SAVONIA
University of Applied Sciences

THESIS – BACHELOR'S DEGREE PROGRAMME
TECHNOLOGY, COMMUNICATION AND TRANSPORT

LORAWAN AND IOT PLATFORMS

As part of remote energy metering

AUTHOR Aaro Kinnunen

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Internet of Things	
Author Aaro Kinnunen	
Title of Thesis LoRaWAN and IoT platforms as part of remote energy metering	
Date 5 April 2023	Pages/Number of appendices 52/2
Client Organisation /Partners Savonia UAS, Voimatel Oy	

Abstract

The aim of this thesis was to study IoT communications and how to utilize them in remote electricity energy metering. The goal was to build an IoT measuring and monitoring system that uses Carlo Gavazzi EM24 energy meter and UWPA LoRaWAN gateway. The gateway would send data using Digita's LoRaWAN network to IoT platforms. The commissioner for this work was Voimatel Oy.

The configuration of the system was done in the premises of Savonia UAS. The metering system was configured to send voltage, current, frequency, active power, apparent power, reactive power, the power factor and the active and reactive energy. The LoRaWAN connection to the meter was configured in Digita's network server Thingpark, which Savonia had permissions to use. As a part of this thesis two different software-as-a-service (SaaS) IoT platforms were studied and tested (Azure IoT Central and ThingsBoard). For the IoT platforms, suitable credentials and subscriptions were created. To allow automatic dataflow from the meter to the platforms, JavaScript converter and decoder codes were created for both platforms. In both IoT-platforms custom dashboards were created to allow the remote monitoring of the Carlo Gavazzi energy meter.

As a result of this thesis a functioning IoT system was created, and two possible communicational architectures were created and tested for it. One leading to ThingsBoard dashboard and the other leading to Azure IoT central application dashboard. The results of this thesis were satisfactory and for further implementations of the system Azure was seen as a more useful route as it has other resources that could be utilized in the analyzation of the measurement data.

Keywords

IoT, LoRaWAN, energy, cloud, wireless communication, remote metering, Azure, ThingsBoard, Digita, ThingPark Wireless, electricity, Carlo Gavazzi

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

With the development of Internet of Things (IoT), there have been several new ways different devices may communicate with each other. One of these new communication networks are so called LPWAN-networks (Low Power Wide Area Network). The best benefit of using LPWAN communications is the low power consumption they offer. One of the most talked about of these low powered networks is LoRaWAN network. In Finland Digita offers a vast LoRaWAN network that covers most of the country. One of the aims of this thesis is to study how Digita's LoRaWAN network can be utilized in remote energy and electricity measurements. Nowadays consumers and operators want more detailed information about their consumption of electricity in real time and one of the ways to provide that information is to create manageable IoT systems, where the dataflow is more frequent compared to the traditional systems and the data is always viewable also remotely.

The goal of this thesis is to build and implement a remotely monitorable IoT system, which measures electrical parameters and sends the measurements to a cloud platform, where the data can be analyzed and observed. The system is set to measure single-phase electric power with Carlo Gavazzi EM24 power meter and the measurement data is sent to Digita's LoRaWAN network. From there the data is routed to two different IoT cloud platforms (Azure IoT Central application and ThingsBoard) for analysis. This thesis is mainly focusing on the data communication side of the proposed IoT system, and it also examines the cloud side of IoT systems and studies the cloud platforms used in the proposed system.

The commissioner of this thesis, Voimatel Oy, is a company that specializes on energy and data communication networks. In regards of this thesis, they are specifically interested how LoRa network can be implemented in remote electricity measurements and what kind of data can be sent with the measuring devices. They are also interested in learning more about the IoT-platforms.

2 IOT COMMUNICATION

2.1 Internet of things

Internet of things (IoT) is in a broad sense a cyber-physical system and a network of physical devices that can communicate in some way over the Internet. IoT devices can be sensors, smartphones, different wearable devices, smart meters or practically any device that has the components and firmware that can communicate over the Internet. (Burges 2018.) In Finland probably the most used IoT devices are remotely monitorable electricity meters that are used to monitor electricity consumption remotely and they can be found in almost every household in Finland (Leskinen 2019). The idea of Internet of things is based on data sharing and communication of different devices. This kind of data sharing offers more efficiency on how to do different things, it saves time, money and even emissions (Burges 2018). Figure 1 is a figure made by IBM that depicts one type of architecture of the Internet of things.

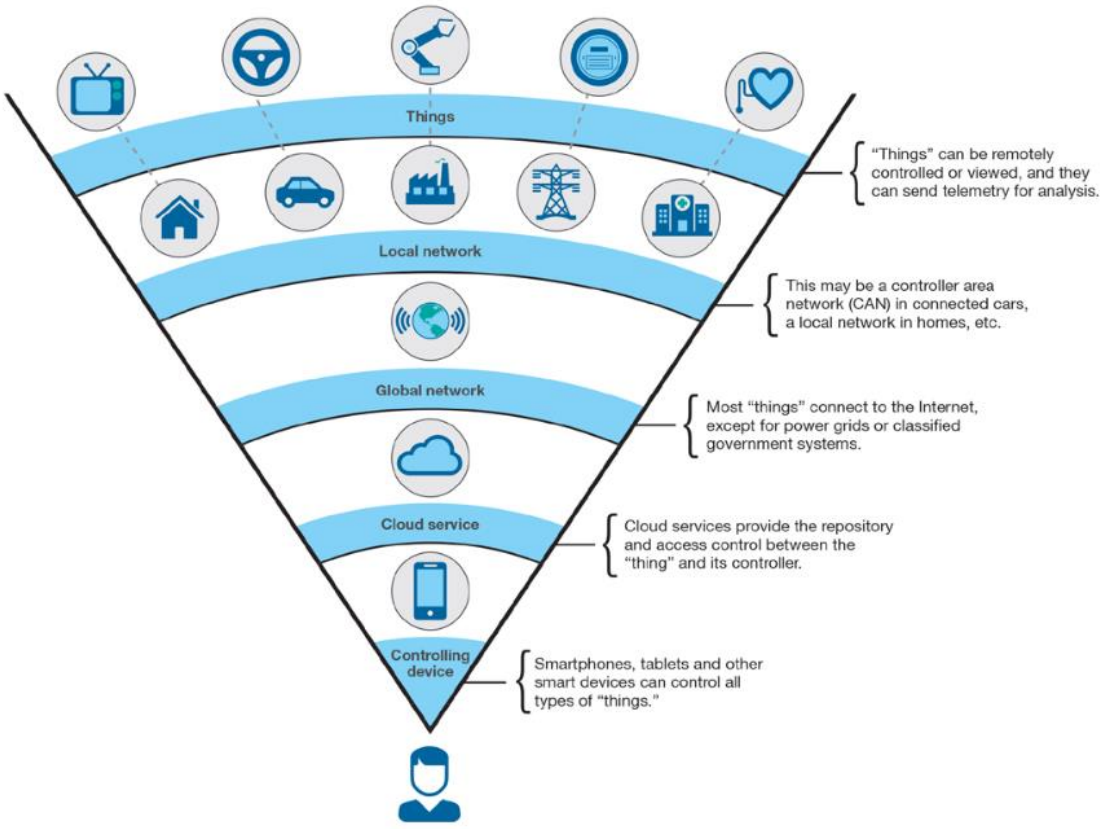

FIGURE 1. What Internet of things consists of (IBM_X-Force 2014)

A Data driven IoT architecture is a model where the data is the key component. Data driven IoT can be divided into three layers as shown in Figure 2: devices layer, edge layer and cloud layer. The device's layer consists of the IoT devices, and the sensors and actuators connected to the actual IoT devices via different IoT gateways. The Edge layer is meant for pre-computing the data the devices send. Edge computing means that data analyzing, and data pre-processing happens on the "edge" of the network locally. Edge computing can help reduce data traffic and cloud costs as less data needs to be sent over the network. The Cloud layer is where the data coming from different devices is combined and stored and further analyzed for example in a custom dashboard. (Gerber 2017.)

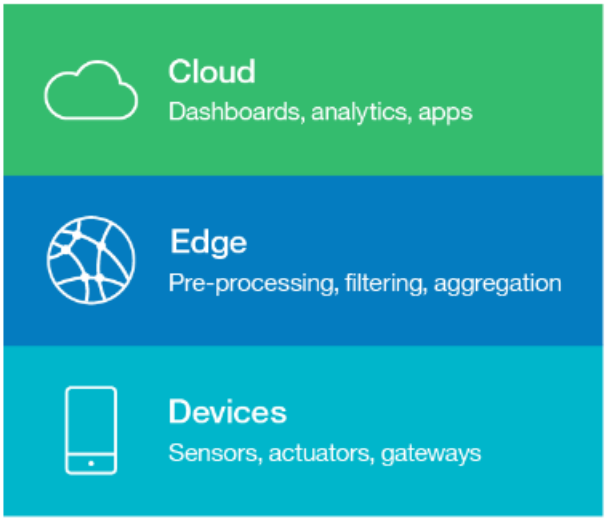

FIGURE 2. Data driven IoT architecture layers (Gerber 2017)

2.2 Wireless communication

Wireless communication means the transfer of data from one point to another wirelessly without any physical medium. Wireless communication is a key aspect of IoT architectures as it provides the data transfer between devices. The protocols of wireless communication can be split in to four different main types by the distance they cover: The Wireless Wide Area Networks (WWAN), The Wireless Personal Area Network (WPAN), The Wireless Metropolitan/Neighborhood Area Network (WMAN/WNAN) and The Wireless Local Area Network (WLAN). (Mokolora 2021.) Figure 3 shows the approximation of the coverage these protocols work. The figure also shows the most known protocols used in the main types. WPAN is the shortest-range wireless communication type, and it is typically used in, as the name suggest, in the reach of a person. The smallest range protocols in this type can be specified as proximity protocols, which includes NFC and RFID, which are used for example in smart locks where a tag opens the lock. Other WPAN networks can be created for example using Bluetooth, Zigbee or Z-wave or other WPAN protocols shown in Figure 3. WLAN works in a bit larger scale than WPAN. It is used in bigger households, offices, and campuses. Wi-Fi is probably the best-known protocol in WLAN networks. After WLAN comes WMAN, which is designed to cover a metropolitan size area. WMANs idea is that it connects several WLANs together and therefore creates a larger scale network. The WWAN is the largest area network, and it can for example connect cities or countries together. The most known WWAN technologies are cellular (2G, 3G, 4G, 5G), Sigfox and the LoRa. (Rouse 2017; Verma 2016.)

