



ICT-56-2020 - Next Generation Internet of Things

# D 4.5

## Final report on wireless communication infrastructure

Document information	
<b>Contract number</b>	957197
<b>Project website</b>	<a href="http://www.vedliot.eu">www.vedliot.eu</a>
<b>Dissemination Level</b>	Public
<b>Nature</b>	Report
<b>Contractual Deadline</b>	31.01.2024
<b>Author</b>	Erik Funke (CHR)
<b>Contributors</b>	Micha vor dem Berge (CHR), Gunnar Billung-Meyer (CHR)
<b>Reviewers</b>	Pascal Felber (UniNE), Karol Gugala (ANT)
<b>The VEDLIoT project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.</b>	

Changelog

<b>v0.1</b>	2021-11-20	Initial draft
<b>v0.2</b>	2021-11-27	Document structure refined
<b>v0.3</b>	2022-04-18	Included input from partners
<b>v0.4</b>	2022-04-20	Content finalized
<b>v0.5</b>	2022-04-21	Ready for internal review
<b>v0.6</b>	2022-04-25	Incorporated change requests from the reviewers
<b>v1.0</b>	2022-04-29	Final version of D4.2
<b>V1.1</b>	2023-11-24	Initial version of D4.5
<b>V1.2</b>	2023-12-06	Extended document structure
<b>V1.3</b>	2023-12-15	Updated existing specifications
<b>V1.4</b>	2023-12-21	Content Finalized
<b>V1.5</b>	2024-01-08	Ready for internal review
<b>V1.6</b>	2024-01-17	Incorporated change requests from the reviewers
<b>V2.0</b>	2024-01-22	Final version of D4.5

## Table of contents

1 Introduction	5
2 Evaluation of Wireless Technologies	7
3 Wi-Fi Integration into the Secure IoT Gateway	9
3.1 Extension of the API	10
3.2 Extension of Frontend	10
4 LoRaWAN Integration	11
4.1 Secure IoT Gateway as LoRaWAN Infrastructure Provider	12
4.1.1 Overview of the Communication Infrastructure	12
4.1.2 Hardware Selection and Evaluation	14
4.1.3 API Extension	14
4.1.4 Extension of Network Cockpit Frontend	15
4.2 u.RECS as LoRaWAN End Device	17
4.2.1 Overview of the Communication Infrastructure	17
4.2.2 Chip Selection and Integration	18
4.2.3 API Definition	18
5 Secure IoT Gateway Improvements	20
5.1 Change of Backend API	20
5.2 Refactoring of the Network Cockpit Frontend	22
5.3 System Hardening and TRL Increase	25
6 Conclusion	27
7 References	28

## Executive Summary

The deliverable, “Final report on wireless communication infrastructure”, reports the final state of the evaluation, testing and integration of the wireless technologies Wi-Fi and LoRaWAN into the u.RECS and Secure IoT Gateway.

New requirements regarding improved stability and the technological advancements of the used frameworks of the Secure IoT Gateway required code refactoring and infrastructure improvement. We set our initial focus onto the hardening and refactoring of the codebase, besides evaluating and specifying wireless integration scenarios and technologies inside this Task. Building upon these improved foundations, this report showcases the completion of the successful LoRaWAN and Wi-Fi integration, also documenting the steps taken towards hardening the Secure IoT Gateway and further advancing it towards a tangible product. With the knowledge gained through over a yearlong deployment in production environments at customer sites, and the collaboration with Siemens Electric Motor Condition Classification Use-Case within VEDLIoT, we were able to further expand the feature-set of the Secure IoT Gateway. We improved firewall and routing management, which greatly improved the usability and adaptiveness of the system in real deployment scenarios.

In addition to that, the integration of the u.RECS as a LoRaWAN device has been successfully accomplished on a hardware and software level. It allows its usage coupled with Secure IoT Gateway LoRaWAN infrastructure or other LoRaWAN stacks like The Things Network.

## 1 Introduction

This deliverable is based on the first report D4.2 [1] and documents the development work of Task 4.5, starting with the evaluation process and the resulting development steps taken for integrating wireless technologies (LoRaWAN, Wi-Fi) into the u.RECS hardware platform and the Secure IoT Gateway.

The general aim of the Secure IoT Gateway is securing network-based communication to counteract the current lack of network security multiple IoT devices offer. Generally, the work is partly based on work performed within the LEGaTO [2] project, which aimed at wired Ethernet-based communication. The final project report documenting the initial development of the Secure IoT Gateway can be found in the LEGaTO use case deliverable [3]. When it comes to developing IoT devices, security and sufficient encryption of data traffic is often overlooked, due to hardware limitations of embedded devices [4]. It is also common for IoT devices to send usage statistics or similar information to the manufacturer for analysis. If privacy about device usage is highly valued, this may be of concern. The listed problems led to the development of the Secure IoT Gateway - an easy-to-use VPN solution that encapsulates IoT device traffic, shields the network from sniffing unencrypted data and restricts unwanted communication to the manufacturer's servers. Besides securing the network, it also provides convenient features for secure communication among network devices over the Internet. Wireless communication is essential in IoT infrastructures, as sensors often have to be placed in inaccessible or inadequately connected places. Thus, the integration of wireless technologies into the u.RECS hardware platform and the Secure IoT Gateway is one of our goals within VEDLIoT. The Secure IoT Gateway aims to provide a wireless infrastructure, whilst the u.RECS acts as a wireless end device. Figure 1 shows an example scenario where communication is secured by the Secure IoT Gateway, with the u.RECS as a LoRaWAN end-device.

To enable VPN connections between IoT devices, we supply a variety of so-called "IoT Bridges". These are small ARM-based embedded devices with dual Ethernet ports to which IoT devices connect. From there on, the communication will be secured via VPN. Besides the IoT Bridges, we have an optional infrastructure component "Local Gateway" — hosting VPN servers necessary for securing connections from IoT Bridges directly inside the local customer network. VPN endpoints are also provided by the Cluster Gateway, allowing remote connections secured with VPN accessible from the Internet. This also enables the central routing of multiple Local Gateways from different company sites for a specific customer, as well as direct connections from IoT Bridges. IoT Bridges and Local/Cluster Gateways are controlled by a multi-client capable web application called "Network Cockpit", deployed in the cloud. This allows for easy management of the solution's components from a customer's perspective.

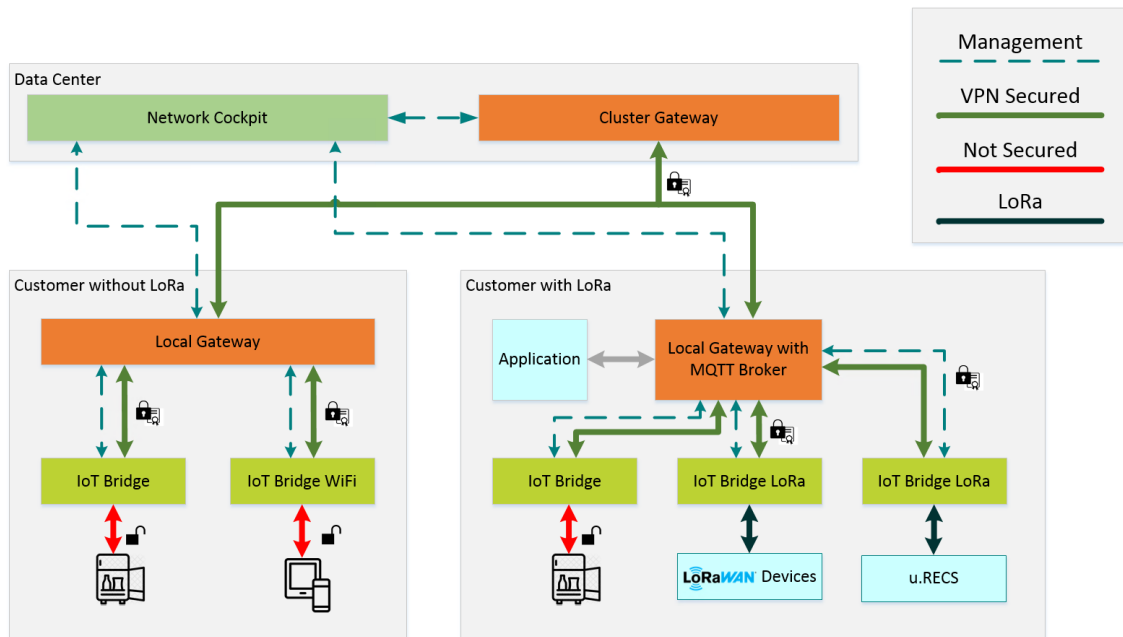


Figure 1: Communication infrastructure and component overview of the Secure IoT Gateway

The diverse layout of networks and their respective requirements for different throughputs, power availability and connection methods for different hosts, led to the line-up of IoT Bridge models showcased in Figure 2. Three Ethernet-based IoT Bridge models are currently available, each with its own use case — determined by the maximum network throughput and power consumption. Figure 2 shows the IoT Bridge hardware line-up of the Secure IoT Gateway with the newly added IoT Bridge LoRa. The integration work done in this Task also led to the support for Wi-Fi in the IoT Bridge 10 and IoT Bridge 50 models.





IoT Bridge name and picture				
	IoT Bridge 10	IoT Bridge 50	IoT Bridge 100	IoT Bridge LoRa
SoC	Atheros AR9331 400MHz	Allwinner H3 4 x 1,2 GHz	Rockchip RK3328 4 x 1,3 Ghz	Atheros AR9330 400Mhz
Power Consumption	< 1.5W	< 10W	< 10W	< 10W
VPN Speed	9 Mbit/s	55 Mbit/s	95 Mbit/s	untested
WiFi   LoRa	✓   ✗	✓   ✗	✗   ✗	✗   ✓
Integrated Flash	✓	✓	✗	✓
Price	~ 45€	~ 60€	~ 80€	~ 200€

Figure 2: IoT Bridge hardware line-up of the Secure IoT Gateway

## 2 Evaluation of Wireless Technologies

This chapter describes the evaluation process of wireless technologies suitable for integration into the u.RECS and Secure IoT Gateway. The proposal lists LoRa [5] and 5G [6] as possible wireless technologies, but other options were also evaluated. 5G provides high bandwidth and high-range transmissions. However, integrating cellular 5G into the Secure IoT Gateway product was discarded, due to the high cost and approval time regarding licensing private usage in the EU. Because the Secure IoT Gateway operates as a wireless infrastructure provider, licensing our own 5G gateways would be inevitable.

