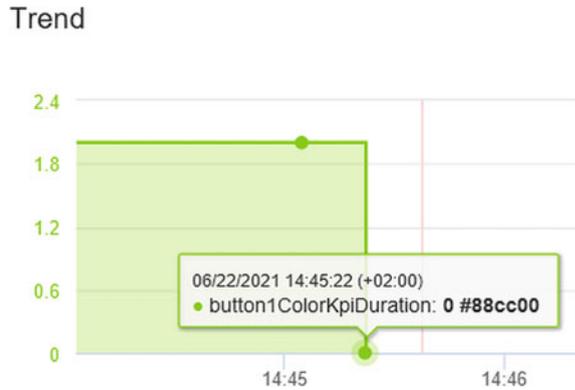
Start

Threshold 1 e.g. 70

Threshold 2 e.g. 70

Fig. 2.34 Datapoint chart: axis and colors

Fig. 2.35 Datapoint chart:
an example



Practical example

The illustrative example being considered aims at realizing the following steps:

1. Go to the KIS.BOX 1 dashboard.
2. Add a new Datapoint Chart widget.
3. Select the `button1ColorKpiDuration` Datapoint.
4. Set the following plot options:
 - Plot style: shaded stairs,
 - Point style: circle,
 - Lines style: solid.
5. Set the plotting interval to 10 min.
6. Save the dashboard with the new widget.
7. Observe the current value of the Datapoint and verify it with Table 2.2.
8. Wait a moment, change the KIS.BOX 1 operational LED color to blue, verify the current value of the Datapoint and compare it with Table 2.2.

The obtained results are shown in Fig. 2.35. Let us proceed to check the obtained results with Table 2.2. Initially, the KIS.BOX Button 1 operational LED color was black, and hence one can see the line at the level of 2. Similarly, after a moment of time, the color was set to blue, which corresponds to the 0-valued point.

➤ Storing and analysing data

As shown in Fig. 2.36, additional features of the Datapoint Chart are as follows :

Fig. 2.36 Datapoint chart: storing and analysing data



Download CSV: the data can be stored as a CSV file, with the first column being a time stamp and the remaining columns corresponding to the values of the associated Datapoints;

Analyze in DP-App: this feature moves the user to the Datapoint view like the one presented in Fig. 2.31.

2.8 Let Us Go to Workspaces: An Introductory Example with the Floorplan Widget

The objective of this section is to introduce a crucial feature of asset groups concerning the possibility of visualizing the system floorplan. It can be simply defined as a graphical representation of the workshop. The floorplan is virtually implemented within KIS.MANAGER using the Floorplan widget. To access it, it is necessary to go to Main menu → Asset Groups, which results in the view presented in Fig. 2.37. Subsequently, the desired asset group has to be selected, e.g., Workspace 1.

> Floorplan vs. assets

The floorplan can only contain the assets which are assigned to a given group. For a comprehensive description pertaining to asset management, the reader is referred to Sect. 2.4. Alternatively, it is possible to perform this task directly from a workspace by simply pushing the Assigned Assets button . Subsequently, the following steps should be realized:

Asset Group ▲	Definition
<u>My Devices</u>	Inventory
<u>Workspace 1</u>	Workspace
<u>Workspace 2</u>	Workspace
<u>Workspace 3</u>	Workspace
<u>Workspace 4</u>	Workspace
<u>Workspace 5</u>	Workspace

Fig. 2.37 Asset groups

1. Push the Edit assigned assets button .
2. Push the Add assets to this group button.
3. Select the desired assets and add them to the group.

Note that the process of removing assets from a group can be realized in an analogous way.

After selecting the workspace, e.g., Workspace 1, one can see the view presented in Fig. 2.38. Similarly as in Sect. 2.5, dashboards can be managed by pushing the Edit Dashboard button . After selecting this option, it is necessary to push the Add widget button, which results in the view presented in Fig. 2.39. However, the resulting set of widgets is different than the one described in Sect. 2.5 (see Fig. 2.23). Indeed, there are the following nine widgets:

- Floorplan,
- Datapoint Chart,
- Data Sheet,
- Aggregation,
- KPI Single Value,
- KPI Aggregated Chart,
- KPI Single Value Column,
- KPI Pie Chart,
- KPI Single Period Chart.

The first three widgets can be directly used for digitalization of asset groups as well as the monitoring and analysis of their behaviour. The remaining five require suitable preprocessing using KPIs, which are discussed in Sect. 4.1.2.

Let us start by selecting the Floorplan widget. This requires an appropriate graphical representation of the real floorplan in the form of an SVG file. Such a kind of

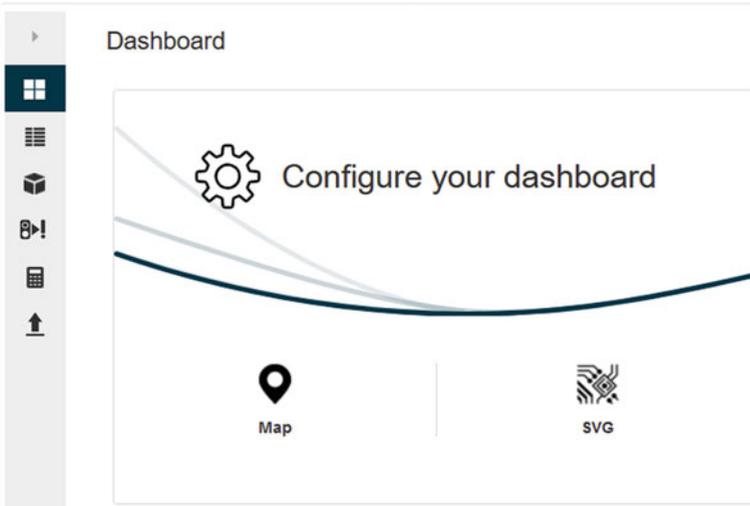


Fig. 2.38 Asset groups: workspace 1

Please choose a widget type

 Floorplan A group floorplan	 Data Point Chart	 Data Sheet
 Aggregation	 Kpi Single Value	 Kpi Aggregated Chart
 Kpi Single Value Column	 Kpi Pie Chart	 Kpi Single Period Chart