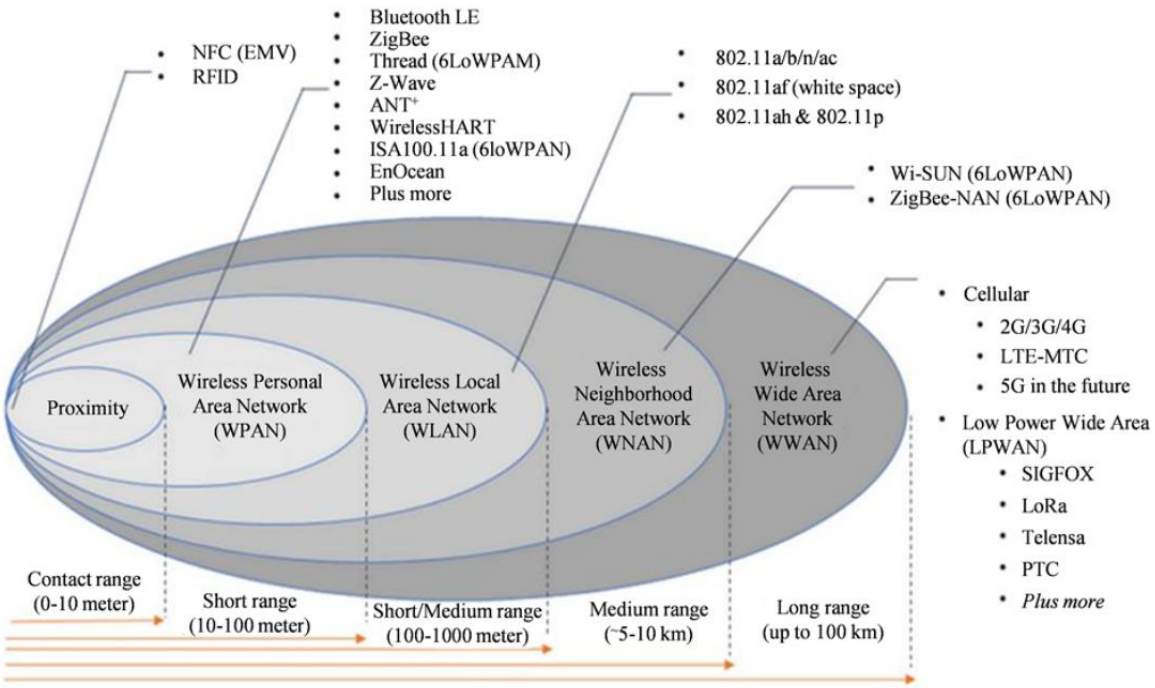

FIGURE 3. Wireless communication protocols and their range (WirelessIoT 2016)

All these wireless communication protocols have a similar structure that contains three main elements: the transmitter, the channel, and the receiver. The transmitter is what encodes the data for the channel. The channel is the medium that transfers the data over a distance to the receiver. The receiver is the destination, and it is where the data encoding is decoded back to its original form. (Mokolora 2021.) This thesis mainly focuses on the wireless wide area networks (WWAN) and more specifically on the low power wide area networks (LPWAN) and the LoRaWAN protocol it includes.

2.3 LoraWAN

LoRaWAN (Long-Range Wide Area Network) is one of the LPWANs (Low-Power Wide Area Networks) designed for IoT devices. It is a communication network intended for wireless and fast but also low-power data transfer. Its main characteristics are two-way data transmission, positioning services, mobility, and easy commissioning of devices to the network. The LoRaWAN network is especially suitable for sending and receiving small amounts of data and usually the data amounts sent are around 10-30 bytes. (Digita-LoRa 2023.)

LoRaWAN was developed in 2009, the development was carried out by a company called Cyclio. The technology was initially known as LoRaMac, due to the MAC-layer used in it (see Figure 5). A company called Semtech acquired Cyclio in 2012. A few years after this acquisition, the association LoRa Alliance was formed, which is currently responsible for marketing the technology, developing, and maintaining the protocol. With the establishment of the LoRa Alliance, LoraMac was changed to now known LoRaWAN. (Slats 2020.)

In the LoRaWAN network, particular attention has been paid to the low electricity consumption of IoT devices, which makes it possible to manufacture devices that run on batteries for years. LoRaWAN is asynchronous, and the devices in it only need to connect to the network when they have

data to send. As a result, the power consumption of IoT devices, such as various sensors, is significantly lower than, for example, devices operating on a mobile network.

2.3.1 Architecture and communication

A typical LoRaWAN network consists of IoT-devices, gateways, a network server, join server and application server. Figure 4 depicts how LoRaWAN comprises communication from an IoT device all the way to the network server, from where traffic is again routed using secure IP network to its receiving software. (Lora-developers 2023.)

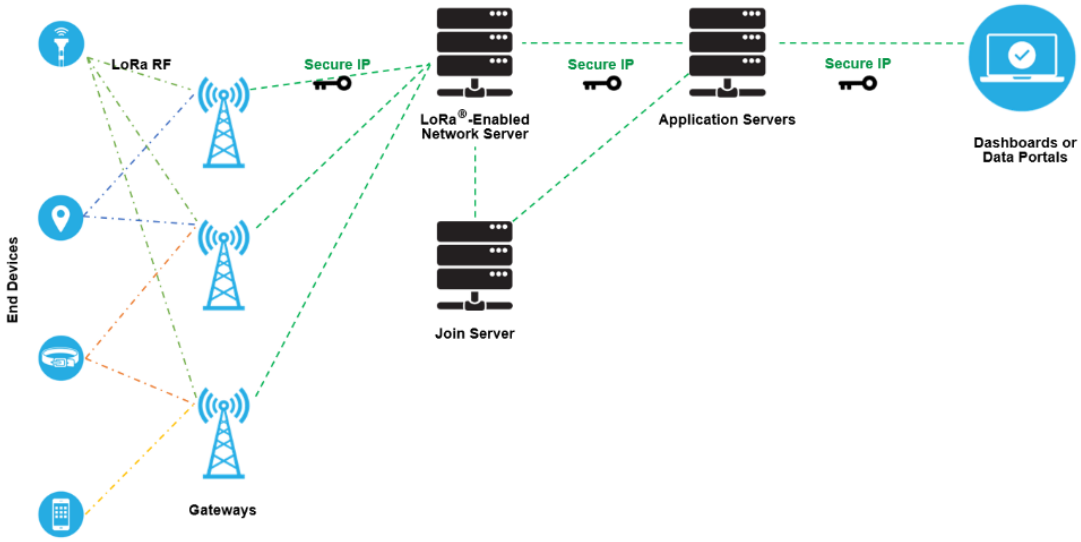

FIGURE 4. Typical architecture of LoRaWAN network (Lora-developers 2023)

As Figure 4 shows the IoT end devices send data wirelessly to radio gateways using LoRa RF Modulation. The LoRa radio frequency modulation is based on chirp spread spectrum (CSS) modulation, which is very fault tolerant, and reflections and diffraction have very low effect on the signal. Usually, the data rate of LoRaWAN network is between 0.3-50 kilobits per second depending on the distances and the amount of data sent. The CSS modulation uses so called "spreading factors" (SF) to transfer data. There are six different spreading factors and lower spreading factors mean faster data rate and shorter distances from device to gateway. The spreading factors are one way a LoRaWAN network can lower devices power consumption as lower spreading factor means lower power consumption. The device data can be received by multiple different gateways inside the reach of the device, which reduces the risks of message lost. The gateways forward the data using IP-protocol to the LoRaWAN network server. This server manages the traffic between the IoT devices, and for example it discards the extra packets that may have arisen when two different gateways receive packets sent by one IoT device. The network server also checks the integrity of the messages using message integrity code (MIC) that uses specific session keys to verify that the message was not tampered during the transmission. If the received LoRa signal from a device is seen as too weak by the network server then the network server can inform the IoT device to adjust the spreading factor and the data rate of the signal, so the signal is better next time but energy consumption remains as low as possible at the same time. The join server handles the device over-the-air activation (OAA)

join process and it contains the session keys of the connected devices. The join procedure and session keys are one of the key elements of the security of LoRaWAN. The application servers manage where the data is forwarded next, for example is it to be viewed on a dashboard or some other application. The application servers also handle any downlinks sent back to them. (Digita-LoRa 2023; Lora-developers 2023.)

The communication of LoRa and LoRaWAN can be viewed in an OSI (Open Systems Interconnect) model. The OSI model is a conceptual framework that represents networking or other telecommunications systems as different layers, each with their own purpose (Simoneau 2006). As Figure 5 shows LoRa is the first, physical layer, of the network, meaning it is responsible of the first connections to the device via the wireless radio modulation (CSS). LoRaWAN is the MAC layer on top of the LoRa physical layer and its functions include for example assigning each device to a frequency, spreading code and data rate, and eliminating duplicate packets. (Lora-developers 2023.)

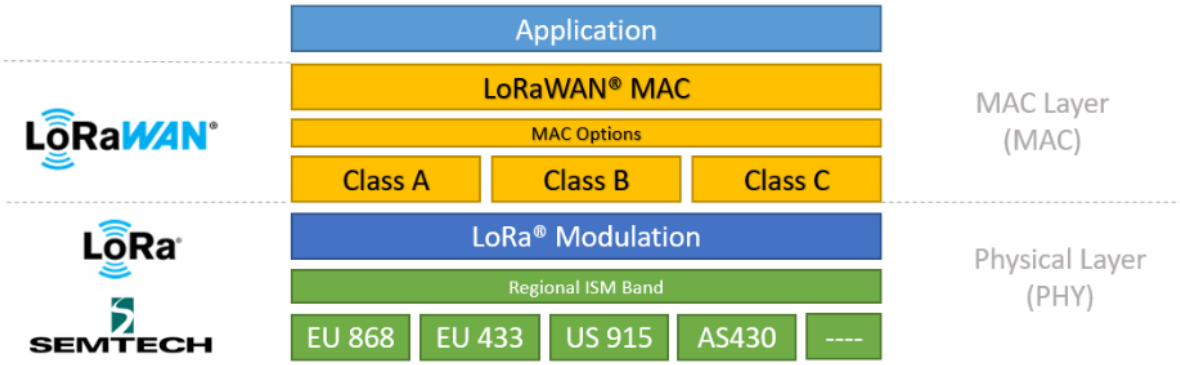

FIGURE 5. The layer model for LoRaWAN (Lora-developers 2023)

LoRaWAN devices communicate in the license-free ISM frequency range using low frequencies below 1 GHz. The frequency bands used vary from region to region, as shown in the LoRa physical layer in Figure 5. In Europe, the 868 MHz and 433 MHz frequency bands are used, in the USA 915 MHz, and in Asia the 433 MHz frequency bands. (LoRaWorkgroup 2018.)

Since LoRaWAN utilizes license-free frequencies, it also reduces its operating costs compared to, for example, commercial licensed frequencies. It is thanks to this that technology has become a strong candidate for various applications, especially in the fields of industry and science.

2.3.2 LoraWAN devices

The LoRaWAN network is ideal for devices in remote locations due to its low power consumption and long range, allowing battery-powered terminals to be installed very far from gateways. The low data transfer rate and the optimization of battery life also limit the amount of data transmitted, in which case the devices usually measure only a few variables at most and transmit the measured values, for example once an hour as raw data in hexadecimal format to minimize the length of the message. The devices usually spend the rest of the time in power saving mode, activating either at regular intervals or, for example, when moving the device. For this reason, popular applications for

LoRaWAN technology include condition monitoring, positioning and remote metering. As the network does not verify the arrival of messages, it is not recommended to use the network in critical infrastructure, where the loss of even one message could endanger people's health or make damage to property. (Semtech 2023; SemtechLoRa-applications 2023.)

There are different connection procedures for LoRaWAN, which are ABP (Activation-By-Personalization) and OTAA (Over-the-Air-Activation). The procedure used affects how the encryption keys used for traffic are handled. In ABP, the keys are pre-written on the device and the device is provisioned into the network with identical keys, so that the device starts to operate directly on the network. In OTAA, the device initially asks for a connection to the network, during which the encryption keys to be used are generated for that session in the join server. OTAA is a more secure method, but ABP may be more functionally secure in some applications because a separate network connection is not required. (ThingIndustries 2023.)