The u.RECS already contains Wi-Fi and Bluetooth capabilities provided by the BMC, which are optimal for high bandwidth, low range applications, as shown in Figure 3. To complement the existing wireless technologies, we settled for LPWAN (Low Power Wide Area Network) in the u.RECS. Wireless technologies with unlicensed frequencies are preferred because the Secure IoT Gateway supplies private base stations/gateways for the chosen technology. SigFox [7] and LoRa operate in unlicensed frequencies, which makes them possible candidates for integration. The downside of SigFox comes from the lack of private network deployment. SigFox is a public network, which relies on the usage of already existing infrastructure. We settled for LoRa as integration candidate regarding the Secure IoT Gateway and the u.RECS. It complements the existing wireless technologies (Wi-Fi and Bluetooth) of the u.RECS by providing long-range, low-bandwidth transmissions, as shown in Figure 3. To ensure a good balance between range and bandwidth, the Secure IoT Gateway implements Wi-Fi and LoRa. The Secure IoT Gateway IoT Bridge line-up already contains Wi-Fi-capable hardware, thus making the decision easy. Besides the technical benefits of the chosen standards, LoRa and Wi-Fi are popular technologies in the IoT sector. This increases the number of potential customers for the Secure IoT Gateway, due to the high compatibility.

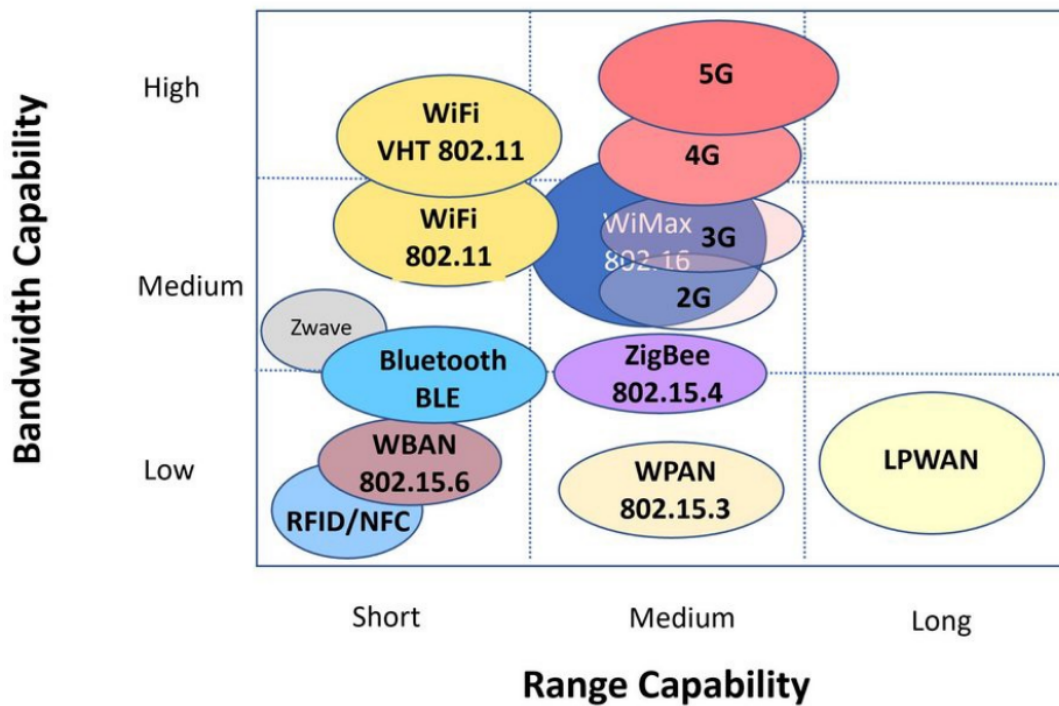


Figure 3: Range to bandwidth comparison of wireless technologies, © UnitingDigital.com

When it comes to LoRa, the integration was achieved by using the LoRaWAN [8] standard. LoRaWAN stands for "Long Range Wide Area Network" and classifies as an LPWAN. It is built on top of the physical LoRa standard and provides the Medium Access Control (MAC) layer to form a star topology network. LoRaWAN is suitable for low-bandwidth, high-range applications while consuming the smallest amount of power in the LPWAN sector. The physical LoRa communication can achieve transmission ranges up to multiple kilometres, but this is only the case in optimal external conditions. The bandwidth varies from 0.3 to 11 kbit/s, depending on the SF (Spreading Factor). Further technical details regarding LoRaWAN and Wi-Fi will be presented in the upcoming integration Chapters 3 and 4.



### 3 Wi-Fi Integration into the Secure IoT Gateway

Figure 4 shows the point of use for Wi-Fi technology within the Secure IoT Gateway. We are using the IoT Bridge 10 and IoT Bridge 50 as Wi-Fi access points because they already have the required hardware integrated. An IoT Bridge acts as an access point for one or more IoT devices and encrypts the traffic from there on.

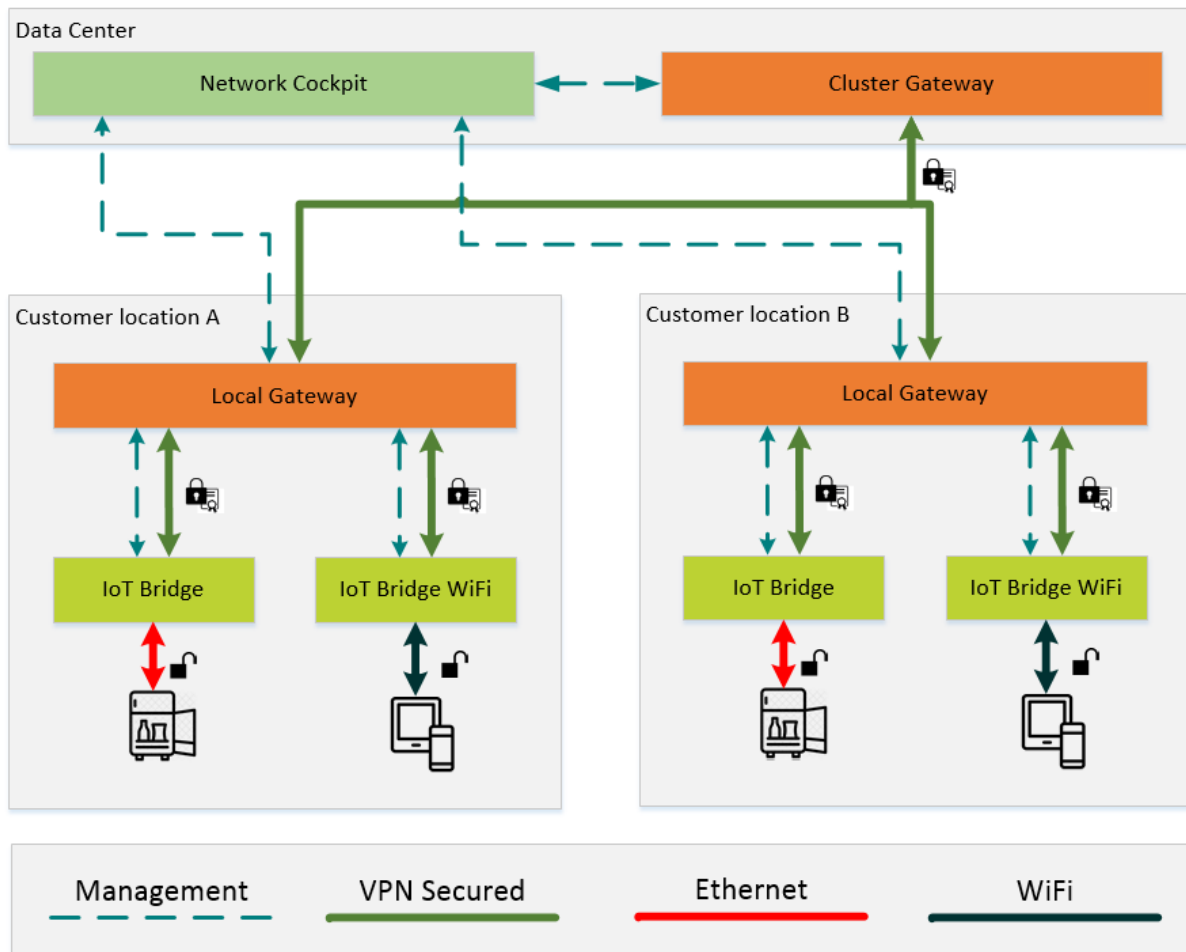


Figure 4: Communication infrastructure of the Secure IoT Gateway with Wi-Fi

As shown in Figure 4, IoT Bridges with Wi-Fi functionality behave as access points for IoT devices. The outbound connection to the Local Gateway is still established over Ethernet. The OpenWRT [9] operating system runs on all IoT Bridge platforms in our line-up, including the newly added IoT Bridge LoRa, which will be relevant in Chapter 4 (LoRaWAN Integration). To configure the access point on an IoT Bridge, the current IoT Bridge API of the Network Cockpit was adjusted accordingly. Further details are described in Section 3.1. Besides altering the API on the IoT Bridges to configure OpenWRT to accept and manage Wi-Fi configurations, the frontend and backend of the Secure IoT Gateway's Network Cockpit had to be extended. These changes are described in further detail in the upcoming sections.

### 3.1 Extension of the API

OpenWRT — the IoT Bridge operating system — provides a UCI (Unified Configuration Interface) [10], which stores the system’s configuration as key/value pairs. Due to the renewed backend API documented in the Section 5.1, the API of the IoT Bridges had to be adjusted. The new backend changed the request structure of the REST API, making communication between the components more efficient. By implementing the configuration queue system described in the Section 5.1, the API of the IoT Bridges does not impact the performance of the system anymore. In the previous API revision, the complete stored configuration was fetched every time a component requested the REST API of the Network Cockpit. This approach impacted the performance of the system, due to the continuous key/value comparison between the locally stored UCI data and the newly fetched configuration.