Fig. 2.39 Workspace: nine widgets

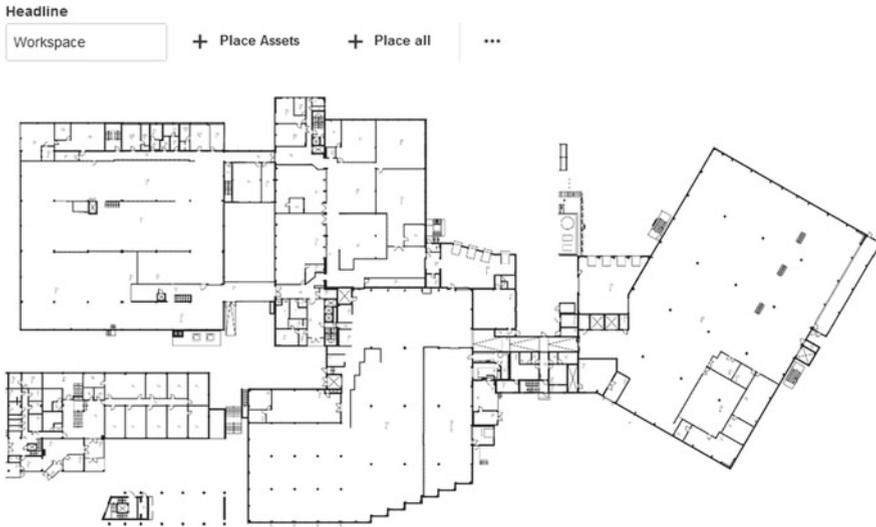


Fig. 2.40 Floorplan widget: an initial configuration

files employs a two-dimensional vector graphic format created by the World Wide Web Consortium. It expresses images with a text format that is based on XML. There are plenty of free and commercial tools which can be used for preparing a floormap using the SVG format. A good representative example is the freely available Inkscape package [1]. Having an SVG-based floorplan, it is possible to use the Floorplan widget. Its initial configuration reduces to providing a desired Headline and the above-mentioned SVG image. As a result, the view portrayed in Fig. 2.40 is obtained. Subsequently, either all or selected assets can be introduced within the floorplan. Let us proceed with selected assets. To perform this action, it is necessary to use the Add widget button (cf. Fig. 2.40). The desired assets can be added as depicted in Fig. 2.41. Finally, the assets (KIS.BOX 1 and KIS.LIGHT 0) can be freely located within the floorplan, which results in the dashboard presented in Fig. 2.42.

> Asset group time drive

Similarly as in Sect. 2.6, it is possible to monitor or reconstruct historical states of the asset group. This can be easily realized by pressing the Start time drive button  (cf. Fig. 2.38).

Change Settings

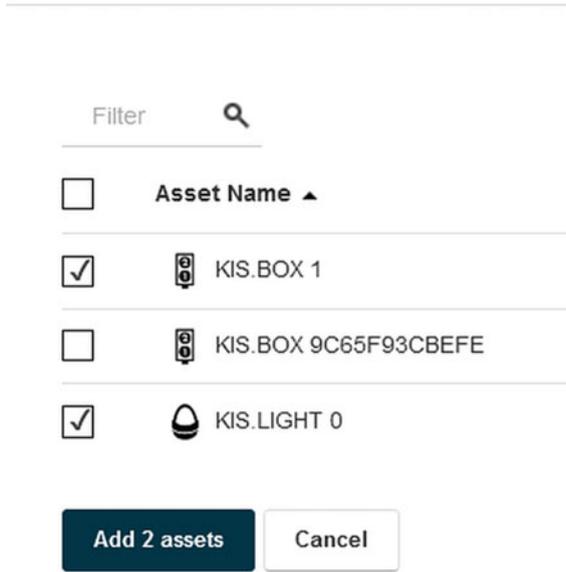


Fig. 2.41 Floorplan widget: adding assets

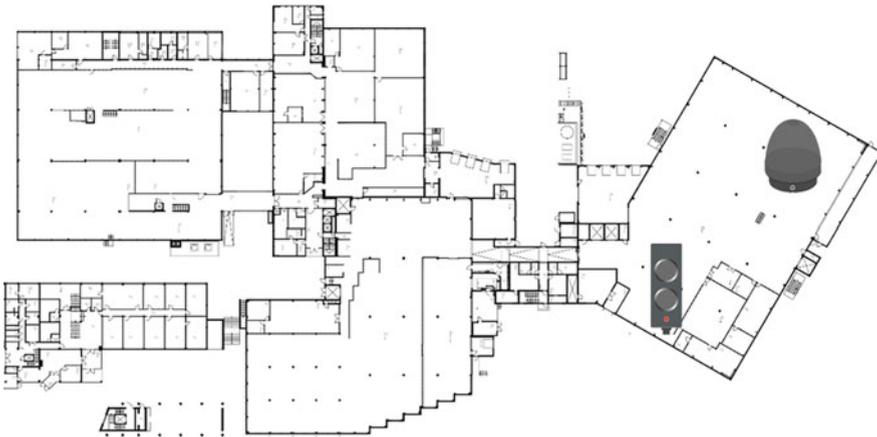


Fig. 2.42 Floorplan widget within the dashboard

2.9 Let Us Rule: Managing Rules Within a Workspace

A rule-based system can be perceived as a way of transforming a human expert's knowledge into an automated framework [2, 3]. For the purpose of KIS.ME, such a framework is called Rule engine. On the other hand, it can be seen as a KIS.MANAGER functionality, which makes it possible to implement IF-THEN rules governing interactions between assets. Thus, the rule-based system can be simply designed using a set of assets and a set of rules dictating their behaviour. In KIS.ME, rules are exemplified as a set of IF-THEN statements labelled with a unique name:

$$\text{rule name: IF antecedent THEN consequent.} \quad (2.1)$$

Generally, a rule may have multiple antecedents linked by or and/or or operators. Similarly, it may have multiple consequences. For example,

$$\begin{aligned} \text{rule 1: IF A is black and C is white or X is green THEN} \\ \text{A is white, B is black.} \end{aligned} \quad (2.2)$$

Note that the antecedent of a rule possesses two parts:

1. a linguistic object (LO),
2. the value of the linguistic object.

The linguistic object is linked with its value through various operators, e.g., is, or mathematical operators: \leq , $<$, etc. Since KIS.ME operates on KIS.Devices, LOs can be designed either with KIS.BOXes or KIS.LIGHTs, which yields the following:

KIS.Device | Status | Operating Mode: it expresses functional access of a KIS.Device to WiFi and it can be either Online or Offline;

KIS.Device | GPIO | GPIO No: it corresponds to the logical status of a selected GPIO, either On (High) or Off;

KIS.BOX | Button No | Button No Color: it expresses the color of the button operational LED, which can take a value from Table 2.2 with an additional Boolean-valued Flashing option;

KIS.LIGHT | LED | LED Color: it expresses the color of the operational LED, which can take a value from Table 2.2 with an additional Boolean-valued Flashing option.

There are two operators linking LOs with their values:

EQUAL: checks if an LO has a given value;

NOT: checks if an LO does not have a given value.

Subsequently, a consequent can be defined as an Action, which can assign a value to one of the above-listed LOs except for the first one, i.e., KIS.Device | Status | Operating Mode. Another restriction is that only digital outputs can be set, i.e., KIS.Device | GPIO | GPIO 1 and KIS.Device | GPIO | GPIO 2. The above actions

should be perceived as the ones acting on a device. An action can also be associated with sending a predefined notification e-mail.

> Notification templates

A predefined notification e-mail is based on a notification template. Such a template can be defined by a user with admin rights (cf. Table 2.5) by simply going to Main menu → Portal Admin → Notification Templates. As a result, by using the Create new notification template button **+**, a notification template editor is obtained, which is presented in Fig. 2.43. The crucial features of such a template are as follows:

- Name: uniquely identifies a template within Rule engine actions;
- Subject: stands for the title of a predefined e-mail;
- Message: constitutes the body text of the predefined e-mail.