LoRaWAN divides devices into three classes A, B and C as shown in Figure 5 earlier. The division is made according to the latency and power consumption of the downlink. All classes allow two-way communication. The difference between the classes comes from how the device, or node in Lora terms, receives messages from the gateway. Class A and B devices can be battery-powered but class C devices consume significantly more and require a constant power supply. (TheThingsNetwork 2023.)

Class A (All Devices) device sends a message to the access point, after which it has two time-windows in which it can receive an acknowledgment message. At other times, the node is in a state of sleep. Class A is the least power consuming. All end-devices must support at least class A communication. (TheThingsNetwork 2023.)

Class B (Beacon) device has similar time windows for reception, but in addition to them it has an extra period when it listens to future messages. The device opens a receive window after it receives a time-synchronized beacon from the gateway. Timed slots allow the server to talk to a specific node (unicast transmission) or multiple nodes simultaneously (multicast transmission) at a specific time. (TheThingsNetwork 2023.)

Class C (Continuously listening) device continuously receives messages from access points, excluding transmission time. It is the most reliable, but at the same time also consumes the most power. (TheThingsNetwork 2023.)

2.3.3 Digita’s LoRaWAN network

In 2015 Digita and Actility launched a LoRaWAN network solution in Finland (Digita 2015). The network has grown to cover whole of Finland. Figure 6 shows estimate of the LoRaWAN coverage at the beginning of 2023. The network server of Digita’s system is integrated from Actilitys ThingPark Wireless solution (Digita 2015).

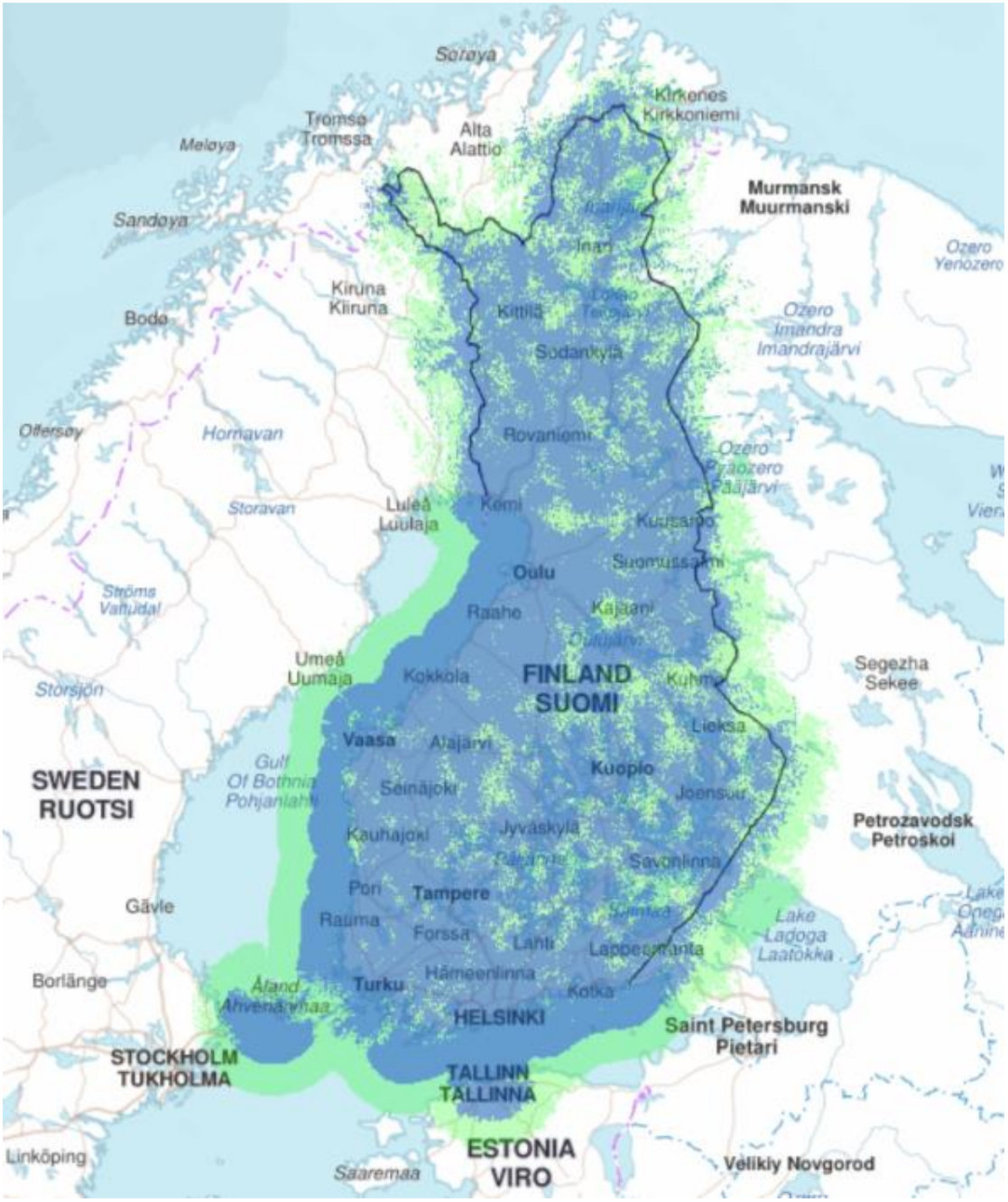

FIGURE 6. Estimate of the coverage in Finland (Blue=Indoor coverage, Green=Outdoor coverage) (IoT-map 2023)

Digita's LoRaWAN network consists of IoT base stations or gateways shown in Figure 7. The stations are usually located in cell towers, and they are light and easy to install (Digita-jyväskylä 2022). The base station shown in Figure 7 is taken at the Jyväskylä Electricity & Light Tele Av -fair in spring 2022, where Digita was representing their IoT-network.

FIGURE 7. IoT-Base station at the Digita booth at Jyväskylä Electricity & Light Tele Av -fair 2022 (Kinnunen 2022, CC BY-NC)

2.4 Remote electricity metering in Finland.

Finland is the first country where almost every household's electric consumption can be read hourly and remotely (Miettinen 2023). Electricity meters remote reading in households became mandatory in Finland in 2013 (TurkuEnergia 2023). Prior to remote readings electric meters needed to be read manually on site and this was done maybe only once a year. The biggest upside to consumers in hourly remote readings is that the electricity billing is based on the actual consumption and not just an estimate. One of the advantages of remotely readable meters is also that they can be used to monitor voltage quality and interruptions more accurately in electricity distribution (Miettinen, 2023). Currently in Finland many big electricity distribution companies are upgrading their smart electricity meters to new faster models, that allow data transfer to be even quicker than earlier. The data transfer in these systems happens every fifteen minutes or so. The new meters also allow to connect the meter to home automation systems via home area network (HAN) port, which allows better optimization of overall consumption of the household. (Kuivasmäki 2023; Laitinen 2023; VantaanEnergia 2023.)

The remote reading of meters is mainly handled over a cellular (GPRS: GSM, 3G) communication or a powerline communication or using radio technology. The power line communication is an older technology that first moves the meter data to a centralized hub or a concentrator via powerlines and from there transmits the data using GPRS. In the GPRS remote reading a GSM-modem or a gateway is installed to the meter or a concentrator, and it then transmits the data using existing cellular networks. (TurkuEnergia 2023.) More recently the use of LPWAN technologies (NB-IoT, LoRaWAN) has become a hot topic in remote readings. In 2018 Telia and Landis+Gyr started a project to bring NB-IoT (Narrow-band IoT) data transmission protocol to smart electricity reading (Kurunsaari 2018). NB-IoT is a LPWAN technology, meaning it has low power consumption. It is based on the existing 4G/LTE -network and commissioning a device to the network is more straight forward than in other previously mentioned traditional protocols. Its benefits over more traditional methods are that it has a good coverage, and it can also reach the remote locations (Kurunsaari 2018). In regards of this thesis a brief comparison was made between LoRaWAN and NB-IoT (see Table 1).

TABLE 1. Comparison between LoRaWAN and NB-IoT (Lorrain 2022)

LPWAN - network	**LoRaWAN**	**NB-IoT**
Deployment options	Public or private	Public
Ecosystem	Open source	Paid
Spectrum	ISM-unlicensed spectrum	Licensed (cellular (LTE))
Power consumption	Transmit: 18mA-84mA, Receive: 5mA. (Acquires less power)	Transmit: 100mA-220, Receive: 40mA. (Acquires more power)
Protocol	Asynchronous	Synchronous
Data rate	0.3 – 50 kb/s	Peak data rate: 250 kb/s
Mobility of devices	Good mobility	Good for stationary devices

In a Landis+Gyr article about the NB-IoT and other communication protocols used in remote monitoring Juerg Haas states that no communication protocol is superior compared to others and the choosing of right protocol should always be made case-by-case basis (Haas 2020).

3 CLOUDS

Cloud computing means that different services including servers, storage, databases, networking, software, and analytics are handled over the Internet. Usually cloud services are provided by different companies. The biggest cloud providers in 2023 are Amazon Web Services, Microsoft Azure, and Google Cloud Platform (Zhang 2023).

3.1 Clouds as a service

Cloud computing can be divided into three models as shown in Figure 8: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS contains the basic aspects of an infrastructure, like storage and virtualization but handled in the cloud. PaaS is where the cloud provider hosts the hardware and software on their own infrastructure and offers the platform as an integrated solution or a service running on the internet. SaaS is probably the most all-inclusive form of the cloud services. SaaS entails a fully managed applications that are run and updated by the cloud provider. (AWS 2023.)

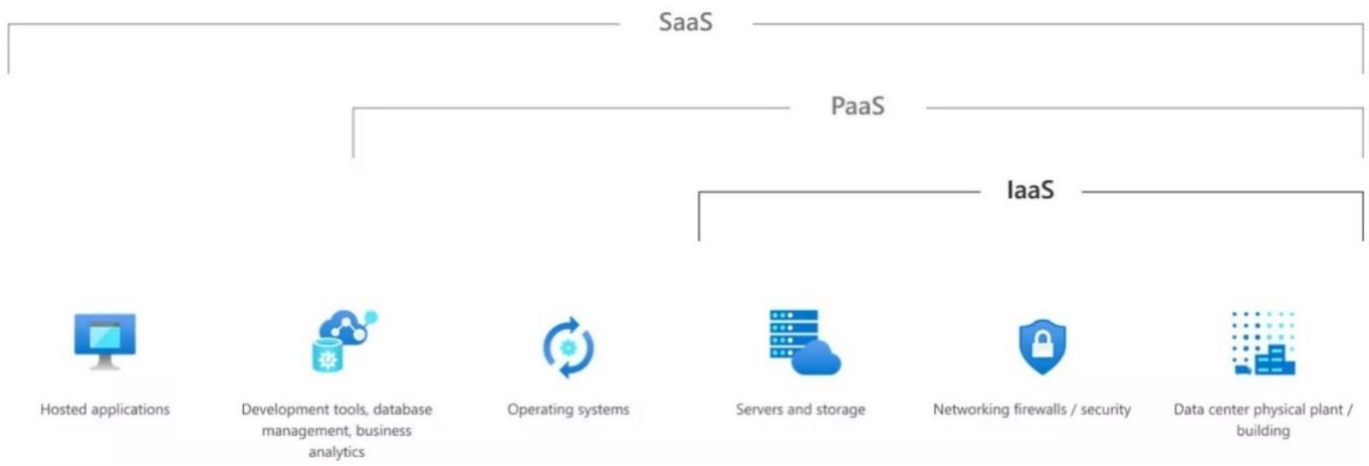

FIGURE 8. As-a-service models (Azure-as-a-service 2023)

3.2 IoT-Cloud platforms

The IoT platform is one of the most important parts when talking about the Internet of Things. The IoT platform brings together hardware, connections, software, and application layers. It offers a powerful solution for managing and configuring devices, data collection and analysis. In addition, the IoT platform enables connection to cloud services or on-premises servers. (Vasanth 2023.)