UCI allows the necessary Wi-Fi-related key/value pairs to be set. Thus, the API interfacing UCI can control the needed configurations remotely from the Network Cockpit. In order to apply a configuration related to a Wi-Fi interface, the API detects the changes and reloads the adapter automatically upon configuration. Development of the locally deployed controller software for the IoT Bridges was written in Lua, due to its lightweight nature and strong support within OpenWRT systems. We developed an OpenWRT package that can be installed with Opkg [10] on the IoT Bridges, which contains all Secure-IoT-Gateway-related functionalities. Besides installing the Secure IoT Gateway package via repository management, we also build OpenWRT from source on our Jenkins server, thus creating a flashable image file with the latest changes and versions.

### 3.2 Extension of Frontend

For managing Wi-Fi related configurations of an IoT Bridge, a new section inside the settings window was added. This will only show up when the selected component is marked as a Wi-Fi capable device. As of right now, this would include the IoT Bridge 10, 50 and LoRa. The tab displays the necessary fields for access point configuration to the customer. Wi-Fi is disabled per default, but the customer can change that by accessing the slider shown in Figure 5. When enabled for the first time, the default SSID will be the IoT Bridge name. The PSK is generated randomly, but we recommend entering a new PSK created by the customer. The “Apply” button passes the settings through backend user-input checks and stores the inputs in the configuration queue for the selected device. Device settings are later retrieved by the configured IoT Bridge in fixed intervals. The custom controller software for OpenWRT, as described in Section 3.1, handles the change of configuration on the hardware component.

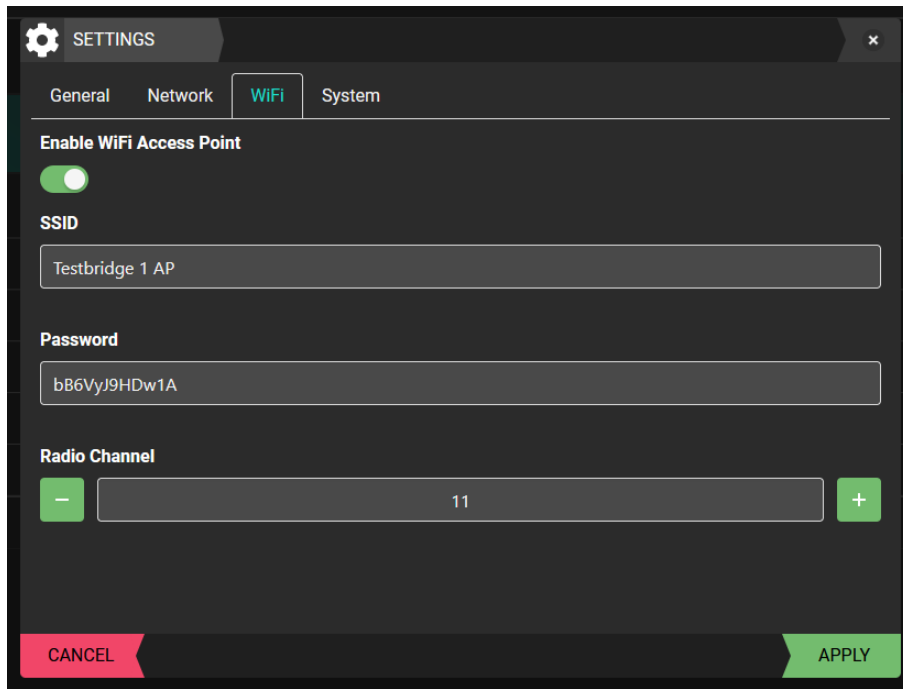


Figure 5: Screenshot of the Network Cockpit wireless settings popup

## 4 LoRaWAN Integration

This chapter describes the integration process of LoRaWAN into the Secure IoT Gateway and u.RECS. As previously described in Chapter 2, LoRaWAN is the MAC layer on top of the physical LoRa standard. With the help of LoRaWAN gateways, a star topology network is formed. LoRaWAN end devices are able to connect to the network by utilizing OTAA (Over The Air Activation) or ABP (Activation By Personalization) [11]. This will be relevant in the upcoming LoRaWAN integration Section 4.1.1 of the Secure IoT Gateway.

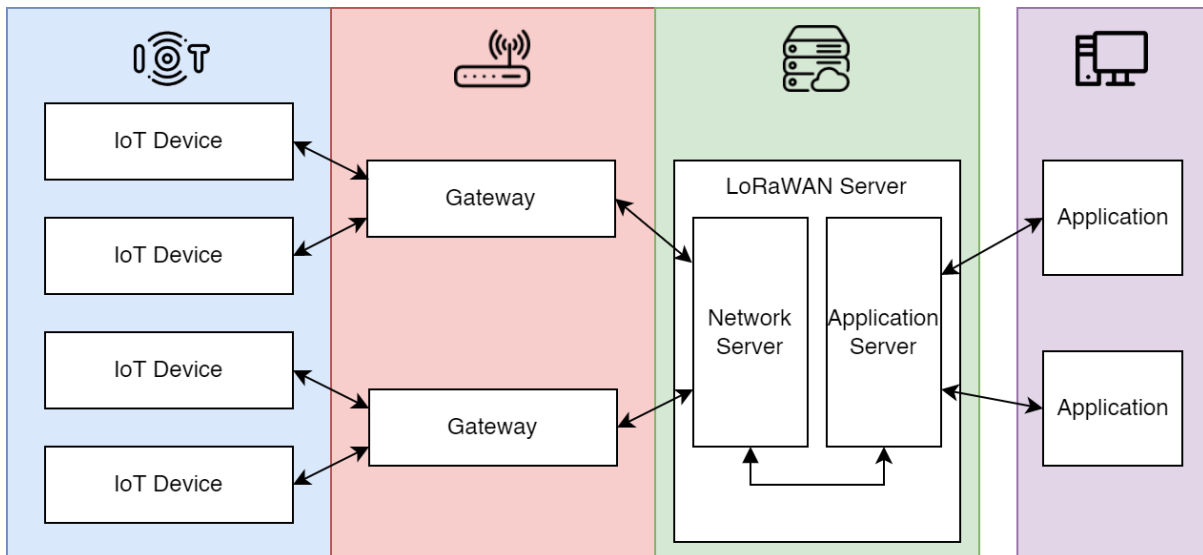


Figure 6: Default LoRaWAN architecture

The LoRaWAN standard specifies the usage of network servers, which interconnect the deployed LoRaWAN gateways. Application servers are used to allow the incoming and outgoing LoRaWAN data from the IoT devices and applications as shown in Figure 6. LoRaWAN uses two 128-bit AES encryption layers on the payload transmitted in the network. This ensures end-to-end encryption from the IoT device to the application server

and vice versa, making LoRaWAN a secure wireless technology. OTAA and ABP define the key handling necessary for creating the AES layers between the network components.

The u.RECS acts as a LoRa end device, whilst the Secure IoT Gateway provides the LoRaWAN communication infrastructure. Because the technology was used in two independent ways throughout the documented task, the chapter is split into the different integration scenarios of u.RECS and Secure IoT Gateway.

#### 4.1 Secure IoT Gateway as LoRaWAN Infrastructure Provider

The Secure IoT Gateway is capable of providing a LoRaWAN infrastructure, which can be used by various IoT devices. Different scenarios for implementing LoRaWAN into the existing structure of the Secure IoT Gateway were evaluated, with the conclusion of discarding the use of a LoRaWAN server. This is further described in the following section 4.1.1. For supplying the infrastructure, the use of a new local component is necessary. Section 4.1.2 shows the evaluation and testing of LoRaWAN gateway hardware, resulting in the addition of the IoT Bridge LoRa product line-up. The necessary steps for the LoRaWAN integration are documented in Sections 4.1.3 and 4.1.4.

##### 4.1.1 Overview of the Communication Infrastructure

We decided to discard the usage of LoRaWAN servers, thus terminating the LoRaWAN network at the point of the IoT Bridge Lora. This was necessary because evaluation showed it was not possible to run LoRaWAN server stacks in parallel to the OPNsense operating system due to incompatibility issues. This is further described in the evaluation Section 4.1.2. All LoRaWAN traffic is terminated and forwarded using MQTT (Message Queuing Telemetry Transport) [12] at the newly introduced IoT Bridge LoRa, which makes the use of a separate LoRaWAN server obsolete. By using MQTT forwarding on the IoT Bridge LoRa, all functionalities except OTAA are still given. MQTT is a messaging protocol commonly used in the IoT sector due to its low bandwidth requirements and M2M (Machine-to-Machine) topology. ABP is used as a joining protocol for LoRaWAN end devices due to the LoRaWAN stack termination on the IoT Bridge LoRa.