Both Subject and Message can be conveniently designed using a set of variables, which can be accessed after pushing the Add variables... button. The meaning of the crucial parameters should be interpreted as follows:

- asset.name: the name of a KIS.Device,
- asset.properties.type: either sBox or sLight,
- event.key: EMAIL_ACTION,
- event.timestamp?datetime: the date and time of an event.

Before proceeding to designing a sample rule, a set of suitable definitions has to be provided:

- Conditions: the set of antecedents merged with and/or operators,
- Triggers: the set of antecedents merged with or operators,
- Actions: the set of consequents.

Under the above definitions, triggers can be perceived as necessary conditions for performing a Rule engine-based inference. Additionally, triggers can be formed with all of the above-defined logistic objects. However, they can also use a linguistic object associated with pressing the KIS.BOX button. Unlike conditions, triggers verify if the value of a linguistic object has given instants, e.g., a button is pressed. Thus, a full list of triggers for linguistic objects is formed by extending the above-defined one with what follows:

- KIS.BOX | Button No | Pressed: it expresses the fact of pressing the KIS.BOX button.

Apart from the above functionalities, triggers have also optional settings:

- after x times: the trigger is fired when a given value of a linguistic object has been counted x times;

← Create Template

Name *

Type * E-mail

Languages

▾ default

Subject *

+ Add variables...

Message *

Fig. 2.43 Notification templates

Trigger ⓘ

Optional Settings

ⓘ

[Trigger Details](#) ⓘ

Fig. 2.44 Sample trigger with optional settings

after x minutes: the trigger is fired after x minutes from the time when a given value of a linguistic object has been recorded;
after x hours: the trigger is fired after x hours from the time when a given value of a linguistic object has been recorded.

Figure 2.44 shows a sample trigger, which is fired after pressing the KIS.BOX button two times. It should be also pointed out that this kind of trigger has an internal counter, which is automatically reset after reaching a given threshold. It can be also reset manually after using the Trigger details link along with the Reset button (cf. Fig. 2.45). Finally, let us note that such a manual reset is not available for the after x minutes and after x hours optional settings.



Fig. 2.45 Trigger reset

Sample rule

The objective of this example is to define a rule which satisfies the following requirements:

Environment: It is defined with Workshop 1 as well as employs KIS.BOX 1 and KIS.LIGHT 0.

Triggers: The triggers are associated with pressing KIS.BOX 1 Button 1 or Button 2.

Conditions: The operational LED color of KIS.LIGHT 0 can be either black or green and its status should be online.

Actions: The associated actions are as follows:

- change the operational LED color of KIS.LIGHT 0 to green;
- send a notification email to `john.doe@controlintech.pl` with the subject and title “Color change” while the body of the message being KIS.Device color has changed.

Let us start with defining an email notification template by going to Main menu → Portal Admins → Notification Templates. As has already been discussed, such a template can be designed according to Fig. 2.46. Before proceeding to Rule engine definitions, it can be observed that the above conditions may have a visible effect if the KIS.LIGHT 0 operational LED color is either black or green. Thus, an appropriate initial condition has to be imposed by going to Main menu → Assets, selecting KIS.LIGHT 0 and using its digital twin to set an appropriate operational LED color (Sect. 2.6). Under the above preliminary setup, triggers, conditions and actions can be intuitively defined by pushing the Rule engine button (cf. Fig. 2.42). Subsequently, the Create rule button should be used to open the Rule engine editor and provide the required ingredients, i.e., the name of the rule, triggers, conditions and actions. As a result, the view presented in Fig. 2.47 is obtained. After saving the rule, it is activated and operates within KIS.ME.

Create Template

Name * Type * E-mail

Languages

▾ default

Subject *

Message *

Fig. 2.46 Notification template: “change color”

> Rule interactions

As can be expected, each asset group/workspace has its own set of rules. However, when sharing assets between workspaces, the users must be cautious about their possible unappealing interactions.

2.10 State-Space Modelling

The objective of this section is to introduce the concept of the system state, which is a set of variables that can be used to describe any past and future system behaviour. Consequently, the system state-space is a space of admissible state values. Subsequently, the state-space model is defined as a set of rules which enables cyclical transition between the consecutive states.

Name: Exemplary rule Active:

Trigger

	KIS.BOX 1	Button 1	Pressed	×
	KIS.BOX 1	Button 2	Pressed	×

OR
×
+

Conditions

	KIS.LIGHT 0	LED	LED color	EQUAL		<input type="checkbox"/> Flashing	×
	KIS.LIGHT 0	LED	LED color	EQUAL		<input type="checkbox"/> Flashing	×
	KIS.LIGHT 0	Status	Operating Mode	EQUAL	Online		×

OR
×
+

Actions

	KIS.BOX 1	Set LED	Button 1 Color		<input type="checkbox"/> Flashing	×
	john.doe@controlintech.pl	Change color				×

Fig. 2.47 Sample rule

Traffic lights state-space model

To illustrate the concept of the state-space model, let us employ a traffic lights example. In this case, the transition rules can be clearly visualized using Fig. 2.48. The objective of the remaining part of this example is to attain a similar functionality using KIS.ME. In particular, the following features should be achieved:

Environment: It is defined within Workshop 1 and employs KIS.BOX 1, hence the traffic lights system presented in Fig. 2.48 is reduced to one KIS.BOX, which changes the colors of its operational LEDs to mimic the behaviour of the traffic lights system.

Triggers: A trigger is associated with pressing KIS.BOX 1 Button 2.

Conditions: The conditions are simply defined by the current state, which is one of those presented in Table 2.7;

Actions: The associated action is simply a consecutive state.

It should be noted that each state described in Table 2.7 is uniquely defined, which makes it possible to form the state-space model using a set of four rules. A sample rule evolving the system from state 1 to state 2 is provided in Fig. 2.49. Moreover,



Fig. 2.48 Traffic lights

Table 2.7 KIS.BOX-based states

State	KB operational LED 1	KB operational LED 2
1	Red	Black
2	Red	Yellow
3	Green	Black
4	Green	Yellow

Trigger

 KIS.BOX 1 Button 2 Pressed

Conditions

 KIS.BOX 1 Button 2 Button 2 Color EQUAL Flashing

Actions

 KIS.BOX 1 Set LED Button 2 Color  Flashing

Fig. 2.49 Sample traffic lights rule

the initialization of the system requires that KIS.BOX 1 operational LEDs be in one of the quadruple of states defined in Table 2.7. This can be easily achieved using the KIS.BOX digital twin (see Sect. 2.6).

2.11 Mastering Rule Management: Completeness and Consistency

The objective of the two preceding sections was to provide a concise introduction into Rule engine design and the inference mechanism. However, for more complex systems, the number of rules will proliferate. Thus, it is customary to have a tool capable of checking their completeness and consistency. For that purpose, the celebrated decision table [2] is introduced. It operates on Boolean-valued conditions, and hence it is beneficial to recall the essential logical operators provided in their priority order:

- ¬ negation,
- ∧ conjunction,
- ∨ disjunction,
- ⇒ implication,
- ⇔ equivalence.