One of the aims of this thesis was to study few of the IoT platforms and how they could be utilized in remote electricity monitoring. Currently there are many different IoT cloud platforms to choose from, the most popular being Azures IoT solutions, AWS IoT platform and Google Cloud IoT and ThingsBoard (Rana 2022). This thesis focuses on two of these platforms: ThingsBoard and Azure and its IoT Central platform.

3.2.1 ThingsBoard

ThingsBoard is an open-source IoT platform that offers a well-made "out-of-the box" IoT platform solution. Thingsboard offers a free community version as well as a paid professional version. The Community version that can be implemented as PaaS includes the Apache 2.0 license, and it is also possible to take advantage of it for commercial use. It is possible to purchase the Professional version as a ready-made cloud service (SaaS), but there is also the possibility of purchasing it with a "self-hosted" license, which can be maintained on users own server. (ThingsBoardDevelopers 2023.) ThingsBoard has several connectivity options for different devices as seen in the Figure 9, that shows the architecture of the ThingsBoard platform (ThingsboardAuthors 2023).

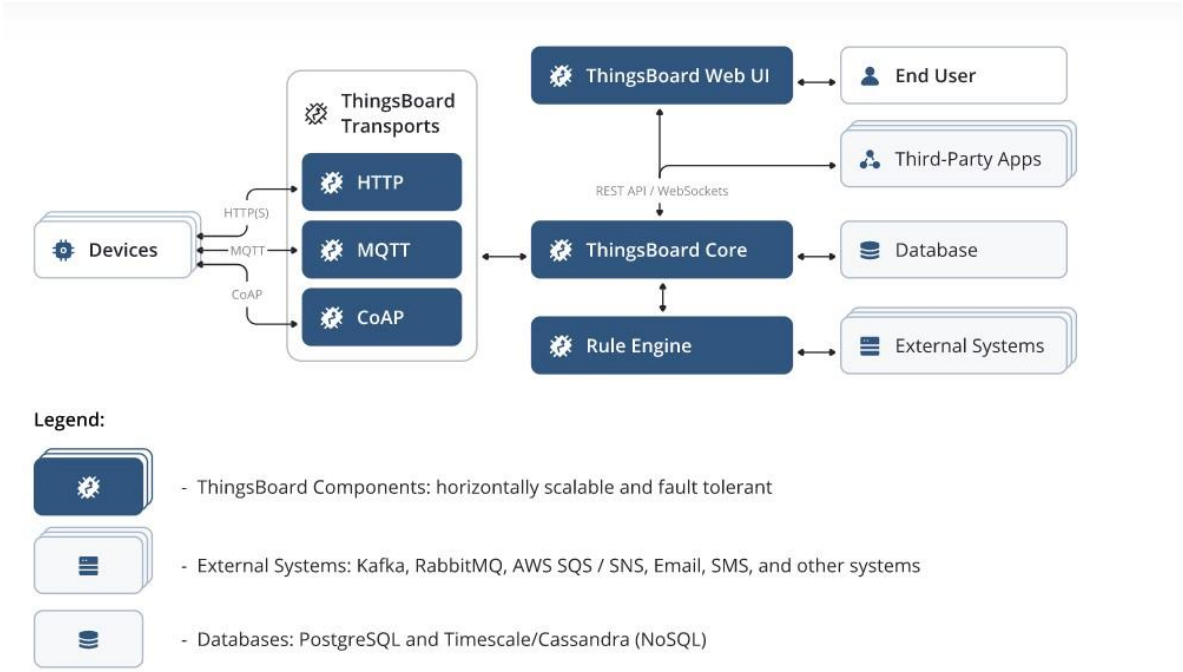

FIGURE 9. Architecture of Thingsboard (ThingsboardAuthors 2023)

ThingsBoard offers rich visualization capabilities and includes a ready-made and easily customizable web-based interface and a customizable dashboard. It is also possible to add self-customized dashboard components to the ThingsBoard dashboard. ThingsBoard's interface also allows users to manage users and create rules. The Professional version has the option to customize the user interface to be suitable and fitting for a company's own image. (ThingsBoardDevelopers 2023.)

The Community version is free, and the cost comes solely from maintaining the platform. The Professional version as self-managed costs from 10 to 500 dollars per month and the cloud version costs from 10 to 800 dollars, which includes the cloud hosting and managing costs. Prices vary depending on the sent data points to the platform as well as the connected devices. Data points are parameters of a message or "telemetry" that the device is sending to ThingsBoard. For example, sending the 'kwh' information once from an electricity meter would take up one data-point. (ThingsBoardPricing 2023.)

3.2.2 Azure and IoT Central

Developed by Microsoft, Azure is one of the market leaders and one of the most well-known public cloud computing platforms. Azure was released in February 2010 as Windows Azure, but soon changed its name to Microsoft Azure. Azure can be used for example as a platform for virtual servers or as a service that provides application platforms. The resources and services Azure provides are related to many different areas, such as artificial intelligence, performance computing, digital marketing, and the Internet of Things. (Azure 2023.)

Azure provides its users with IoT services (see Figure 10) ranging from simple and straightforward solutions for small businesses to versatile and high-level methods that require large resources. Data security and the versatility of the platform are the main features by which the product is described and advertised on the Azure website (Azure 2023). Although Azure is a product of Microsoft, it is still compatible with many third-party applications and operating systems.

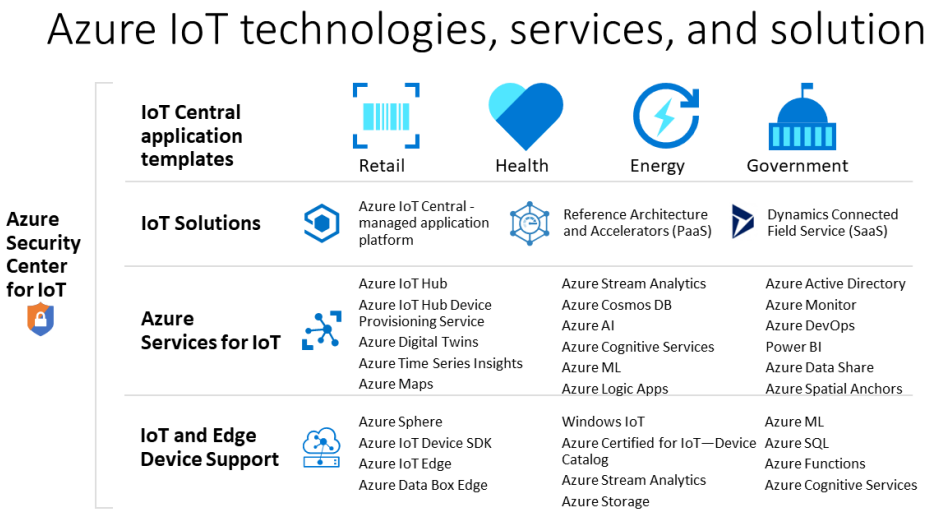

FIGURE 10. Different IoT based solutions from Azure (Betts 2023)

Azure IoT Central is a complete IoT- platform, and its applications have easy-to-use user-interfaces. In an IoT Central application there is possibility to create custom dashboards, secure data and manage devices as seen in the architecture figure of IoT Central in Figure 11. IoT Central itself is application PaaS, but its applications are usually managed and hosted in Azure making it SaaS. IoT central application deployment is easy and there are many readymade templates, as shown in Figure 10, that can be used to speed the process of building the application. (Microsoft_IoTCentral 2022.)

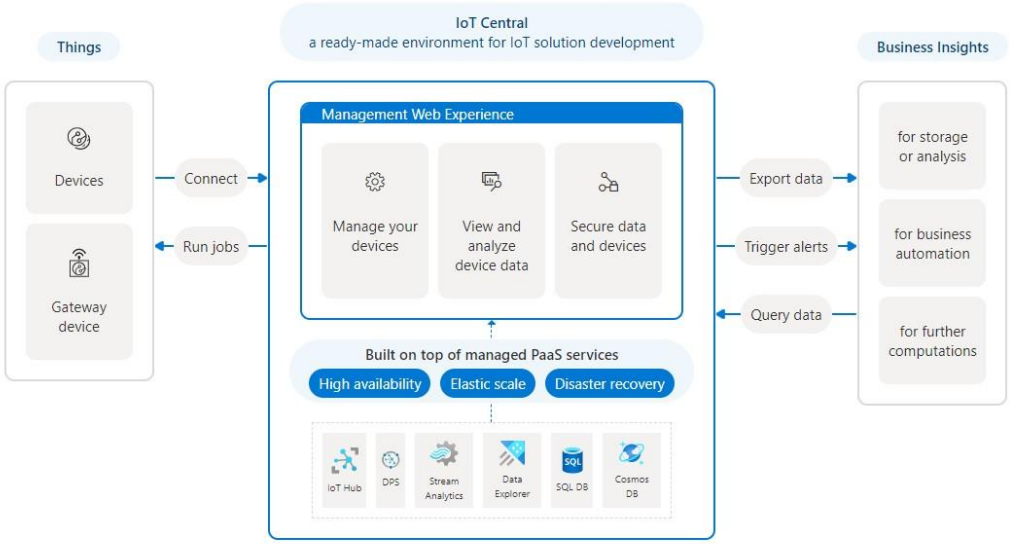

FIGURE 11. Architecture model of IoT Central (IoTCentralArchitecture 2023)

IoT Central applications have a simple and predictable pricing as shown in Figure 12. The pricing is based on the number of messages a device sends to the platform. The price varies also based on how many devices is added to the platform.

Pricing Tier	Standard Tier 0	Standard Tier 1	Standard Tier 2
Use Case	For devices sending a few messages per day	For devices sending a few messages per hour	For devices sending a message every few minutes
Price per device per month	€0.08 per Month	€0.38 per Month	€0.67 per Month
Monthly device message allocation*	400 messages	5,000 messages	30,000 messages
Included free quantities per application	2 free devices (800 included messages)	2 free devices (10,000 included messages)	2 free devices (60,000 included messages)
Overage pricing per 1K messages[1]	€0.067 per 1K messages	€0.015 per 1K messages	€0.015 per 1K messages

* Total message allocation is shared across all devices in an IoT Central Application

[1] The standard message size is 4KB. For example, if the device sends a 4.5KB message it will be billed as 2 messages.

FIGURE 12. IoT Central application pricing (IoTCentralPricing 2023)

4 ENERGY MONITORING IMPLEMENTATION

As part of this thesis a remotely monitorable IoT-system was configured. The system uses Carlo Gavazzi's EM24 as the measuring device and Carlo Gavazzi UWPA as the gateway to Digita's LoRaWAN network. The system utilizes Digita's network server called Thingpark (provided by Actility). The system is monitorable from ThingsBoard or Azure IoT Central application. The physical devices were connected and tested in the premises of Savonia UAS.