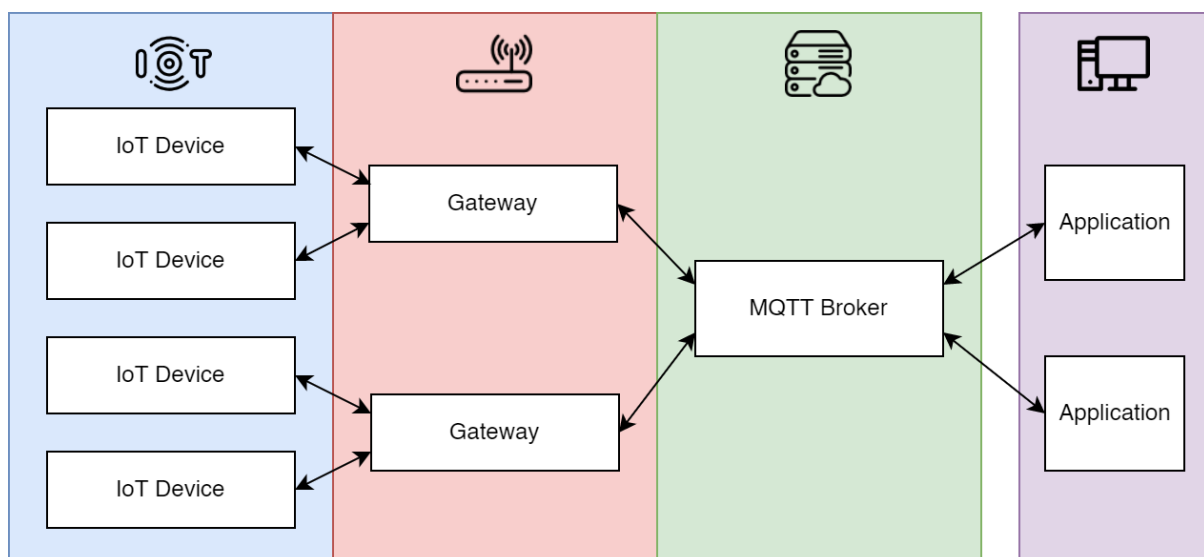


Figure 7: LoRaWAN architecture of the Secure IoT Gateway

Figure 6 showed the possible integration scenario with the LoRaWAN stack, contrary to the architecture we settled on, as shown in Figure 7. A centralized MQTT Broker handles the communication and allows access to data via the publish/subscribe principle. As written before, we decided on MQTT forwarding because evaluation showed that it is not possible

to reliably run a LoRaWAN server on the Local Gateway in parallel to the OPNsense [13] operating system. With MQTT forwarding, customers can register LoRaWAN ABP end devices inside the newly created LoRa management interface of the network cockpit as explained in Section 5.2. The system behaves like a normal LoRaWAN implementation, the only exception being the missing OTAA functionalities. Each registered end device has its own MQTT topic, which can be accessed by the applications. The MQTT broker is provided by the Local Gateway and can later be accessed by the applications using the LoRaWAN device data.

To register devices inside the network, ABP credentials can be set inside the LoRaWAN management interface inside the Network Cockpit for the specified gateways. ABP credentials include the AES keys used for encrypting the payload of the LoRaWAN transmissions. In order to configure the LoRaWAN-related settings, the existing API on the IoT Bridges was extended. It is now capable of performing LoRaWAN specific configurations, necessary for ABP decryption and MQTT forwarding. For every new LoRaWAN device, the corresponding ABP credentials (Device Address, Network Session Key, Application Session Key) and an MQTT topic need to be set in the configuration.

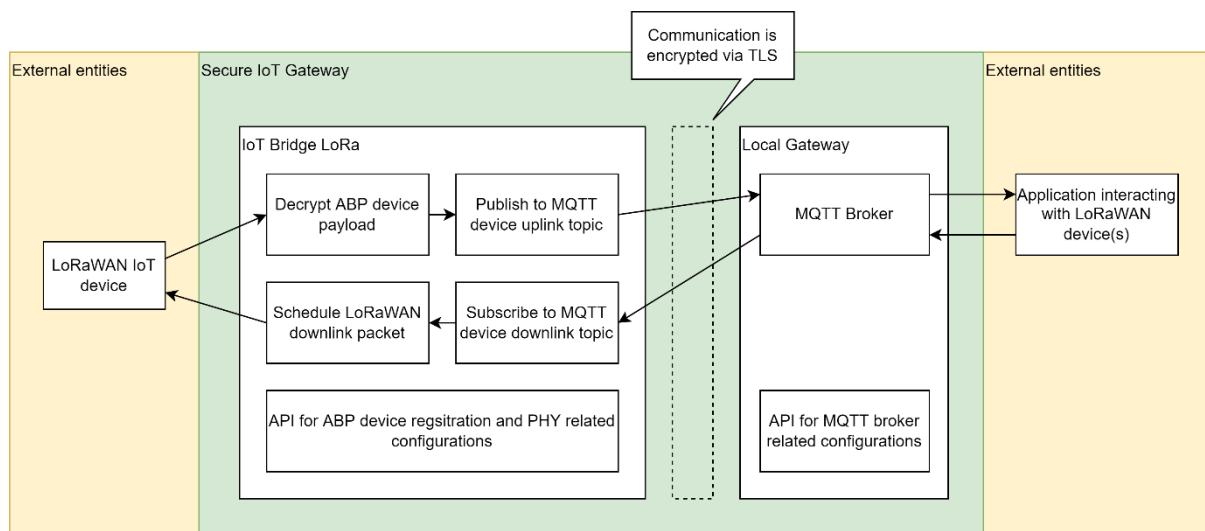


Figure 8: Detailed view of the LoRaWAN components and their interaction

Figure 8 shows the complete structure of the MQTT forwarding method in the Secure IoT Gateway environment. The IoT Bridge LoRa decrypts/encrypts the payload transmitted over LoRaWAN from the IoT devices. The decrypted payload is sent over to the MQTT Broker deployed on the Local Gateway over a TLS secured MQTT connection, which ensures the encrypted communication. In the previous deliverable [1] we specified a VPN encryption between IoT Bridge LoRa and MQTT Broker, but the implementation turned out to be redundant, and impacted the performance of the LoRa gateway, thus settling with the standard TLS implementation that MQTT offers. Applications can interface with the MQTT broker and schedule uplink packets or collect the downlink packets from the IoT devices registered to the IoT Bridge LoRa. Every IoT device has its own MQTT topic, which allows the incoming data to be separated for each IoT device. To ensure configuration of the newly introduced technologies, the API of the Secure IoT Gateway components had to be extended. This is further described in Section 4.1.3.

### 4.1.2 Hardware Selection and Evaluation

In order to use the LoRaWAN technology in the scope of the Secure IoT Gateway, the IoT Bridge LoRa was added to the hardware line-up. It consists of a Dragino LPS8 [14], an open source LoRaWAN Gateway which runs a customized version of the OpenWRT operating system. This was crucial for integration into the existing project architecture because the product line-up already contains OpenWRT-based devices. This ensured a smooth integration into the existing VPN and API structure used by the Secure IoT Gateway. The IoT Bridge LoRa contains a Semtech SX1308 baseband chip and two Semtech SX1257 transceiver modules, which allow for multi-channel LoRaWAN transmissions. The hardware was first tested with The Things Network (TTN) [15] and multiple Dragino LHT65 [16] temperature sensors. TTN is an open-source LoRaWAN server solution, which created the largest public LoRaWAN network as of now. The Secure IoT Gateway aims to provide a standalone LoRaWAN solution, which requires a local implementation of the LoRaWAN infrastructure. This includes the implementation of a LoRaWAN network and application server. Open-source solutions like ChirpStack [17] or TTN provide the necessary software for handling the LoRaWAN server infrastructure. Both solutions were tested on the Local Gateway running the OPNsense operating system with limited success. It is impossible to run the LoRaWAN server stack parallel to the OPNsense system without cutting down on crucial OPNsense functionalities. This was due to dependency incompatibilities (the most severe being OpenSSL) with the current FreeBSD version on which the OPNsense system is based. Later the decision was made to discard the usage of LoRaWAN servers, which led to evaluating the MQTT forwarding method as already described in Section 4.1.1.

### 4.1.3 API Extension

To allow LoRaWAN-related configurations to be made by the Network Cockpit, the API had to be extended. This was done on top of the existing API structure, which had been improved in the scope of this project, as described in the Section 5.1. The API on the OpenWRT-based IoT Bridges was extended to handle the configuration of the necessary LoRaWAN-related settings if deployed on the IoT Bridge LoRa. Besides handling LoRa-related configurations, a local log-parsing and statistics collection application was developed for the IoT Bridge LoRa. It gathers relevant information about LoRa PHY statistics like SNR, RSSI, packet count and RX/TX packets of specific LoRaWAN devices. The collected statistics data is sent securely over HTTPS to the REST API of the Network Cockpit, and later visualized in the frontend of the Network Cockpit as described in Section 4.1.4. Development of the control and statistics application for the LoRa Bridge was done in Lua as well, due to its lightweight nature and strong support for OpenWRT, which also runs on the IoT Bridge LoRa. In conclusion, the IoT Bridge LoRa controller software was built on top of the optimized API structure described in Section 5.1. It was adjusted to support the LoRa-specific management parameters and statistics, which are later controllable via the Network Cockpit frontend. The API counterpart on the Network Cockpit side was also expanded to handle LoRaWAN-related configuration management and statistics.

#### 4.1.4 Extension of Network Cockpit Frontend

To configure and monitor the newly introduced components that allow the LoRaWAN infrastructure of the Secure IoT Gateway, new UI elements had to be integrated into the Network Cockpit. These were added to the new UI structure described in Section 5.2. The Network Cockpit handles the LoRaWAN end-device management and the IoT Bridge LoRa-related configurations separately. The customer can configure the IoT Bridge LoRa in the Device Manager, where all Secure IoT Gateway components are shown. LoRaWAN end devices are handled separately on a new page because they are not part of the configurable system's infrastructure.