The behaviour of the above operators is explained in Fig. 2.50. It can be also observed that the implication and equivalence can be expressed by

$$a \Rightarrow b \text{ can be calculated with } \neg a \vee b;$$

$$a \Leftrightarrow b \text{ can be obtained with } (a \Rightarrow b) \wedge (b \Rightarrow a).$$

Therefore, a typical way of expressing a logical implication is IF a THEN b . Thus, according to the truth table for $a \Rightarrow b$, if a is false then it does not matter what b is, and hence the implication is true. Similarly, if a and b are true then the implication is true as well. The last case, i.e., when a is true and b is false, can be explained using the following example:

$$\text{IF } \sin(z) = 0 \text{ THEN } z = 0.$$

Such an implication is false as $z = 0$ is not the only value for which $\sin(z) = 0$. Thus, the implication which is true should be

$$\text{IF } \sin(z) = 0 \text{ THEN } z = k\pi, \text{ with } k \text{ being an integer value.}$$

To summarize these preliminaries, two crucial definitions have to be provided:

Tautology: a statement that is true for every possible interpretation, e.g., (KIS.Box|Status|Operating Mode is Offline) or (KIS.Box|Status|Operating Mode is Online);

Contradiction: a statement that is false for every possible interpretation, e.g., (KIS.Box|Status|Operating Mode is Offline) and (KIS.Box|Status|Operating Mode is Online).

Indeed, it is obvious that in the first case the statement is always true as the KIS.BOX status can be either Offline or Online. Contrarily, it is evident that the second state-

a	$\neg a$
0	1
1	0

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

a	b	$a \Rightarrow b$
0	0	1
0	1	1
1	0	0
1	1	1

a	b	$a \Leftrightarrow b$
0	0	1
0	1	0
1	0	0
1	1	1

Fig. 2.50 Truth tables

ment is always false as the KIS.BOX status cannot be Offline and Online simultaneously. Thus, it is evident that one should avoid both cases while designing rules with KIS.ME.

2.11.1 Transforming Conditions

A preliminary step for implementing a rule base is to collect all rules and check if it is possible to simplify them. For that purpose, a standard set of transformation strategies can be used:

- double negation: $\neg\neg a = a$,
- commutativity of conjunction: $a \wedge b = b \wedge a$,
- commutativity of disjunction: $a \vee b = b \vee a$,
- associativity of conjunction: $(a \wedge b) \wedge c = a \wedge (b \wedge c)$,
- associativity of disjunction: $(a \vee b) \vee c = a \vee (b \vee c)$,
- distributivity of conjunction: $(a \vee b) \wedge c = (a \wedge c) \vee (b \wedge c)$,
- distributivity of disjunction: $(a \wedge b) \vee c = (a \vee c) \wedge (b \vee c)$,
- idempotency of conjunction: $a \wedge a = a$,
- idempotency of disjunction: $a \vee a = a$,
- De Morgan's law: $\neg(a \wedge b) = \neg a \vee \neg b$,
- De Morgan's law: $\neg(a \vee b) = \neg a \wedge \neg b$,
- contraposition law: $a \rightarrow b = \neg b \Rightarrow \neg a$.

Having the above strategies, one can simply associate logical variables with antecedents (see Sect. 2.9). For that purpose, it is suggested to use the following nomenclature:

$$a := \text{LOs EQUAL Value,} \quad (2.3)$$

$$\neg a := \text{LOs NOT Value.} \quad (2.4)$$

Figure 2.51 presents two sample antecedents which can be associated with a logical variable a and its negation $\neg a$. Having such variables, it is easy to write and operate on conditions in a consistent way. Another important aspect pertains to an appropriate use of parentheses. Indeed, due to the operators' priority, $a \vee (b \wedge c)$ can

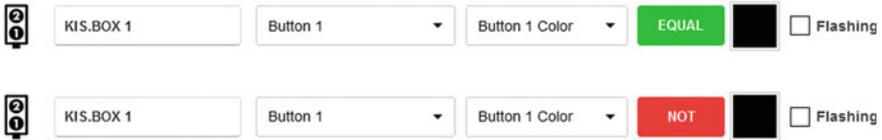


Fig. 2.51 Sample a (top) and $\neg a$ (bottom)

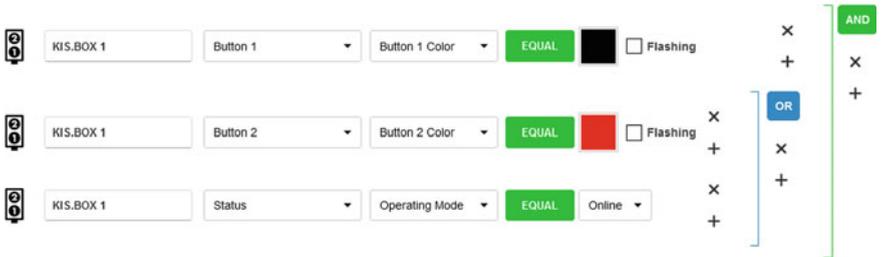


Fig. 2.52 Sample $a \wedge (b \vee c)$ (a : top, b : middle, c : bottom)

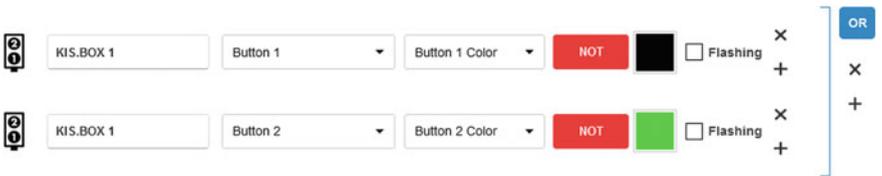


Fig. 2.53 Implementation of $\neg a \vee \neg b$

be simplified to $a \vee b \wedge c$. Contrarily, $a \wedge (b \vee c)$ differs from $a \wedge b \vee c$. This can be clearly observed within Rule engine, which uses different right square brackets for indicating appropriate priorities. Thus, $a \wedge (b \vee c)$ can be exemplified with the conditions portrayed in Fig. 2.52.