4.1 Carlo Gavazzi EM-24

The EM-24 is an energy analyzer made by Carlo Gavazzi. The measurement of energy consumption and the measurement of electrical variables are its main functions. It is suitable for single-phase, two-phase, or three-phase load measurements. It is specifically suited for active and reactive energy measurements and cost allocation. Its interfacing capabilities are RS485, M-Bus, Dupline or Ethernet communication port. The EM24 allows direct connection up to 65A and indirect using external voltage and current transformers. (GavazziEM24 2020).

In Table 2 the main features of the EM24 meter are listed as stated in the device's datasheet.

TABLE 2. Main features of the EM24 (GavazziEM24 2020)

• Energy measurements: total and partial active (kwh) and reactive (kvarh) energy or based on 4 different tariffs; single phase measurements
• Gas, cold water, hot water, kWh remote heating measurements via digital inputs
• TRMS measurements of distorted sine waves (voltages/currents)
• Data encryption
• Compliant with the international accuracy standard IEC/EN62053-21, and IEC/EN61557-12 performance requirements (active power and active energy)

For the proposed system the EM24 was wired to perform single phase measurements of a selected load. Figure 13 shows the measurement system built. The meter physical connections and wirings were done prior to this thesis by a group of Savonia students as a project. As a test load a simple lamp was used to test the data communication to the IoT-platforms. From the EM-24 the following variables were selected to be sent: voltage (V), current (A), frequency (Hz), active power (kW), apparent power (kVA), reactive power (kVAr), the power factor (PF) and also the active and reactive energy (kWh, kvarh) and the timestamp from when these variables were measured.

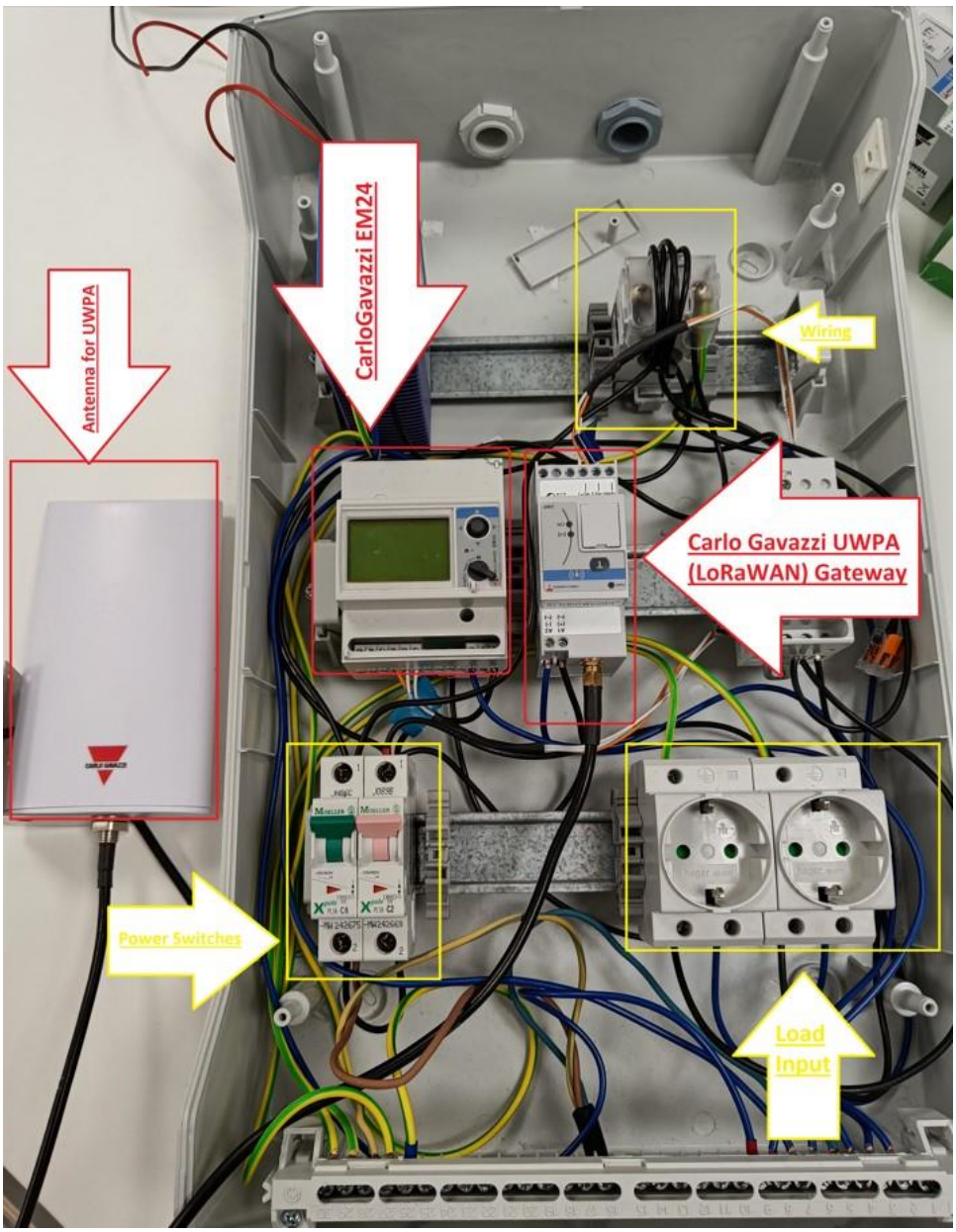

FIGURE 13. The measurement system and gateway (Kinnunen 2023a, CC BY-NC)

4.2 Carlo Gavazzi UWPA Gateway

Carlo Gavazzi's UWPA is an endpoint adapter that provides LoRa and LoRaWAN communication to an RS485 Carlo Gavazzi meter. It is compatible with many of the Carlo Gavazzi electricity meters. It has a long communication range: up to 10 km in open air and in typical applications from 200m to 3km. It is also secured with embedded end-to-end AES128 encryption. (GavazziUWPA 2022.) In the built system the UWPA device was connected to the EM24 meter via the RS485 connection option.

4.2.1 Configuration via UCS7 software

The UWPA and EM24 were configured using Carlo Gavazzis UCS 7 (Universal configuration software) software, that was freely available for download at Carlo Gavazzis website. The UWPA was connected to a computer via microUSB/USB wire to perform the LoRaWAN communication configuration. First step in configuring the device was to create a new communication configuration for the

UWPA device. There were three steps in creating the configuration. First step is shown in Figure 14, where the configuration is named, and the meters system is picked. For this thesis's purposes, the system was put to perform single-phase measurements, but the data transfer would work similar way whether it is single- or three phased. The second step in the configuration was to select the variables that would be sent forward via LoRaWAN. Figure 15 shows the configuration page in which it was possible to select the variables and data to be sent. The sending interval shown in Figure 15 increased when more variables were added to be sent. Due to this the selected variables were limited to ten, to allow packets to be sent every ten minutes. Figure 18 shows the selected variables. The third step shown in Figure 16 was the configuration of the LoRaWAN connection. The activation type of the UWPA device was set to OTAA (Over the air activation). The activation security keys were also generated in this step and these keys were later added to ThingPark described in Figure 23. The internal clock of the EM24 needed to be synchronized in this step to match the current server time, because as a default the clock was set to start from 1970, which was not ideal for the transmission.

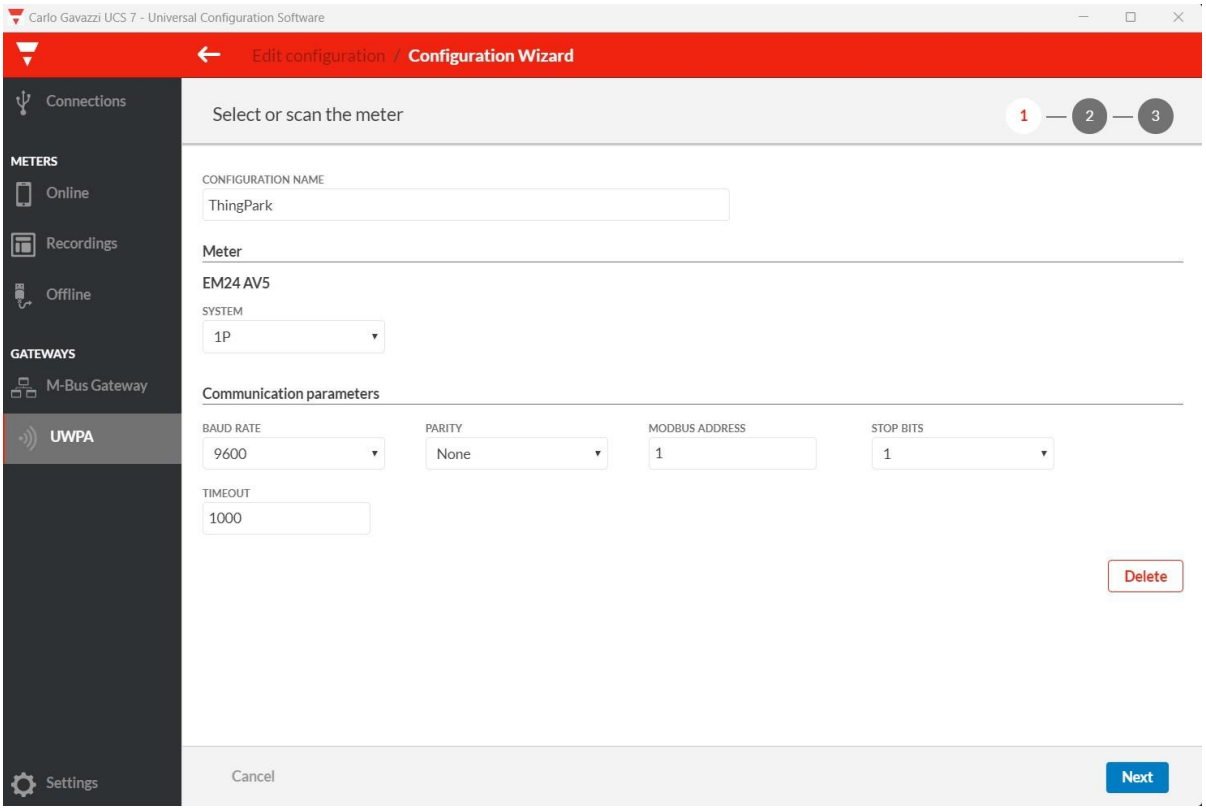

FIGURE 14. First step configuring the UWPA communication to ThingPark

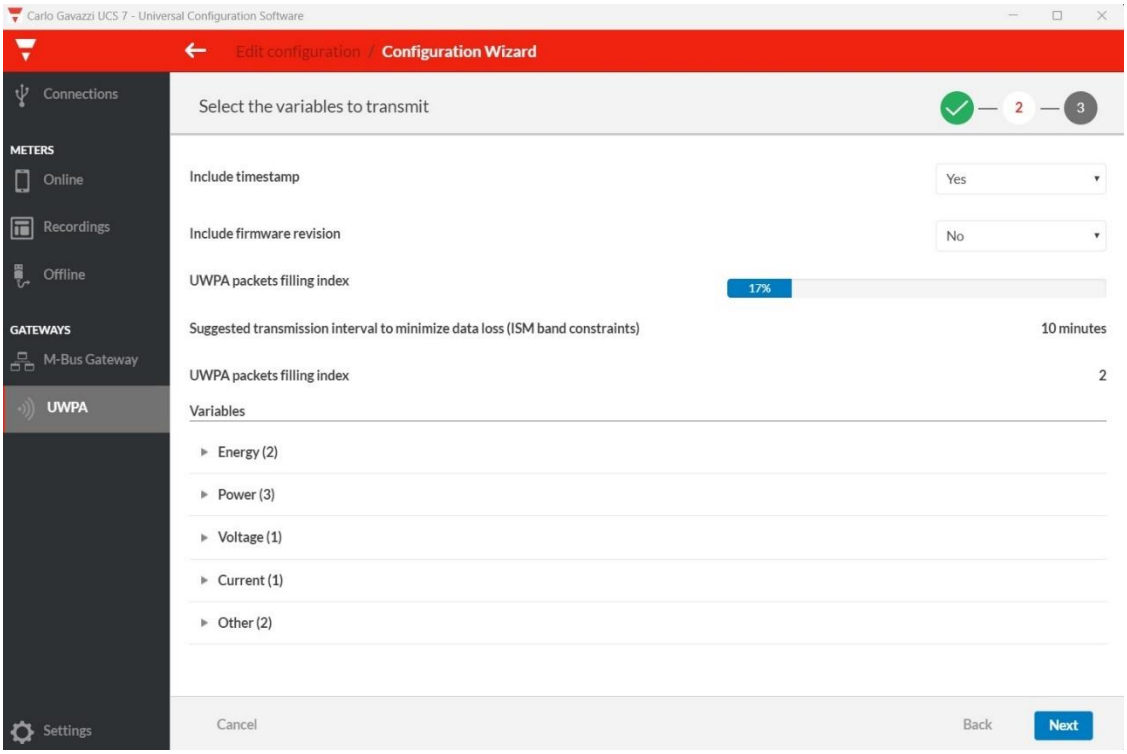

FIGURE 15. Step two: selecting the variables to be sent.