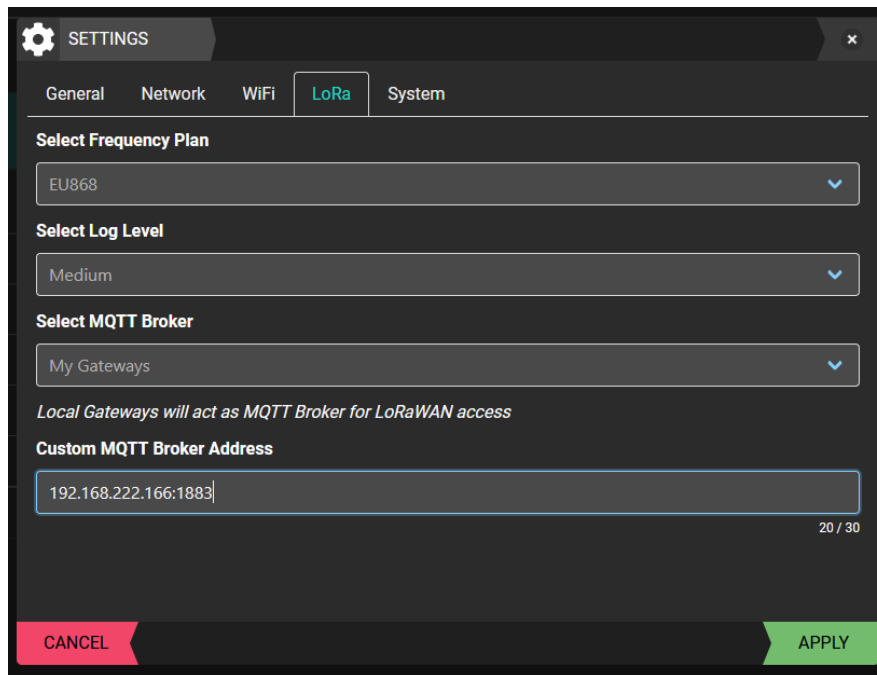


Figure 9: LoRa settings menu in the Network Cockpit frontend

In order to configure the IoT Bridge LoRa, the customer can use the conventional settings dialogue of the components table. This was implemented by adding a new tab into the settings dialogue of the IoT Bridge LoRa, which contains the needed settings for gateway configuration. Customers can change the frequency plan according to their location, as well as the preferred MQTT Broker for application access as shown in Figure 9. MQTT access is granted via username and password, which are changeable over the Local Gateway configuration menu inside the Network Cockpit. When the Secure IoT Gateway's Network Cockpit detects the usage of the IoT Bridge LoRa, a new menu is accessible to the customer, which contains the LoRaWAN device management, as shown in Figure 10.

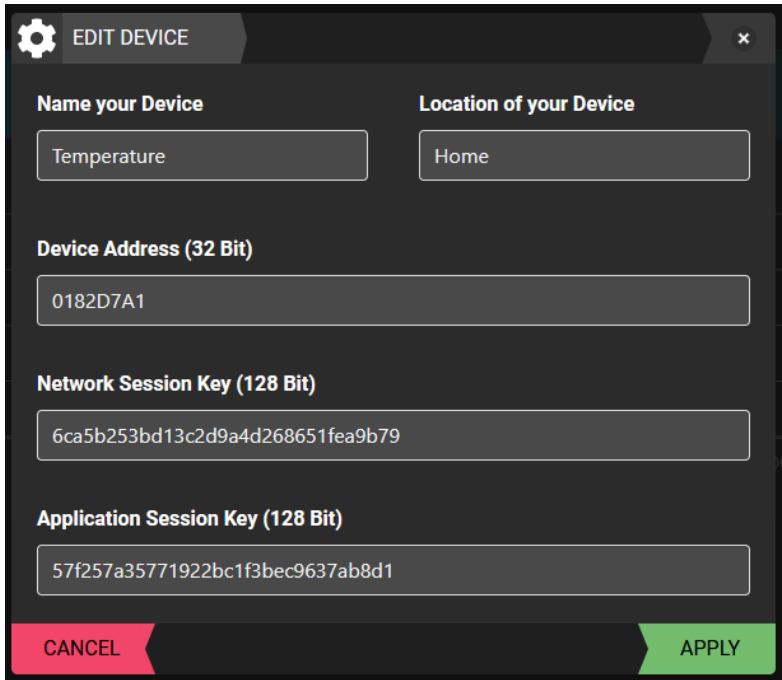


Figure 10: LoRaWAN console inside the Network Cockpit frontend

Monitoring data on the registered LoRaWAN devices is displayed in the extendable section of the table shown in Figure 10. The Secure IoT Gateway provides the following LoRaWAN end device monitoring data:

- Device address of the LoRaWAN end device for identification
- Packet counter of up/downlink packets per device
- The last used SF (Spreading Factor) and bandwidth of the transmission
- Last RSSI (Received Signal Strength Indicator) and SNR (Signal-to-Noise Ratio)
- MQTT topic for application access

Each LoRaWAN device can be configured by the provided settings button. This allows the customer to set the end device metadata, like device name and location, as well as the ABP credentials. The settings are accessible via the modal popup, which unifies the configuration layout with the regular component settings. The settings menu is shown in Figure 11.



**EDIT DEVICE**

**Name your Device**  
Temperature

**Location of your Device**  
Home

**Device Address (32 Bit)**  
0182D7A1

**Network Session Key (128 Bit)**  
6ca5b253bd13c2d9a4d268651fea9b79

**Application Session Key (128 Bit)**  
57f257a35771922bc1f3bec9637ab8d1

CANCEL APPLY

Figure 11: LoRaWAN device settings menu in the Network Cockpit frontend



## 4.2 u.RECS as LoRaWAN End Device

LoRaWAN provides a simple yet practical interface for managing and monitoring the u.RECS in an edge environment. Sensor information like temperature and power consumption, as well as direct management actions like node power control are supported. The functionalities covered by the LoRaWAN interface incorporate BMC related actions by default. The interface is also accessible for installed nodes on the u.RECS carrier via the internal Ethernet, which enables user access to the LoRaWAN network. To access the interface, the BMC provides a REST API to the nodes present on the u.RECS carrier. The u.RECS inherits the role of the LoRaWAN device, which communicates with a LoRaWAN application server. An Espressif ESP-32 WROVER-IE was used as management controller (BMC) for the u.RECS, as later described in the chip selection Chapter 4.2.2. The built-in web interface of the BMC enables access to the LoRaWAN API by authenticating over HTTP Basic Auth. The API usage is further described in Section 4.2.3, as well as the more detailed explanation of the technical background in Section 4.2.1.

### 4.2.1 Overview of the Communication Infrastructure

The communication infrastructure describes LoRa related components and their interactions necessary for transferring data between the u.RECS and an application. The u.RECS acts as a conventional LoRaWAN device, which allows usage of off-the-shelf application servers like TTN or ChirpStack. To differentiate between packets from different modules, the LoRaWAN Frame Port (FPort) byte is set as a module identifier. It allows for a total of 223 different application-specific ports to be used. The LoRaWAN specification states that FPort 0 and 224–255 are used for internal management communication. FPort 1 will be reserved for the BMC, while the remaining 222 Ports can be used by different modules. Each module that uses the HTTP REST API has a fixed port in the 2-223 range assigned. Our first idea regarding module identification and separation was to assign the FPort to an API Key on the BMC, as described in the first deliverable [1]. Since this would imply a proprietary software component on the application server side of the LoRaWAN stack, we discarded this design choice. The proprietary software would have been necessary to map the API Key to FPort assignment on the customer's application endpoint, thus introducing unnecessary complexity to the system. We chose to let the modules freely select the desired FPort directly but reserved FPort 1 for BMC-related communication. The actual implementation of the LoRaWAN API of the BMC is documented in Section 4.2.3.

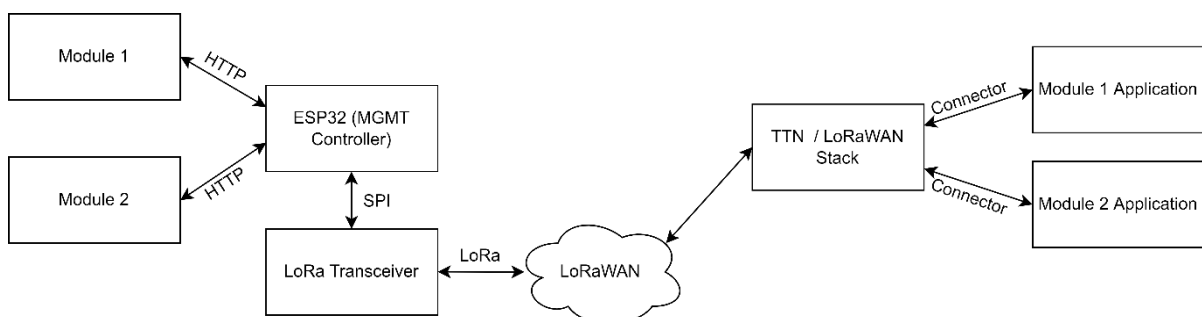


Figure 12: LoRaWAN communication infrastructure of the u.RECS

The communication structure shown in Figure 12 displays a high-level view of the component interactions necessary for LoRaWAN communication for the u.RECS. Module specific applications can schedule or retrieve LoRaWAN data by using different FPorts, which was tested during development with The Things Network.

### 4.2.2 Chip Selection and Integration

A Semtech SX1276 [18] based chip (HopeRF [19]) was selected for integration into the u.RECS platform as LoRaWAN transceiver. It provides the needed functionalities for LoRa MAC layer integration, which is used as a LoRaWAN end device. The provided SPI interface of the LoRa chip is directly connected to the BMC, an Espressif ESP32 WROVER-IE, which is responsible for the basic monitoring and management functionalities of the u.RECS. Those functionalities are also accessible via the LoRaWAN interface, which is further described in Section 4.2.3.

In order to create the MAC layer on top of the physical LoRa standard, a modified version of the LMIC (LoraWAN MAC in C [20]) library is used on the ESP32.