Sample simplification

Let us suppose that there are two logical variables defined as

$$a := \text{KIS.BOX 1} \mid \text{Button 1} \mid \text{Button 1 Color EQUAL black},$$

$$b := \text{KIS.BOX 1} \mid \text{Button 2} \mid \text{Button 2 Color EQUAL green},$$

and an action must be performed when $\neg(a \wedge b)$ (NAND operation) is true. Unfortunately, such a condition is impossible to implement in Rule engine. However, it is straightforward to observe that by applying De Morgan’s law it is possible to simplify it into $\neg a \vee \neg b$, which can be easily implemented in Rule engine (see Fig. 2.53).

```
(%i17) load(logic);
(%o17) C:/maxima-5.45.1/share/maxima/5.45.1/share/logic/logic.mac

(%i18) logic_simplify( not (a and b));
(%o18) nota or notb
```

Fig. 2.54 Logical expression simplification with Maxima

Table 2.8 Decision table

Condition/action	r_1	r_2	...	r_k
Condition 1	$c_{1,1}$	$c_{1,2}$...	$c_{1,k}$
Condition 2	$c_{2,1}$	$c_{2,2}$...	$c_{1,k}$
⋮	⋮	⋮	...	⋮
Condition n	$c_{n,1}$	$c_{n,2}$...	$c_{n,k}$
Action 1	$a_{1,1}$	$a_{1,2}$...	$a_{1,k}$
Action 2	$a_{2,1}$	$a_{2,2}$...	$a_{2,k}$
⋮	⋮	⋮	...	⋮
Action m	$a_{m,1}$	$a_{m,2}$...	$a_{m,k}$

> Automatic simplification

There are several free and commercial packages which can be used for automatic simplification of logical expressions. Maple [4] (package `Logic`, command `BooleanSimplify`) and Maxima [5] (package `logic`, command `logic_simplify`) are good representative examples of commercial and freely available tools, respectively. Let us employ Maxima for the purpose of simplifying the expression used in the preceding example, i.e., $\neg(a \wedge b)$. The Maxima session implementing this task is presented in Fig. 2.54.

To conclude this section, it should be pointed out that the rule

$$\text{IF antecedent 1 OR antecedent 2 OR } \dots \text{ antecedent } n \text{ THEN Action(s)} \quad (2.5)$$

can be replaced by a set of n rules of the form

$$\text{IF antecedent 1 THEN Action(s),} \quad (2.6)$$

⋮

$$\text{IF antecedent } n \text{ THEN Action(s).} \quad (2.7)$$

2.11.2 Decision Tables

The classical decision table [2, 3, 6, 7] is used to express k rules of the form

$$r_i : \text{ IF Condition 1 and Condition 2, \dots and Condition } n \\ \text{ THEN Action 1 and Action 2, \dots and Action } m, \quad i = 1, \dots, k, \quad (2.8)$$

which can be presented using Table 2.8. The internal horizontal line between conditions is perceived as the and conjunction. Additionally, the double horizontal line separates conditions and actions while the vertical ones distinguish the rules. Thus, any rule r_i can be easily reconstructed from Table 2.8 to (2.8) by reading the column corresponding to r_i in a top-to-bottom order. The entries of the decision table with respect to the conditions, i.e., $c_{i,j}$, in Table 2.8 are as follows:

- T: if the condition must hold;
- F: if the condition does not hold;
- : if the condition is ignored;

while for the actions $a_{i,j}$ we have

- X: if the action has to be executed;
- : if the action has not to be executed.

A set of rules or a decision table have the following important features:

Redundancy: If there is a situation in which conditions of two rules with the same actions hold, then they are called redundant ones.

Inconsistency: If there is a situation in which conditions of two rules with different actions hold, then they are called inconsistent ones.

Completeness: For every situation there is a rule whose conditions will be satisfied.

Checking redundancy and inconsistency

Let us consider three conditions which have to be implemented using Rule engine within Workshop 1 with KIS.BOX 1 and KIS.LIGHT 0:

Condition 1: KIS.BOX 1 | Button 1 | Button 1 Color is black;

Condition 2: KIS.BOX 1 | Button 2 | Button 2 Color is blue;

Condition 3: KIS.LIGHT 0 | LED Color | Color is green.

There are also two actions:

Action 1: KIS.LIGHT 0 | Set LED | LED Color | is black;

Action 2: KIS.LIGHT 0 | Set LED | LED Color | is blue.

Let us suppose that the above conditions and actions were used to implement three rules, r_1 , r_2 and r_3 , expressed in the form of the decision table detailed in Table 2.9. Let us consider a situation in which Condition 1 is T (true) while Condition 2 and

Table 2.9 Decision table with redundancy and inconsistency

Condition/Action	r_1	r_2	r_3
Condition 1	T	–	T
Condition 2	–	F	T
Condition 3	F	F	–
Action 1	X	X	–
Action 2	–	–	X

Table 2.10 Decision table with inconsistency

Condition/Action	r_1^1	r_3
Condition 1	–	T
Condition 2	–	T
Condition 3	F	–
Action 1	X	–
Action 2	–	X

Condition 3 are F (false). In such a case both rules r_1 and r_2 are active. Since they have identical actions (Action 1), they are redundant, which means that they can be merged into one equivalent rule r_1^1 . The resulting decision table is presented in Table 2.10. Let us consider a situation in which Condition 1 and Condition 2 are T (true) while Condition 3 is F (false). It can be easily observed that rules r_1^1 and r_3 are inconsistent because their condition sets are satisfied while they have different sets of actions. The inconsistent rules denote the situation in which different things may happen under the same circumstances. Indeed, two contradictory actions will be initiated:

Action 1: KIS.LIGHT 0 | Set LED | LED Color | is black;
 Action 2: KIS.LIGHT 0 | Set LED | LED Color | is blue.

To summarize, a simple rule reduction principle can be stated as below.

➤ Rule reduction principle

If there are two rules l and s with the same actions and identical condition entries $c_{i,l}$ and $c_{i,s}$ except for $c_{j,l} \neq c_{j,s}$, then they are replaced with a single new rule f with a condition entry $c_{j,f}$ equal to “–”. Note that as “–” represents T/F, it should be perceived as equal to both T and F.

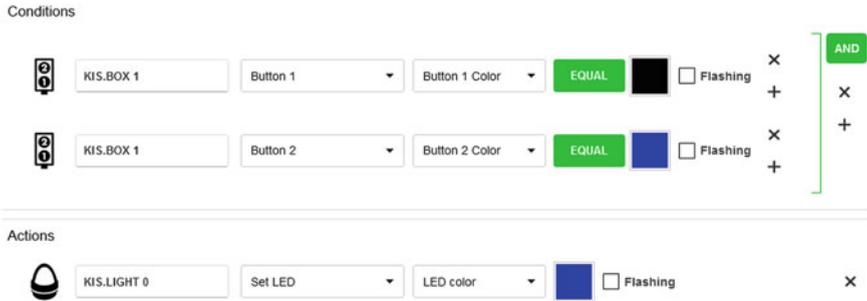


Fig. 2.55 Rule r_3 implemented within Rule engine

> **Implementing decision tables with Rule engine**

Implementation of decision tables within Rule engine can be easily realized using the following procedure:

1. Select the i -th rule.
2. Read the rule in order from top-to-bottom and perform the following translation:
 - if an entry is equal to “T”, then introduce the condition using (2.3);
 - if an entry is equal to “F”, then introduce the condition using (2.4);
 - if an entry is equal to “–”, then ignore it.

As an example, let us consider rule r_3 in Table 2.10, which is exemplified in Fig. 2.55.