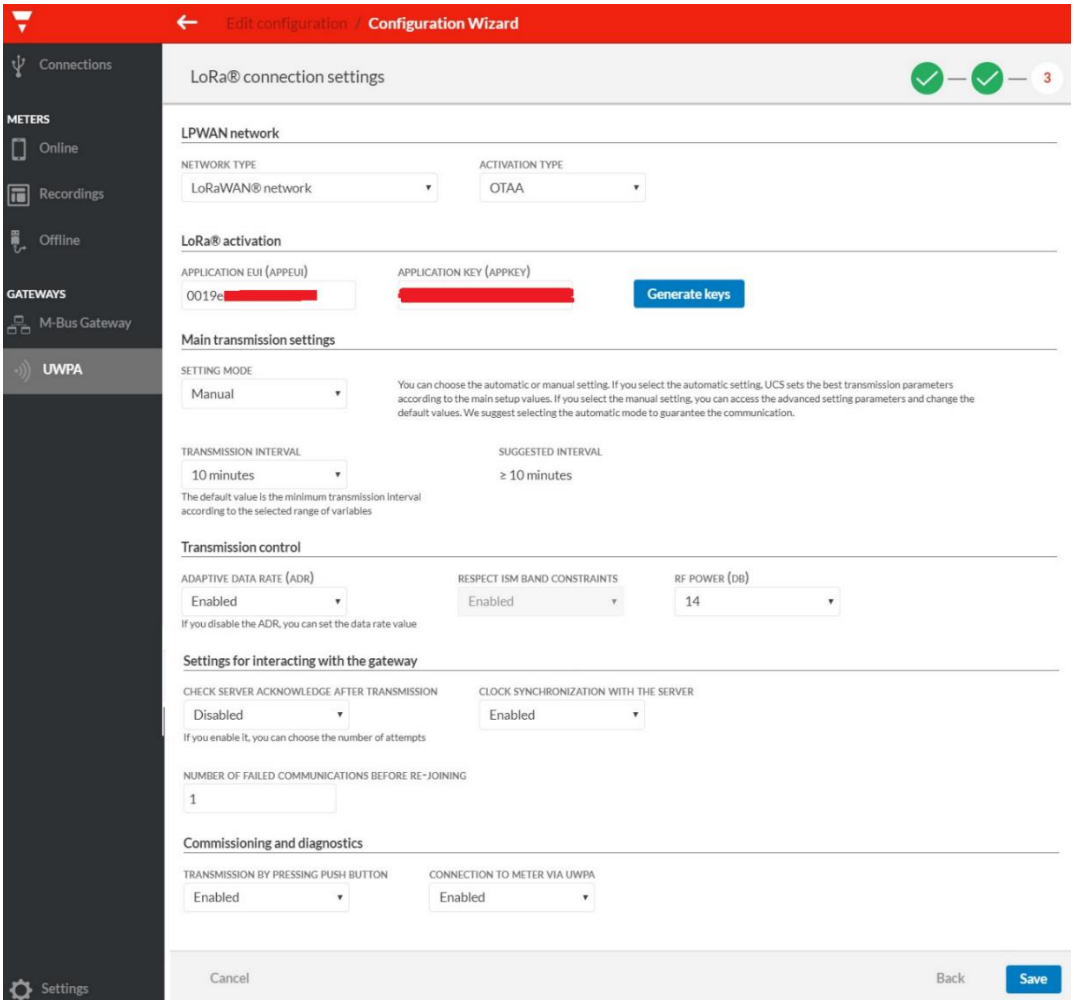

FIGURE 16. Step three of the configuration

After the new configuration was done and saved it was written to the actual device as shown in Figure 17. After that the device configuration in the ThingPark needed to be done. When the UWPA was connected to the Digitas ThingPark the transmissions and the status of the meter could be monitored with the UCS 7 software as shown in Figures 18 and 19.

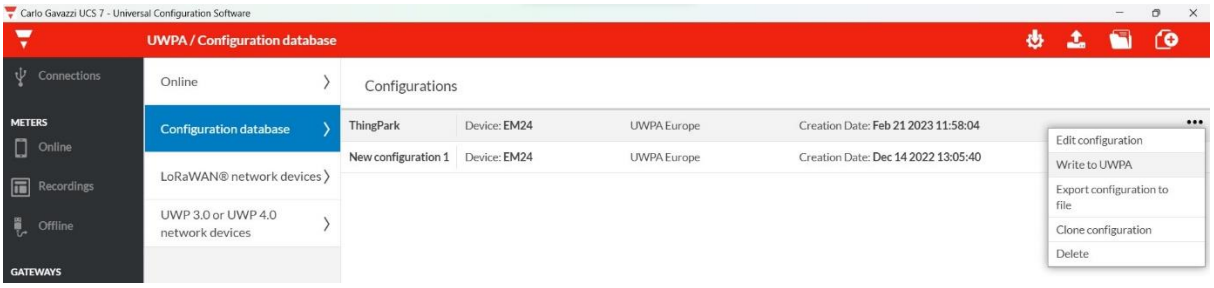

FIGURE 17. After the new configuration was done, it was written to the UWPA device.

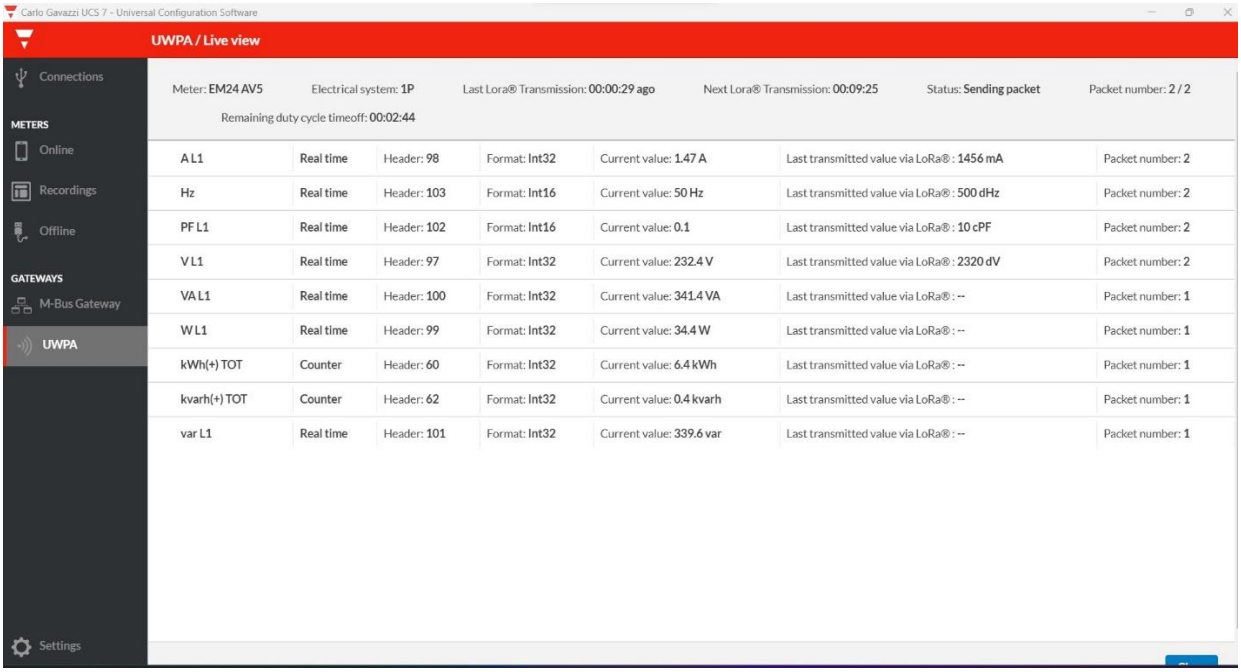

FIGURE 18. The variables to be sent via LoRaWAN shown in the UCS 7s live view of the UWPA.

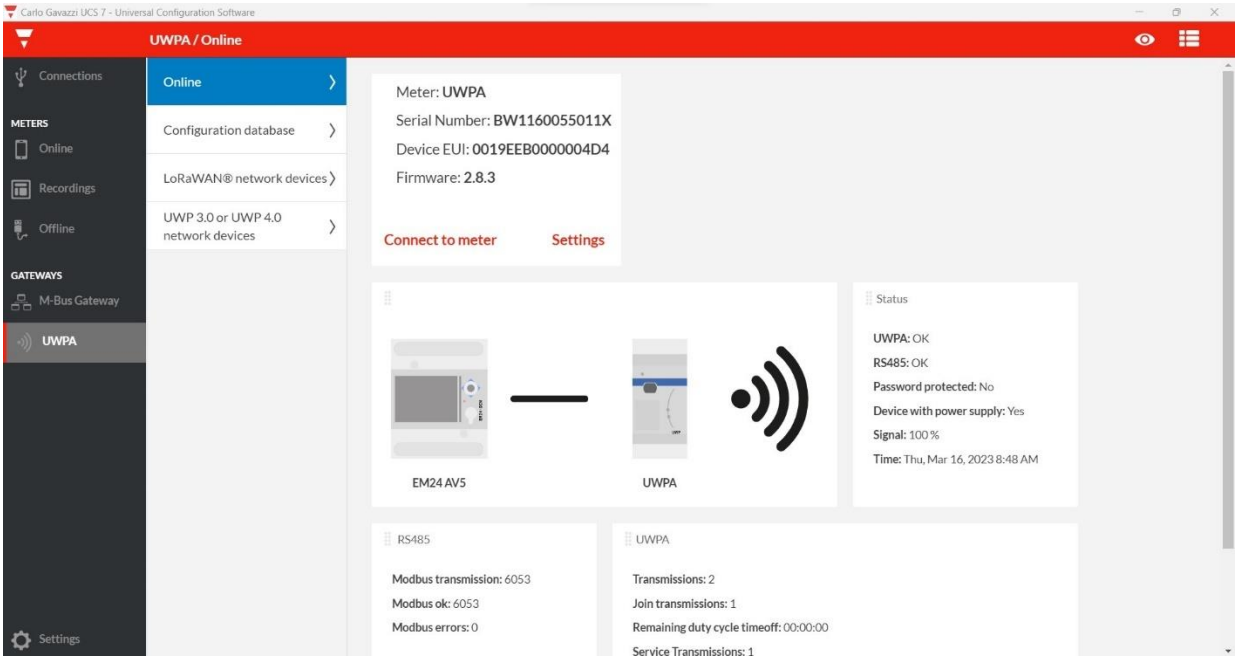

FIGURE 19. Overview of the status and transmissions in UCS7

4.2.2 Hex encoding

The variables are encoded in to hexadecimal before they are sent via LoRaWAN. Figure 20 shows the picked variables and how they are transformed. As can be seen in Figure 18 the UWPA divides the variables in two different packages and then sends them. This is to ensure that data size remains small for the LoRaWAN to work efficiently. The idea behind the hex encoding of the Carlo Gavazzi measurements is explained below in tables 3-5.