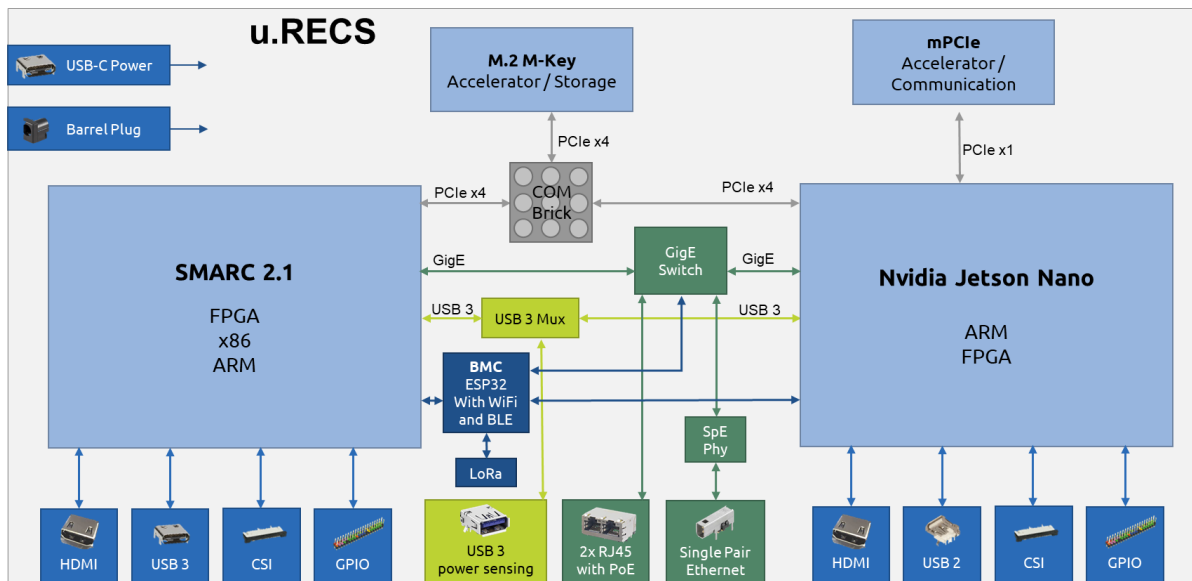


Figure 13: Block diagram of the u.RECS [21]

Having the LoRa transceiver connected to the BMC has some advantages for the u.RECS architecture. On the one hand, the BMC can act as a switch providing LoRaWAN communication to both processing modules (SMARC 2.1 and Nvidia Jetson Nano) and on the other hand, it can be used to remotely control and monitor the system over the air. This enables power-saving features like wake-up from sleep modes for the processing modules and low-power modes where only the ESP32 is running and all other peripherals, e.g. Gigabit Ethernet or USB3, are switched off.

The LMIC library for the Semtech SX1276 on the ESP32 generates a MAC layer on top of the physical LoRa standard, providing easy packet-based IP communication. In the switch mode, the BMC can route these packets to the processing modules via the Gigabit Ethernet interface.

### 4.2.3 API Definition

The API allows u.RECS modules to access LoRaWAN functionalities of the u.RECS, as described in Section 4.2.1., The API in the schema described in Figure 14 is used in order to schedule or retrieve LoRaWAN packets. The FPort can later be used to identify the message sender and receiver. Besides supporting the API for external LoRaWAN functionalities, the BMC of the u.RECS is also able to send out monitoring data as described in Figure 15. Currently we only support the change of node power states as downstream payload on the BMC, as shown in Figure 16, but it is planned to expand on the LoRaWAN control capabilities on the BMC in the future. The previous deliverable [1] defined the API schema with an API

Key authentication in the HTTP Packet body. This was now replaced with the FPort inside the URI path, due to the change of design stated in Section 4.2.1 . Overall authentication at the API is done with HTTP Basic Auth. Figure 14, Figure 15 and Figure 16 are screenshots from the [RECS Wiki](#) [22], documenting the LoRaWAN related API.

### LoRaWAN API

The LoRaWAN interface allows up and downlink connections to an application server. Payload can be scheduled and collected by interfacing the Management [REST API](#).

Attribute	Description	HTTP method
<code>/lorawan/uplink/{fport}</code>	Schedules uplink packet to the application endpoint for the specified fport	POST
<code>/lorawan/downlink/{fport}</code>	Responds with incoming downlink LoRaWAN messages for the specified fport	GET

Example HTTP Body on GET request:

```
<lorawan>
  <payload>{custom lorawan payload}</payload>
  <time>{timestamp}</time>
</lorawan>
```

Example HTTP Body on POST request:

```
<lorawan>
  <payload>{custom lorawan payload}</payload>
</lorawan>
```

Figure 14: Screenshot from the u.RECS wiki showing the LoRaWAN REST API

### LoRa Message

The u.RECS supports upstream and downstream LoRa messages to [The Things Network \(TTN\)](#). The following table gives the LoRa message meaning of version 0.

All system related management communication (excluding the [REST API](#)) uses **FPort 1**.

Upstream message payload layout:

Byte(s)	Description	Unit	Data type
0	u.RECS Lora Message-Version	-	Byte
1	Node Info Smarc/Jetson, Bits: present_smarc / present_jetson / on_smarc / on_jetson	-	2 x 2 Bits
2	Regulators temperature:	°C	Byte (-127..+127)
3	Ambient temperature:	°C	Byte (-127..+127)
4-5	System fan 1:	RPM	Unsigned Short (0..65535)
6-7	System fan 2:	RPM	Unsigned Short (0..65535)
8-9	Smarc Power Usage:	mW	Unsigned Short (0..65535)
10-11	Jetson Power Usage:	mW	Unsigned Short (0..65535)
12-13	u.RECS Power Usage:	mW	Unsigned Short (0..65535)
14-15	USB Power Usage:	mW	Unsigned Short (0..65535)
16-17	mPCIe Power Usage:	mW	Unsigned Short (0..65535)
18-19	M.2 Power Usage:	mW	Unsigned Short (0..65535)
20-21	Ethernet Switch Power Usage:	mW	Unsigned Short (0..65535)
22-23	PoE Eth Port 1 Power Usage:	mW	Unsigned Short (0..65535)
24-25	PoE Eth Port 2 Power Usage:	mW	Unsigned Short (0..65535)
26	PoE Status Port 1	- (see below)	Byte
27	PoE Status Port 2	- (see below)	Byte

Figure 15: Upstream message format for BMC related monitoring

Change power state for node:

Byte(s)	Description	Unit	Data type
0	Lora Message-Version	-	Byte
1	Node <u>ID</u>	-	Byte
2	LoRa Command (0x01 = ON, 0x02 = OFF, 0x03 = RESET)	-	Byte

Figure 16: Downstream message format for Node power control

## 5 Secure IoT Gateway Improvements

The Secure IoT Gateway improvements described in this document build the foundation for the newly added wireless functionalities in this project. A new database structure, backend and frontend were introduced to the Network Cockpit. The IoT Bridge and Local Gateway API have also been renewed in order to improve the system's stability and allow the new wireless infrastructure to be integrated seamlessly. By using the new tech stack of “CIM elements” — the in-house developed Node.js library from Christmann — new UI components were introduced, necessary for creating the wireless configuration interface inside the Network Cockpit as described in Section 5.2. The backend was also altered to process the new configurations necessary for wireless integration, as described in Section 5.1. Besides the integration of the wireless technologies, the Secure IoT Gateway was also hardened and got new features necessary for successful exploitation and product usage. The steps taken to increase the TRL from 6 to 7 are further described in Section 5.3. In order to achieve better user experience, we added new features due to the insights and feedback we got from integrating the Secure IoT Gateway in real working environments. The changes related to the testing feedback are further described in Sections 5.2 and 5.3.

### 5.1 Change of Backend API

The backend API of the Secure IoT Gateway manages the communication and configuration between the components (IoT Bridges, Local Gateways and Network Cockpit). The previous structure of the API was renewed to increase the stability and simplicity of the communication between the locally deployed components and the network cockpit. Each locally deployed component had its configurations stored as separate fields inside the database. This approach was discarded in the renewed API version, due to the lack of flexibility when new configurations need to be integrated. The JSON field type provided by prisma [23] (a Node.js based Object Relational Mapper) is now used to store the key/value pairs of the device configuration, which ensures a constant structure of the database schema. Furthermore, we introduced a queue system to improve the efficiency of supplying locally deployed components with the configuration created by the systems backend logic. Previously, the complete stored configuration was passed to the requesting device, which impacted the performance of the system as described in Section 3.1. The new structure provides only the changes made in relation to the already existing configuration of the local component.

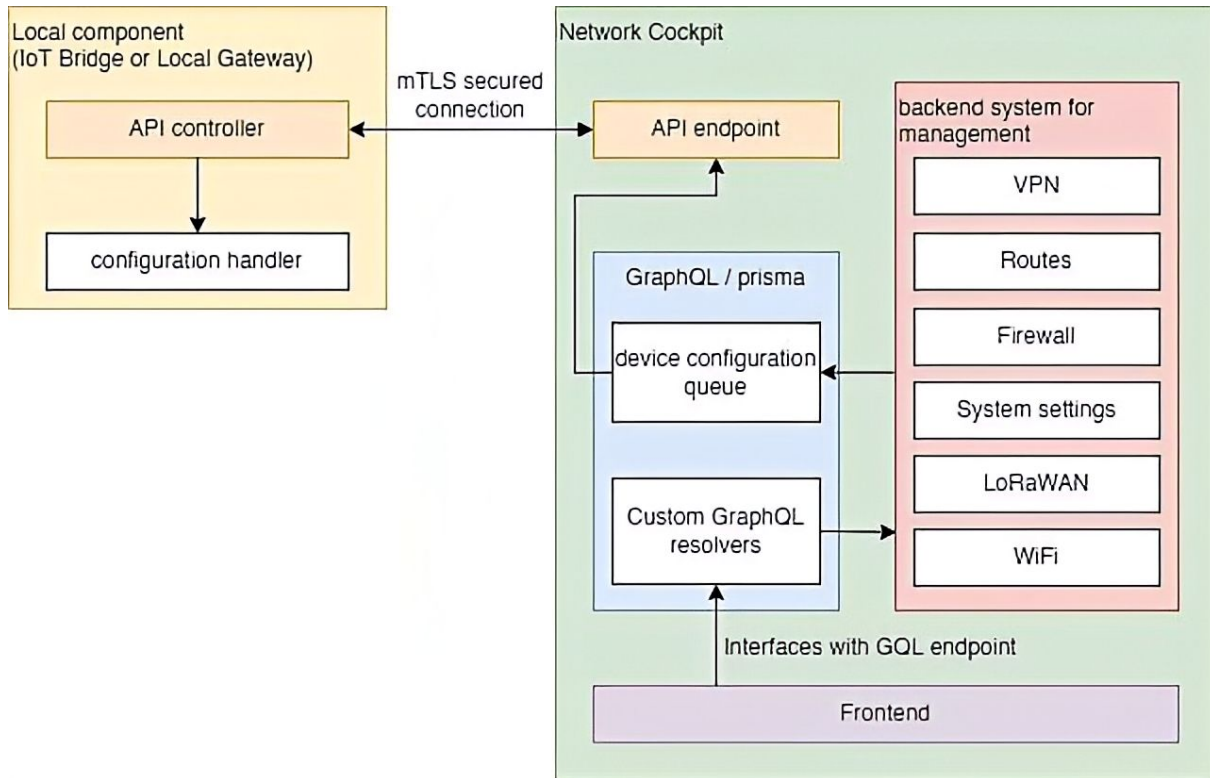


Figure 17: Configuration flow diagram for the Secure IoT Gateway

Figure 17 shows a high-level description of the integrated backend concept regarding the configuration handling of the Secure IoT Gateway. The backend system interfaces with the queue and stores the device configuration based on the settings made by the customer in the frontend. Components are now able to apply configurations locally by requesting the queued configurations at the API endpoint of the Network Cockpit.