The above implementation strategy is applicable to a set of simple conditions in the form of either (2.3) or (2.4). However, it can be easily extended to more advanced structures. On the other hand, the logical expressions can be simplified using the strategies proposed in the preceding section (see, e.g., (2.5) and (2.7)) or transformed into conjunctive normal form (see, e.g., [2] for a comprehensive explanation).

> **Rule base completeness**

Having a way of checking the redundancy and consistency, it is possible to provide a strategy for verifying the completeness of a set of rules. Since each condition in a decision table (Table 2.8) is a Boolean-valued one, this simply means that the complete number of rules is equal to

$$k = 2^n, \tag{2.9}$$

where k is the total number of rules while n is the number of conditions (cf. (2.8)). Thus, a rule should be defined for every possible situation. As an example, let us consider the decision table presented in Table 2.10. Thus, for three conditions one can easily see that (2.9) implies that there should be $k = 8$ rules. Contrarily, there are two rules in Table 2.10. However, due to the use of “–”, rule r_1^1 may have four alternative forms:

$$r_1^1 \in \left\{ \begin{bmatrix} F \\ F \\ F \end{bmatrix}, \begin{bmatrix} T \\ T \\ F \end{bmatrix}, \begin{bmatrix} F \\ T \\ F \end{bmatrix}, \begin{bmatrix} T \\ F \\ F \end{bmatrix} \right\},$$

while rule r_2 has two alternative ones:

$$r_2 \in \left\{ \begin{bmatrix} T \\ T \\ F \end{bmatrix}, \begin{bmatrix} T \\ T \\ T \end{bmatrix} \right\}.$$

This clearly means that there are six rules and the decision table (Table 2.10), and hence the set of rules is incomplete. Thus, it is possible to verify (2.9) with

$$k_r = \sum_{i=1}^{n_r} 2^{n_{i,-}}, \quad (2.10)$$

where n_r stands for the number of rules in Table 2.8 while $n_{i,-}$ is the number of instances of “–” in the i -th rule, $i = 1, \dots, n_r$. In the example presented in Table 2.8, one can easily identify what follows

- there are two rules $n_r = 2$;
- there are two instances of “–” in rule r_1^1 , which results in $n_{1,-} = 2$;
- there is one “–” in rule r_2 , i.e., $n_{2,-} = 1$;
- thus, (2.10) implies that $k_r = 2^2 + 2^1 = 6$, which clearly means that $k_r < k$ and the set of rules is incomplete.

It should be pointed out that there are situations in which rule base completeness is not necessary. Indeed, if it is guaranteed that not all possible situations pertain to input variables, and hence, conditions are possible, then the number of rules can be smaller. Such a situation may occur during the state-space modelling presented in Sect. 2.10. In such a case, the consecutive states are cyclically realized, which implies appropriate rule order execution. Finally, it should be pointed out that triggers may also have influence on the final number of rules. Indeed, they cause that some rules are not fired for a given trigger setting.

2.12 Case Study: Trend Plotting and Performance Analysis

The objective of the preceding three sections was to introduce Rule engine, which makes it possible to bring a given system alive according to a predefined set of rules. Having such features, it is possible to monitor KIS.Device performance with Datapoints and the associated widgets described in Sect. 2.7. For that purpose, let us reconsider the traffic lights example presented in Sect. 2.10. Let us start with transforming Table 2.7 into a decision table assuming that the initial state of the system is (cf. Sect. 2.6)

```
KIS.BOX 1 | Button 1 |Button 1 Color is red;
KIS.BOX 1 | Button 2 |Button 2 Color is black.
```

By observing an obvious condition stating that an operational LED cannot have two different colors simultaneously, the resulting decision table is given in Table 2.11. This implies that there is no need for implementing “F”-valued entries, which simplifies the Rule engine structure. For `Button 1 Color`, T is equivalent to red while F stands for green. Similarly, for `Button 2 Color`, T is equivalent to black while F stands for yellow. It is straightforward to observed that there are two conditions, and hence (2.9) implies that a complete set of rules should have four rules. This is exactly the case. Moreover, the rules are consistent because each of them pertains to a different set of actions. Finally, a Rule engine-based implementation of r_1 – r_4 is provided in Figs. 2.56, 2.57, 2.58 and 2.59. Having a fully functional system, it is possible to design a dashboard containing

- a floorplan within Workshop 1 (cf. Sect. 2.8 for guidelines),
- a digital twin of KIS.BOX 1 implementing the traffic lights system,
- a Datapoint Chart widget containing two (cf. Sect. 2.7 for details) plots showing the color of KIS.BOX first and second operational LEDs. This can be achieved using

```
button1ColorKpiDuration,
button2ColorKpiDuration.
```

The resulting dashboard is presented in Fig. 2.60. Such a design allows intuitive visualization and analysis of the behaviour of the traffic lights system. Indeed, the plots in Fig. 2.61 correspond to the operational LED colors of the first (green) and the second (yellow) buttons. As can be observed, both of them have two possible values only, which can be recorded and analysed over a specified time period. In particular, the green line switches between the red (5) and the green (3) level. Similarly, the yellow line switches between black (2) and yellow (7).

Trigger

 KIS.BOX 1 Button 2 Pressed

Conditions

 KIS.BOX 1 Button 2 Button 2 Color EQUAL  Flashing

Actions

 KIS.BOX 1 Set LED Button 2 Color  Flashing

Fig. 2.56 Traffic lights rule r_1 implemented within rule engine

Trigger

 KIS.BOX 1 Button 2 Pressed

Conditions

 KIS.BOX 1 Button 2 Button 2 Color NOT  Flashing

Actions

 KIS.BOX 1 Set LED Button 2 Color  Flashing

Fig. 2.57 Traffic lights rule r_2 implemented within rule engine

Conditions

 KIS.BOX 1 Button 1 Button 1 Color EQUAL  Flashing

 KIS.BOX 1 Button 2 Button 2 Color NOT  Flashing

Actions

 KIS.BOX 1 Set LED Button 1 Color  Flashing

Fig. 2.58 Traffic lights rule r_3 implemented within rule engine

Conditions

	KIS.BOX 1	Button 1	Button 1 Color	NOT		<input type="checkbox"/> Flashing
	KIS.BOX 1	Button 2	Button 2 Color	NOT		<input type="checkbox"/> Flashing

Actions

	KIS.BOX 1	Set LED	Button 1 Color		<input type="checkbox"/> Flashing
--	-----------	---------	----------------	--	-----------------------------------