First Byte (code)		0x02							
			VARIABLE	ENG. UNIT	UWP Multiplier	LENGTH (bytes)	VARIABLE FORMAT		Note
							HEADER	BODY	
60	sys	counter	kWh (+) TOT	kWh	0,1	9	3Ch	UINT64	
62	sys	counter	kvarh (+) TOT	kvarh	0,1	9	3Eh	UINT64	
97	L1	last istantaneous	V L1-N	V	0,1	5	61h	UINT32	
98	L1	last istantaneous	A L1	A	0,001	5	62h	UINT32	
99	L1	last istantaneous	kW L1	kW	0.000001	5	63h	INT32	
100	L1	last istantaneous	kVA L1	kVA	0.000001	5	64h	UINT32	
101	L1	last istantaneous	kvar L1	kvar	0.000001	5	65h	INT32	
102	L1	last istantaneous	PF L1	--	0,01	3	66h	INT16	positive value=L, negative value=C, 100=PF1
103	L1	last istantaneous	Hz	Hz	0,1	3	67h	UINT16	
255	Others	Time stamp	YY-MM-DD HH-MM-SS			7	FFh	3xUINt16	

FIGURE 20. The selected variables that the UWPA sends in hex form.

TABLE 3. Example of the hex packets sent with empty values.

Packet 1:	02 6300000000 6400000000 6500000000 3c0000000000000000 3e0000000000000000 ff7b02140a0a00
Packet 2:	02 6100000000 6200000000 660000 670000 ff7b02140a0a00

TABLE 4. Dissecting the example hex packet 1.

02	➔ The application header. "02" meaning Single Phase measurements ("01" would be 3-phase)
6300000000	➔ 63 is the header of the "kW L1" variable, and the actual measurement is after the header, total length being 5 bytes, UINT32
6400000000	➔ 64 is the header of the "kVA L1" variable, and the actual measurement is after the header, total length being 5 bytes, UINT32
6500000000	➔ 65 is the header of the "kvar L1" variable, and the actual measurement is after the header, total length being 5 bytes, UINT32
3c00000000000000	➔ 3C is the header of the "kWh (+) TOT" variable, and the actual measurement is after the header, total length being 9 bytes, UINT64
3e00000000000000	➔ 3E is the header of the "kvarh (+) TOT" variable, and the actual measurement is after the header, total length being 9 bytes, UINT64
ff7b02140a0a00	➔ ff is the header of the timestamp, length 7 bytes, format YY-MM-DD-HH-MM-SS. (Example translates to "2023-03-20 10:10:00")

TABLE 5. Dissecting the example hex packet 2.

02	➔ The application header. "02" meaning Single Phase measurements ("01" would be 3-phase)
6100000000	➔ 61 is the header of the "V L1-N" variable, and the actual measurement is after the header, total length being 5 bytes, UINT32
6200000000	➔ 62 is the header of the "A L1" variable, and the actual measurement is after the header, total length being 5 bytes, UINT32
660000	➔ 66 is the header of the "PF L1" variable, and the actual measurement is after the header, total length being 3 bytes, INT16
670000	➔ 67 is the header of the "Hz" variable, and the actual measurement is after the header, total length being 3 bytes, UINT16
ff7b02140a0a00	➔ ff is the header of the timestamp, length 7 bytes, format YY-MM-DD-HH-MM-SS (Example translates to "2023-03-20 10:10:00")

4.3 Digita Thingpark configuration

Building of the monitoring environment for the Carlo Cavazzi EM24 meter began by, creating a user account for Digita's IoT platform. Savonia UAS had acquired the needed licenses for the use of the platform. Digita's IoT platform is built on Actility's Thingpark platform, and it consists of three different applications as shown in Figure 21.

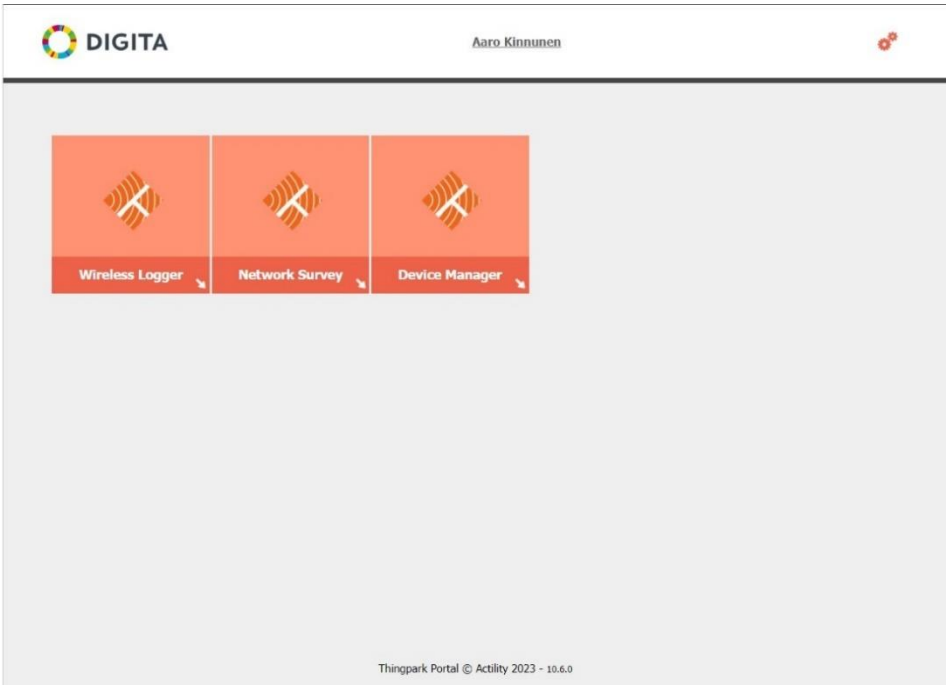

FIGURE 21. Digita's Thingpark consists of three applications.

The Wireless Logger application shows all the incoming and outgoing uplinks and downlinks from the devices added to the account as shown in Figure 22. Device Manager is where adding new devices and managing existing ones is possible. The Network Survey application helps to visualize device data on a map.

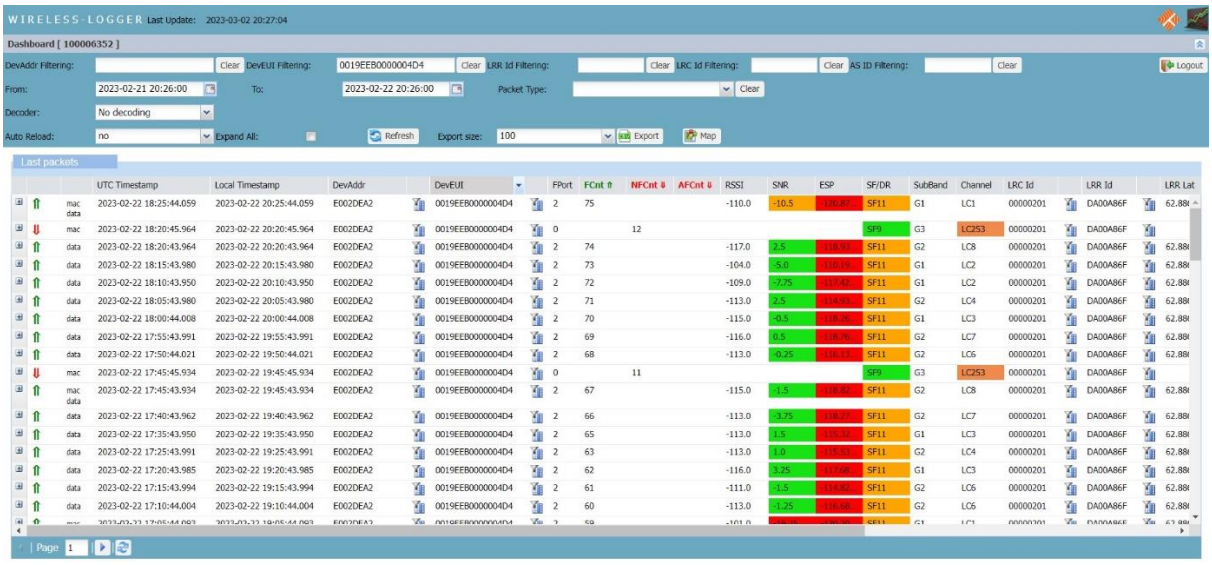

FIGURE 22. Wireless logger shows uplinks and downlinks from devices.

4.3.1 Adding the UWPA-gateway device to Thingpark

Adding the Carlo Gavazzi UWPA device to Thingpark happened in a few simple steps. In Thingpark's device manager, there was a possibility to create a new device. When adding a new device to Thingpark, it asks few details about the device and how will it connect to it. Figure 23 shows information needed to create the connection between the device and Thingpark. The DevEUI address was found labeled in the UWPA devices side. The UWPA gateway device uses the over the air activation and is a Class A device. In Thingpark there are several device profiles for different manufacturers, but Carlo Gavazzi devices were not on the list, so a generic LoraWAN profile was chosen for the device profile. The security hexadecimal codes AppEUI and AppKey were generated when configuring the UWPA devices communication variables with the UCS 7 software.