## 5.2 Refactoring of the Network Cockpit Frontend

Our company owned Node.js module collection “CIM elements” was used to develop the Nuxt.js based UI. The previous CIM elements version had reached its end-of-life cycle and missed some features for a proper UI integration of the wireless configuration features, so an update of the current UI and tech stack was necessary. Besides missing or still incomplete UI elements provided by CIM elements in the prior version, it also had instability issues, which were fixed in the now-used revision. To prevent visual cluttering of the Networks Cockpit Web Interface when adding the wireless features, a more user-friendly UI structure was implemented. The basic structure is still present in the current frontend revision, but the configuration and monitoring interfaces were completely renewed. The new CIM elements version offers better user management functionalities, necessary for handling the customer accounts of the cloud-based Network Cockpit service. The overall stability of the web interface was also greatly improved by using the newer version, due to the incomplete and experimental state of the first used version.

When it comes to the UI integration, the main navigation layout has changed slightly compared to the previous version of the Network Cockpit described in the Legato deliverable [3].



Figure 18: Screenshot of the Deployed Devices page

Figure 18 shows the “Deployed Devices” page, where customers can see all relevant monitoring data of the selected device in the extendable device table. Deployed Devices is the most relevant view in the Network Cockpit from a customer perspective because it allows the registration of new components via the “Add Component” button shown at the top of Figure 18. Besides registration and monitoring, the selected devices can be configured in the shown view. A search bar and type filter were introduced to handle large component quantities and improve usability from a customer’s perspective. As shown in Section 3.2, the configuration of components like IoT Bridges and Local Gateways are now handled in a separate popup window.

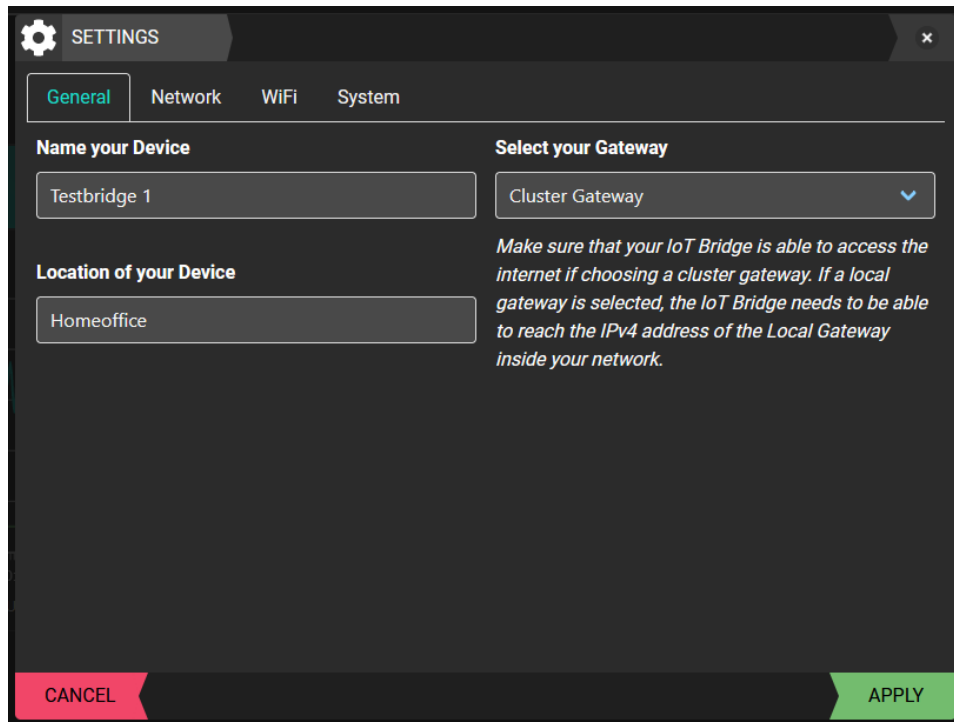


Figure 19: Screenshot of the device settings menu

The tab view shown in Figure 19 allows the device configuration to be separated into the different logical sections. The displayed sections and their content depending on the type of device selected. A Local Gateway or IoT Bridge 100 for instance, would not have the wireless section displayed.

Firewall related management is accessible over the Zones menu in the navbar. A Zone represents a communication unit inside the shielded network provided by the Secure IoT Gateway. Here, the customer is able to group different IoT Bridges together, routing the different subnets of the shielded hosts together. When creating a Zone, all ports are blocked by default inside the Zone, thus lowering the risk of potential security hazards inside the network. The user must manually allow the necessary ports inside the Zone or use the newly added feature of host detection and port scanning, as shown in Figure 20.

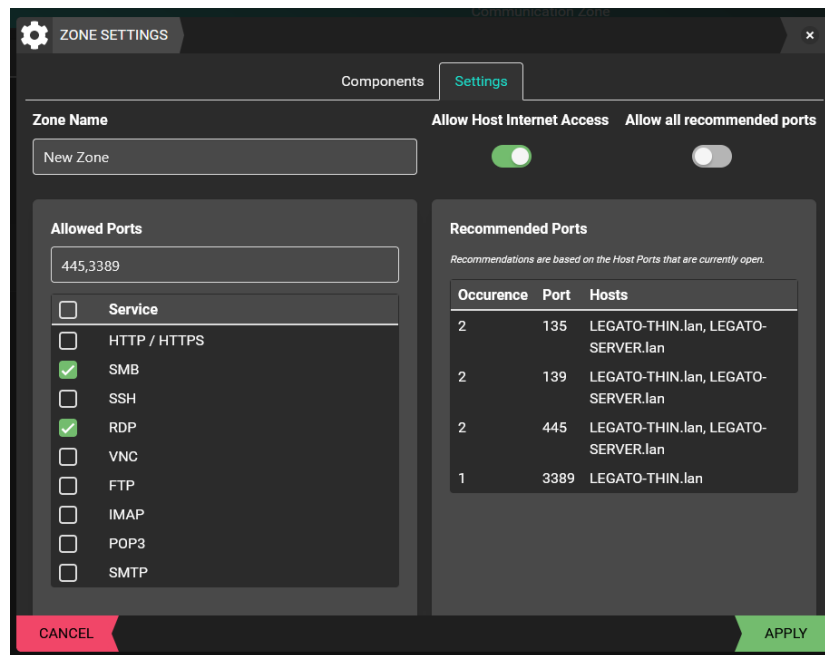


Figure 20: Screenshot of the Zone Settings UI

The port scanning of the shielded IoT devices eases the Zone setup, and the user can directly identify the relevant ports inside the communication unit he wants to create. Zones can overlap by design, allowing a single IoT Bridge and its shielded hosts to join multiple Zones. This can be useful if certain devices should be included on a global Zone topology. Zones allow for an easy way to group and connect different hosts, while maintaining a high level of network security. Routes and firewall rules are abstracted conveniently, allowing the user to easily setup a secure communication infrastructure. Zones in a productive environment are further described in Section 5.3, where we describe the deployed Secure IoT Gateway for the Siemens Electric Motor Condition Classification Use-Case.

Besides Zone management, we also implemented Route Rule management, which can be found as a separate navbar entity inside the Network Cockpit. This feature was identified as a critical part of the application when we collaborated with the Siemens Electric Motor Condition Classification Use-Case and deployed the Secure IoT Gateway at a customer's site. Due to the highly locked-down nature of the Secure IoT Gateway network topology, no external communication was possible to networks outside of the internal VPN tunnels and shielded host subnets. Route Rule Settings allow certain IoT Bridges and Zones, which can be selected in the "Members" Tab in Figure 21, to communicate with external networks. This allows devices to bypass the VPN encryption and communicate with remote networks outside of the Secure IoT Gateway ecosystem. We generally do not advise setting up these bypass rules, but the deployment inside productive networks showed that at least a few hosts in the network rely on external communication. The possible network settings for bypass rules are shown in Figure 21.



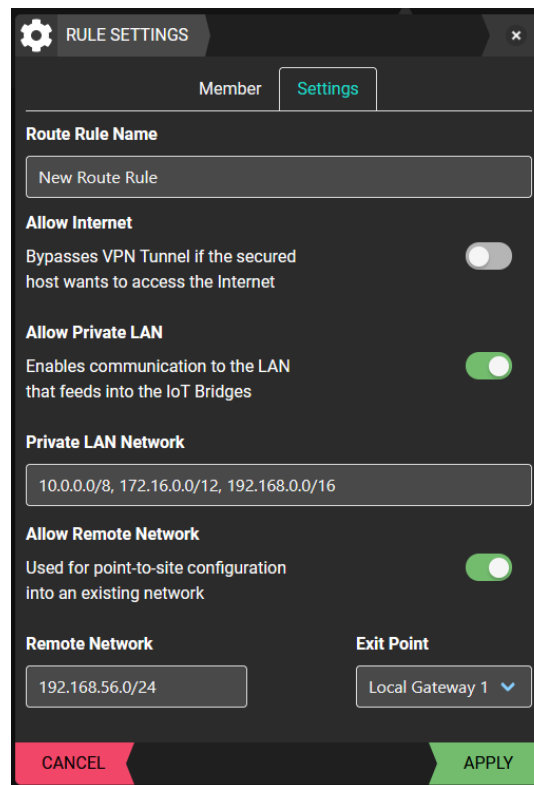


Figure 21: Screenshot of the Route Rule Settings menu

### 5.3 System Hardening and TRL Increase

Hardening the Secure IoT Gateway and increasing the TRL of the project from 6 to 7 was one of our main goals inside VEDLIoT. We deployed the Secure IoT Gateway inside two production environments, collecting user feedback and further improving the system upon the gained insights in the timespan of multiple months. A customer from Christmann was equipped with twelve IoT Bridge 100, connecting remote networks of separate company locations over the cluster gateway. This deployment presented a challenge because this was out of the typical usage scheme of singular host shielding of the system. The deployment required large, shielded subnets that were out of the generation schema that the Secure IoT Gateway provides for the IoT Bridges, requiring manual configurations for some parts of the network infrastructure. It was still possible to deploy a slightly modified version of the system, which allowed us to collect user feedback. When working with the Siemens Electric Motor Condition Classification Use-Case, we could deploy the system in its intended IoT scenario.