Fig. 2.59 Traffic lights rule r_4 implemented within rule engine

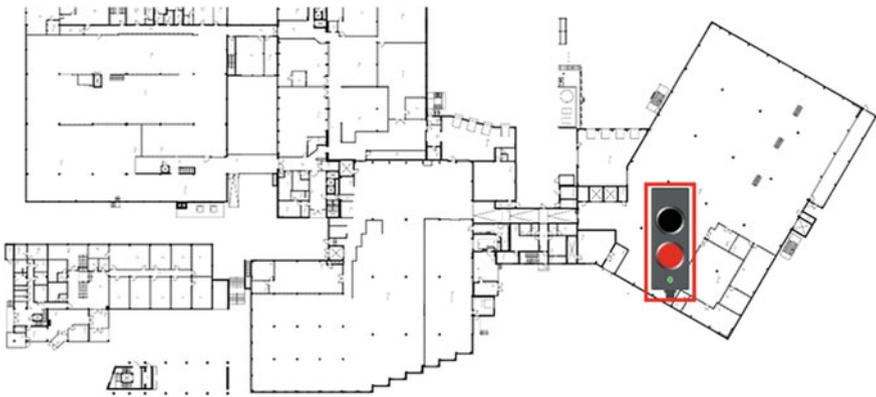


Fig. 2.60 Traffic lights dashboard

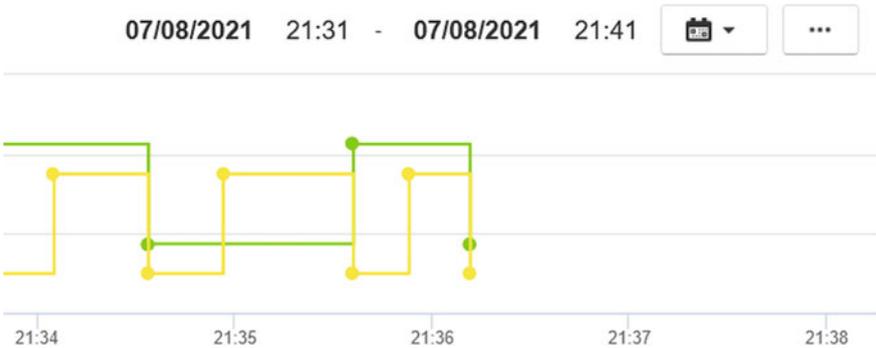


Fig. 2.61 Traffic lights datapoint chart

Table 2.11 Decision table for traffic lights

Condition/Action	r_1	r_2	r_3	r_4
KIS.BOX 1 Button 1 Button 1 Color	–	–	T	F
KIS.BOX 1 Button 2 Button 2 Color	T	F	F	F
KIS.BOX 1 Set LED Button 1 color is red	–	–	–	X
KIS.BOX 1 Set LED Button 1 color is green	–	–	X	–
KIS.BOX 1 Set LED Button 2 color is black	–	X	–	–
KIS.BOX 1 Set LED Button 2 color is yellow	X	–	–	–

2.13 Training Exercises

The objective of this section is to provide a set of practical exercises which can be used for validating the knowledge and skills gathered in this chapter. It is assumed that the user realizing these exercises has appropriate rights and permissions. This can be easily verified using Table 2.5.

2.1 Asset onboarding and management

1. Check the availability/feasibility of the following points:
 - a KIS.Device,
 - an M12-to-USB cable (see Table 2.4),
 - a computer/tablet equipped with a web browser and a USB port,
 - a company account with admin user rights (see Sect. 2.3),
 - WLAN access with permission credentials.
2. Go to <https://kismanager.rafi.de> and log in with user company credentials.
3. Perform the onboarding procedure of KIS.Device according to the guidelines presented in Sect. 2.1.
4. Depending on the KIS.Device type, change the name of the device to either KIS.BOX YOUR NAME or KIS.LIGHT YOUR NAME (see Sect. 2.4, Fig. 2.18).
5. Assign the KIS.Device to Workshop 1 and Workshop 2.

2.2 Creating a new user

Using an arbitrary e-mail address at your disposal, perform the following:

1. Create a user according to the guidelines presented in Sect. 2.3.1.
2. Give the new user installer rights (see Table 2.3).
3. Assign the new user to Workspace 1 and Workspace 2.

2.3 Dashboard and digital twin design

Exercise requirements: The exercise requires access to one KIS.BOX and one KIS.LIGHT. Proceed according to the following tasks:

1. Go to Main menu → Assets.
2. Select KIS.BOX, proceed to its dashboard and modify its name to “Sample dashboard”.
3. Design a KIS.BOX digital twin.
4. Add Datapoint Chart to the dashboard capable of displaying Datapoints:

```
button1ColorKpiDuration,  
button2ColorKpiDuration.
```

5. Using the digital twin, change arbitrarily the colors of both the first and the second operational LEDs.
6. Record the behaviour in the Datapoint Chart and store it with a CSV file.
7. Open the CSV file in a MS Excel-like software and compare its content with Table 2.2.
8. Use the Time drive feature of the dashboard and observe KIS.BOX historical behaviour.
9. Repeat, analogously, points 1–8 with KIS.LIGHT.

2.4 Floorplan widget

Exercise requirements: The exercise requires access to one KIS.BOX and one KIS.LIGHT assigned to a selected workspace.

1. Prepare your own floorplan using vector graphic software (e.g., Inkscape [1]) and save it as an SVG file.
2. Add the Floorplan widget to Workspace and locate KIS.LIGHT and KIS.BOX on it.

2.5 KIS.LIGHT ruling

Exercise requirements: completed Exc. 2.4.

1. Write a rule called `rule b2r`:

```
Triggers: KIS.LIGHT operational LED is black;  
Conditions: KIS.LIGHT operational LED is black;  
Actions: KIS.LIGHT operational LED is red.
```

2. Write a rule called `rule_r2b`:

Triggers: KIS.LIGHT operational LED is red;

Conditions: KIS.LIGHT operational LED is red;

Actions: KIS.LIGHT operational LED is black.

3. Go to Main menu → Assets, select KIS.LIGHT.
4. Using the KIS.LIGHT digital twin, set its operational LED color to black.
5. Add a Datapoint Chart associated with KIS.LIGHT to the Workspace dashboard, and attach its plot to `led1ColorKpiDuration`.
6. Fill in the rest of Datapoint Chart configuration parameters in an arbitrary way.
7. Using the Zoom In/Out feature, observe the behaviour of the switching signal within the last two minute.
8. What can you say about the switching frequency/period? Is it uniform?

2.6 KIS.LIGHT ruling continued

Exercise requirements: completed Exc. 2.5. Using a similar strategy like in Exc. 2.5, perform the following:

1. Write a set of rules enabling KIS.LIGHT transition with consecutive states described in Table 2.2, i.e., Blue, Turquoise, Black, Green, Magenta, Red, White, Yellow.
2. Using the Zoom In/Out feature, observe the behaviour of the switching signal within the last two minutes.
3. What can you say about the transition periods?

2.7 KIS.BOX traffic lights

Exercise requirements: The exercise requires access to one KIS.BOX assigned to a selected workspace.

1. Read and analyse the traffic light case study described in Sect. 2.12.
2. Build your own traffic light system according to the strategy described in Sect. 2.12.
3. Analyse the transition periods. What can you say about them?

2.8 Rule simplification I

1. Analyse the rule transformation and simplification strategies presented in Sect. 2.11.1; in particular, automatic simplification using the Maxima software, which is presented in Fig. 2.54.
2. Install the freely available Maxima [5] software.
3. Repeat the automatic simplification process presented in Fig. 2.54.
4. Let a be a logical variable denoting the fact that KIS.BOX is online. Thus, $\neg a$ signifies the fact that it is offline:
 - What can you say about $a \wedge (\neg a)$?
 - What can you say about $a \vee (\neg a)$?

2.9 Rule simplification II

Exercise requirements: The exercise requires access to one KIS.BOX and KIS.LIGHT assigned to a selected workspace.

1. Let us define the following logical variables:

$$\begin{aligned} a &:= \text{KIS.BOX} \mid \text{Button 1} \mid \text{Button 1 Color EQUAL red,} \\ b &:= \text{KIS.BOX} \mid \text{Button 2} \mid \text{Button 2 Color EQUAL black,} \\ c &:= \text{KIS.LIGHT} \mid \text{LED} \mid \text{LED Color EQUAL green.} \end{aligned}$$

2. Implement a rule with the following condition:

$$(a \wedge b \wedge c) \vee (\neg a) \vee (a \wedge (\neg b) \wedge c), \quad (2.11)$$

taking as a trigger the pressing of the first KIS.BOX button event along with an action:

Action: KIS.LIGHT 0 | Set LED | LED Color | is red.

Hint: Use Maxima to simplify the above logical expression.

3. What can you say about the usage of variable b ?

2.10 Decision tables

Exercise requirements: The exercise requires access to one KIS.BOX and KIS.LIGHT assigned to a selected workspace.

1. Let us define three conditions:

Condition 1: KIS.BOX | Button 1 | Button 1 Color EQUAL red,
 Condition 2: KIS.BOX | Button 2 | Button 2 Color EQUAL black,
 Condition 3: KIS.LIGHT | LED | LED Color EQUAL green;

along with two actions:

Action 1: KIS.LIGHT 0 | Set LED | LED Color | is red;
 Action 2: KIS.BOX | Set LED | Button 1 color | is blue;
 and a trigger associated with pressing the first KIS.BOX button event.

2. The rules involving the above conditions and actions were initially developed and described using the decision table given in Table 2.12.
3. Simplify the decision table into a new one with four rules only.
4. Check completeness of the obtained decision table.
5. Implement the obtained decision table with Rule engine.

2.11 Battery assembly system

Exercise requirements: The exercise requires access to one KIS.BOX assigned to a selected workspace.

Table 2.12 Initial decision table

Condition/ Action	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
Condition 1	T	F	T	F	T	F	T	F
Condition 2	T	T	F	T	F	F	T	F
Condition 3	T	T	F	F	T	F	F	T
Action 1	–	X	–	–	–	X	–	X
Action 2	X	–	X	–	X	X	X	–

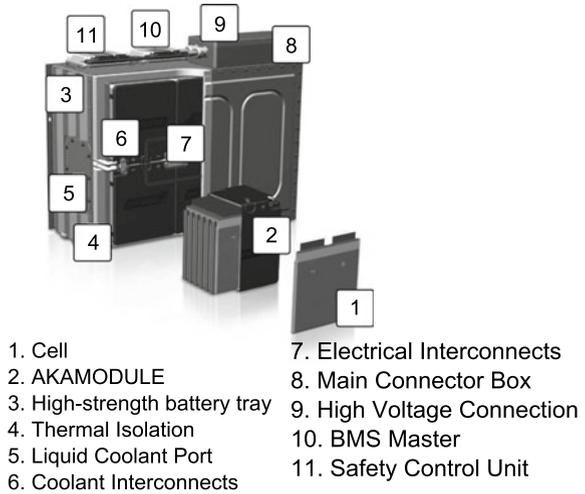
Table 2.13 KIS.BOX states

State	KB Button 1 color	KB Button 2 color	Action
1	Blue	Blue	Can start
2	Red	Blue	Cell mounting
3	Green	Blue	Cell mounting completed
4	Green	Red	Cell-controller linking
5	Green	Green	Mounting completed

1. Let us consider the battery system presented in Fig. 2.62. Such a system undergoes manual assembly, which, in simplified form, can be described by two tasks:
 - a. cell mounting,
 - b. controller linking.

Battery mounting should be realized according to the cyclically repeated states described in Table 2.13. Note that the initial state should be State 1.
2. Prepare and simplify a decision table pertaining to the states described in Table 2.13.
3. Check completeness and consistency of the obtained decision table.
4. Form a digital twin of KIS.BOX as well as an arbitrary floorplan containing selected parts of Fig. 2.62 integrated with KIS.BOX.
5. Use the obtained decision table and implement it with Rule engine.
6. With the KIS.BOX digital twin, set the initial KIS.BOX state to State 1 and check if the system performs correctly (cf. Table 2.13).
7. Add the Datapoint Chart widget to the Workspace dashboard, which makes it possible to analyse the colors of the operational LEDs of both the first and the second KIS.BOX button.
8. analyse both the historical and the current behaviour of the system. A hint: You can also use the time drive feature of the dashboard.

Fig. 2.62 Battery assembly



2.14 Concluding Remarks

The objective of this chapter was to provide a self-contained introduction to the KIS.ME IoT platform. In particular, preliminary information about both the hardware and software layers was introduced along with suitable operating procedures for accessing KIS.MANAGER and onboarding the hardware layer. Subsequently, a hierarchical KIS.ME structure was presented, which associates assets, users and workspaces along with suitable rights and permissions. For that purpose, a systematic set of guidelines concerning users, assets and workspace management was provided. Such essential knowledge enabled introduction of dashboards and widgets, which form the basis for the KIS.ME HMI interface. Concerning the widgets, particular attention was focused on hardware digital twins as well as floorplans exemplifying real life systems with an integrated set of assets. Such a couple was further extended with the Datapoints chart enabling graphical visualization of system performance. The rest of the chapter was devoted to system management using Rule engine. In particular, it started with a concise introduction to the graphical rule building structure. Subsequently, a state-space modelling strategy was proposed, which guarantees cyclical behaviour of the system. Finally, more advanced techniques for managing a set of rules were introduced, which allow their simplification as well as completeness and consistency verification. The chapter was concluded with a series of practical exercises, which certify the knowledge provided within it.

References

1. Inkscape. <https://inkscape.org/>. Accessed 25 June 2021
2. A. Ligeza, *Logical Foundations for Rule-Based Systems* (Springer, Berlin, 2006)
3. C. Grosan, A. Abraham, Rule-based expert systems, in *Intelligent Systems* (Springer, Berlin, 2011), pp.149–185
4. Maple. <https://www.maplesoft.com/>. Accessed 15 July 2021
5. Maxima. <https://maxima.sourceforge.io/>. Accessed 15 July 2021
6. F. Alsolami, M. Azad, I. Chikalov, M. Moshkov, *Decision and Inhibitory Trees and Rules for Decision Tables with Many-valued Decisions* (Springer, Berlin, 2019)
7. J.R. Metzner, B.H. Barnes, *Decision Table Languages and Systems* (Academic Press, New York, 2014)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