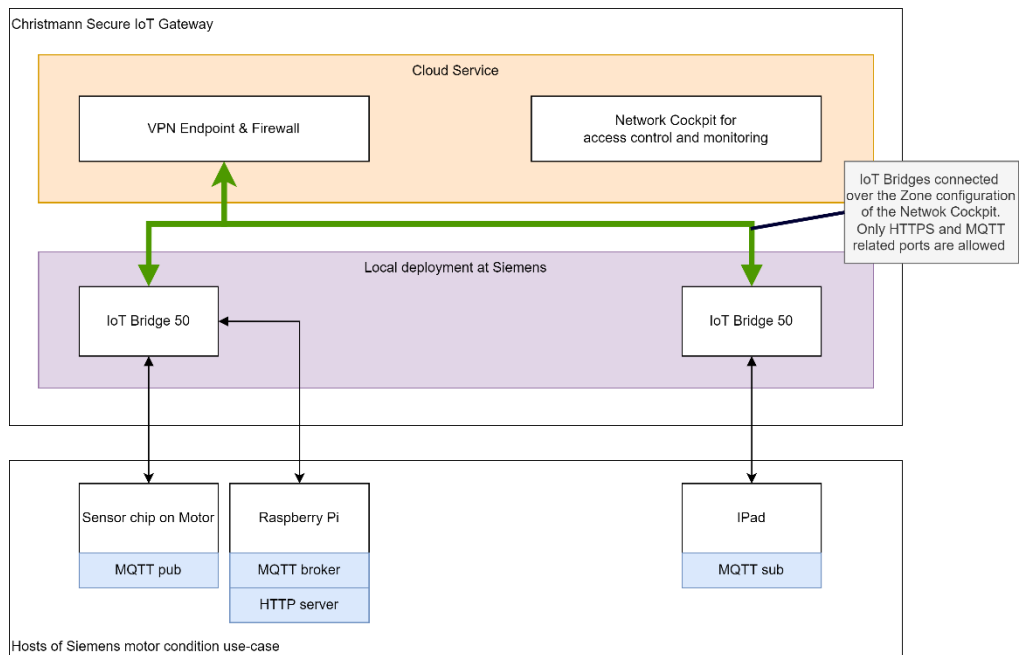


Figure 22: Deployment overview for the Siemens Motor Condition Classification Use-Case

Figure 22 shows the current deployment of the Secure IoT Gateway components inside the Motor Classification Use-Case. We gave our partners full control of the system, creating a customer admin account for Siemens on the production deployment of the Secure IoT Gateway. From there on, the partners were able to configure the devices and tailor the Zone and IoT Bridge configurations to their needs. In this specific deployment, two Wi-Fi-enabled IoT Bridge 50 are in use, which are linked together via a Zone in the Network Cockpit. Figure 22 illustrates the different system components and their required protocols. The Zone, which incorporates the deployed IoT Bridges, allows exclusive communication over the MQTT and HTTP / HTTPS Ports of the system. The deployment at Siemens made us aware of the problem, that some hosts may need to bypass the strict communication rules enforced by the Secure IoT Gateway network topology. Therefore, we implemented the Route Rule Settings, as described in Section 5.2, due to the external inaccessibility of the hosts during the development of the motor classification system from our partners.

In preparation for the deployment at the different customer sites, we had to re-build the testbed to accommodate a productive Secure IoT Gateway instance. All instances required to run the Secure IoT Gateway were split into a development and productive deployment, which are run on a Proxmox virtualization server at Christmann. A new domain was bought, and a separate Nginx Server was configured to split the different systems and make them publicly available later.

One development goal, as stated in the previous deliverable, was to implement automatic firmware upgrades for the hardware components, further improving the security of the system. This was only partially possible. We would like to have an OTA update path for our firmware to further increase the robustness of the system. Right now, firmware updates must be manually deployed by flashing the devices with a new image. A Jenkins build server was set up for automatic staging of OpenWRT builds for our hardware platforms. OTA Updates of IoT Bridge and Local Gateway firmware is still a key element that will be further developed in the scope of a market-ready product. In the scope of VEDLIoT, we rather focused on integrating the new requirements for firewall and routing related management, which were identified by the feedback from our production rollouts.

## 6 Conclusion

One major focus of this Task was the improvement and hardening of the Secure IoT Gateway, further driving product-oriented development of the system. Different wireless technologies have been evaluated. Wi-Fi and LoRaWAN were selected as suitable technologies for the Secure IoT Gateway and u.RECS. Integrating LoRaWAN on the u.RECS allows for a reliable far-edge communication path, suitable for remote deployments of the u.RECS hardware. The Secure IoT Gateway is now able to provide a LoRaWAN infrastructure, as specified in the first deliverable [1] of this task. During this Task, we were able to successfully implement both wireless technologies into the Secure IoT Gateway and u.RECS. Besides the successful integration of the wireless technologies, we also improved the Secure IoT Gateway substantially by adding new features like Route Rules and Zones, easing the usage, and further hardening the system. The Secure IoT Gateway has been rolled out in productive environments since 2022, where we gained useful insights and increased our TRL. Since the integration of Wi-Fi and especially LoRaWAN caused some structural changes, the backend of the Secure IoT Gateway had to be refactored to fit the new requirements. This included the implementation of an API to configure both wireless technologies on the IoT Bridges. In addition to that, the web interface had to be adapted to allow easy configuration by administrative users. The usability and consistency, as well as the maintainability of the Secure IoT Gateway web interface, were also improved.

## 7 References

- [1] VEDLiOT Project, "D4.2 First report on wireless communication infrastructure," 2022.
- [2] "LEGaTO Project," [Online]. Available: <https://legato-project.eu/>.
- [3] LEGaTO D5.3 "Final report on development and optimization of use-cases and integration", 30 November 2020. [Online]. Available: <https://legato-project.eu/sites/default/files/uploaded/d5.3.pdf>.
- [4] I. K. D. H. D. M. B. Phillip Williams, "A survey on security in internet of things with a focus on the impact of emerging technologies," Elsevier B.V, 2022.
- [5] Semtech, "LoRa PHY," Semtech, [Online]. Available: <https://www.semtech.com/lora/what-is-lora>.
- [6] 3GPP, "3GPP Specification Set: 5G," [Online]. Available: <https://www.3gpp.org/dynareport/SpecList.htm?release=Rel-16&tech=4&ts=1&tr=1>.
- [7] SigFox, "Technology | SigFox," SigFox, [Online]. Available: <https://www.sigfox.com/en/what-sigfox/technology>.
- [8] Semtech, "LoRaWAN Standard," Semtech, [Online]. Available: <https://www.semtech.com/lora/lorawan-standard>.
- [9] "OpenWrt Project," [Online]. Available: <https://openwrt.org/>.
- [10] OpenWRT, "[OpenWRT Wiki] The UCI System," OpenWRT, [Online]. Available: <https://openwrt.org/docs/guide-user/base-system/uci>.
- [11] T. T. Industries, "ABP vs OTAA | The Things Stack for LoRaWAN," The Things Industries, [Online]. Available: <https://www.thethingsindustries.com/docs/devices/abp-vs-otaa/>.
- [12] MQTT, "MQTT - The Standard for IoT Messaging," [Online]. Available: <https://mqtt.org/>.
- [13] "OPNsense® a true open source security platform and more," Deciso B.V, [Online]. Available: <https://opnsense.org/>.
- [14] Dragino, "LPS8 Indoor LoRaWAN Gateway," [Online]. Available: <https://www.dragino.com/products/lora-lorawan-gateway/item/148-lps8.html>.
- [15] TheThingsNetwork, "TheThingsNetwork," [Online]. Available: <https://www.thethingsnetwork.org/>.
- [16] "LHT65 LoRaWAN Temperature & Humidity Sensor," Dragino, [Online]. Available: <https://www.dragino.com/products/temperature-humidity-sensor/item/151-lht65.html>.
- [17] ChirpStack, "ChirpStack, open-source LoRaWAN® Network Server stack," [Online]. Available: <https://www.chirpstack.io/>.

- [18] Semtech, "Semtech SX1276," Semtech, [Online]. Available: <https://www.semtech.com/products/wireless-rf/lora-core/sx1276>.
- [19] "RFM95CW-LoRa Transceiver Module," HopeRF, [Online]. Available: <https://www.hoperf.com/modules/lora/RFM95CW.html>.
- [20] T. I. Matthijs Kooijman, "TobleMiner/lmic-esp-idf: A good port of the LMIC LoRaWAN library to esp-idf," [Online]. Available: <https://github.com/TobleMiner/lmic-esp-idf>.
- [21] R. Griessel, "First report on cognitive IoT hardware platform and microserver development," 2022.
- [22] Christmann, "RECS®|Box Wiki," [Online]. Available: [https://recswiki.christmann.info/wiki/doku.php?id=doc\\_urecs:software\\_interface](https://recswiki.christmann.info/wiki/doku.php?id=doc_urecs:software_interface).
- [23] Prisma, "Prisma - Next-generation Node.js and TypeScript ORM for Databases," [Online]. Available: <https://www.prisma.io/>.
- [24] OpenWRT, "OpenWRT Wiki - Opkg," [Online]. Available: <https://openwrt.org/docs/guide-user/additional-software/opkg>.