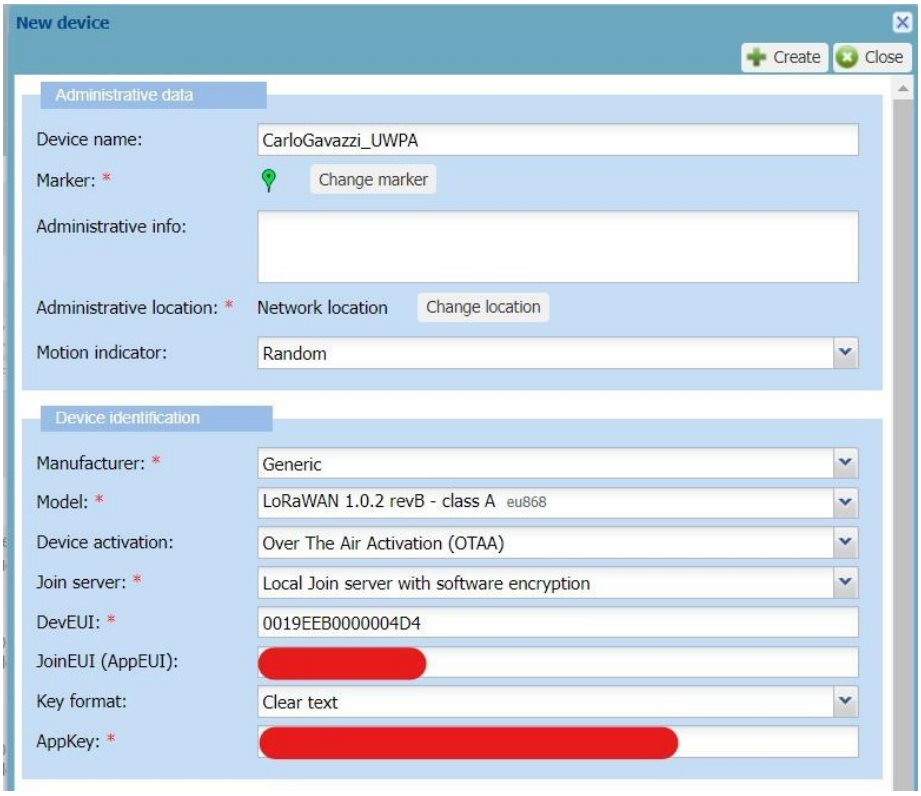

FIGURE 23. Adding a new device to the network happens in the device manager.

When the device was added to the network it sent a join uplink request to Thingpark as shown in Figure 24 and as a reply Thingpark sent a join downlink to the UWPA device. This joining also happens every time the UWPA device is turned off and on again. After this the flow of the measurement data to Thinkpark was possible.

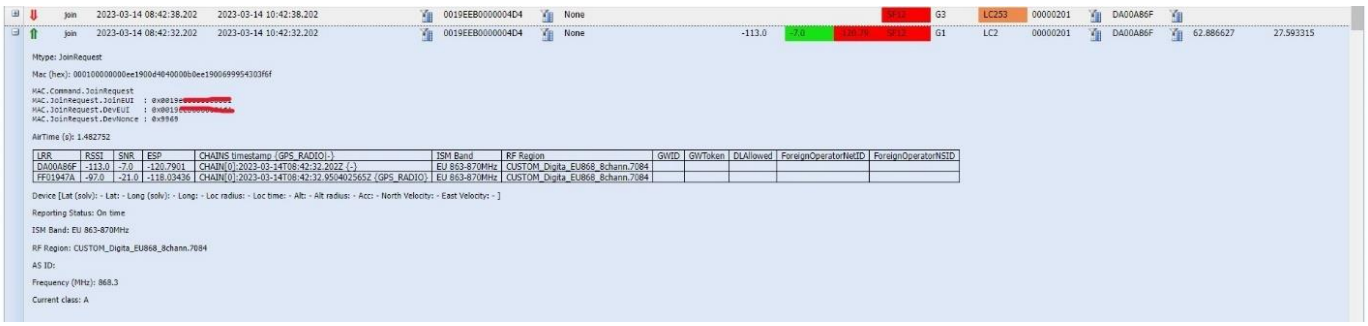

FIGURE 24. Join request from the UWPA shown in the wireless logger.

4.3.2 Network routing

When the device was successfully connected to the LoRaWAN network, and the uplinks were visible in the wireless logger, the next step was to create a route to the IoT platforms. A new HTTP application server needed to be created that handles the sending of the data outside Thingpark. In the details of the new application server the content type needed to be specified to JSON, meaning that the message type leaving Thingpark is in JSON-format. Next a route to IoT-platforms was added to the routing section. For this thesis two different application servers were created: one for ThinsBoard and one for Azure. In Figure 25 route to ThingsBoard was added as the destination IoT platform.

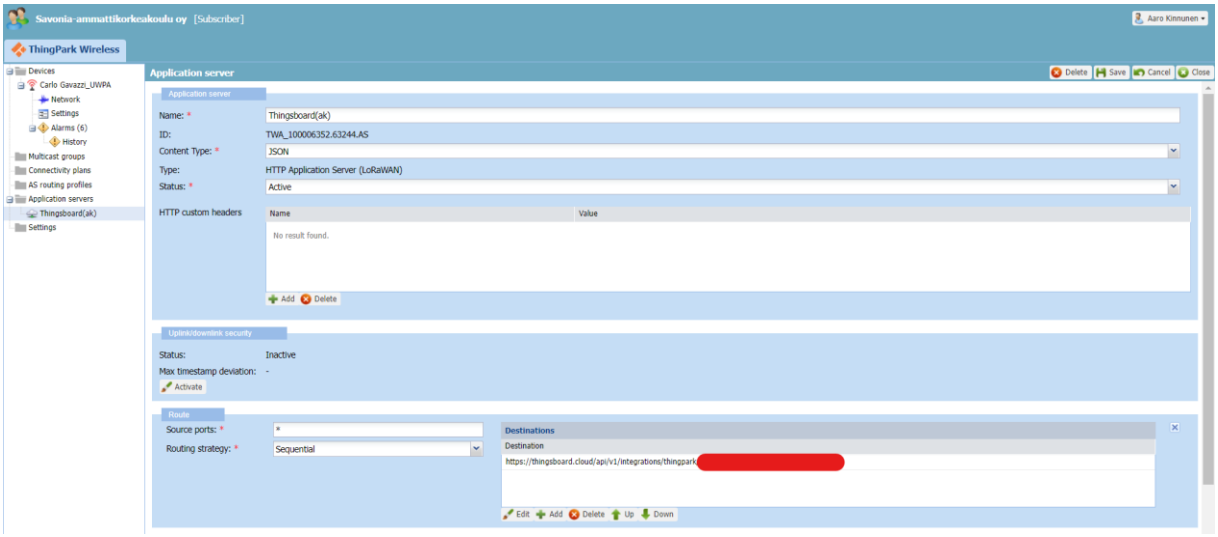

FIGURE 25. Creating an application server

After creation of the application server, it needed to be linked to the device with a custom routing profile that could then be linked to the device. A new routing profile was created, and the previously created application server was added as the destination as shown in Figure 26. This routing profile was then selected as the default routing profile for the Carlo Gavazzi device.

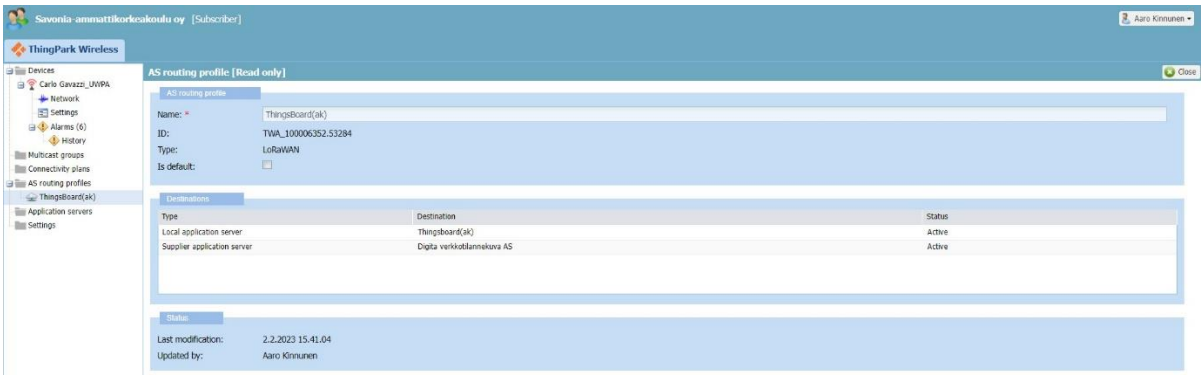

FIGURE 26. AS routing profile

When the network routing was done, Thingpark was then ready to send the measurement data in to the IoT- platforms. When an uplink containing measurement data is received by Thingpark, it places the data into JSON-model shown in Figure 27 and then forwards that message to other platforms via HTTP-POST method automatically.

```
{
    "DevEUI_uplink": {
        "Time": "2023-02-17T11:10:44.353+00:00",
        "DevEUI": "0019EEB0000004D4",
        "FPort": 2,
        "FCntUp": 12,
        "ADRbit": 1,
        "MType": 2,
        "FCntDn": 6,
        "payload_hex": "026120090000627200000066d6ff67f501ff7b020e091400",
        "mic_hex": "126e6d22",
        "Lrcid": "00000201",
        "LrrRSSI": -113.0,
        "LrrSNR": -0.75,
        "LrrESP": -116.401474,
        "SpFact": 12,
        "SubBand": "G2",
        "Channel": "LC6",
        "Lrrid": "DA00A86F",
        "Late": 0,
        "LrrLAT": 62.886627,
        "LrrLON": 27.593315,
        "Lrrs": {
            "Lrr": [{
                "Lrrid": "DA00A86F",
                "Chain": 0,
                "LrrRSSI": -113.0,
                "LrrSNR": -0.75,
                "LrrESP": -116.401474
            }]
        },
        "DevLrrCnt": 1,
        "CustomerID": "100006352",
        "CustomerData": {
            "loc": null,
            "alr": {
                "pro": "LORA/Generic",
                "ver": "1"
            },
            "tags": [],
            "doms": [],
            "name": "Carlo Gavazzi_UWPA"
        },
        "ModelCfg": "0",
        "DriverCfg": {
            "mod": {
                "pId": "generic",
                "mId": "lora",
                "ver": "1"
            }
        },
        "DevAddr": "E002DEA2",
        "TxPower": 16.0,
        "NbTrans": 1,
        "Frequency": 866.8,
        "DynamicClass": "A"
    }
}
```


FIGURE 27. Example of the message body sent to other platforms.

4.4 ThingsBoard configuration

For the ThingsBoard connection a ThingsBoard Cloud account was created, and its free one-month trial of the ThingsBoard Cloud Maker subscription was used as the subscription for the time of this thesis. ThingsBoard offers easy integration of different devices. At the time of this thesis there were up to 27 types of different Integrations available in ThingsBoards Professional edition (ThingsboardIntegrations 2023). For the built system Thingpark type was used as it allows to stream data from the Thingpark servers via HTTP as shown in Figure 28. The Integration option used in this thesis is only available for the ThingsBoard Professional edition, but similar methods can be used also in the free community edition.

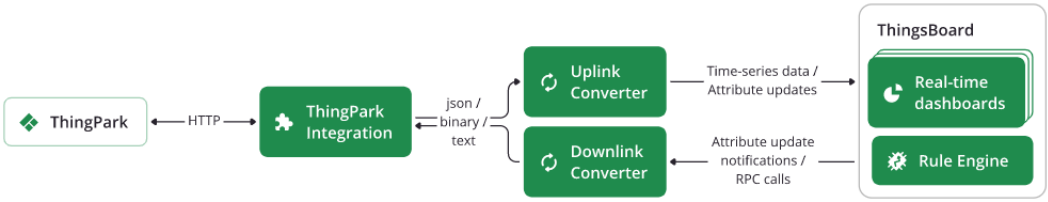

FIGURE 28. Architecture of ThingsBoard-ThingPark Integration (Thingsboard-ThingParkIntegration 2023)

The configuration began with the creation of a new Integration of type Thingpark, shown in Figure 29. An uplink converter was created for it to handle the messages coming from Thingpark. It was decided that there was no need for a downlink converter as there were no reasons to send any custom data back to the device at this stage of the project. The HTTP endpoint URL shown in Figure 29 was added as the destination endpoint to the application server created in Thingpark shown in Figure 25.

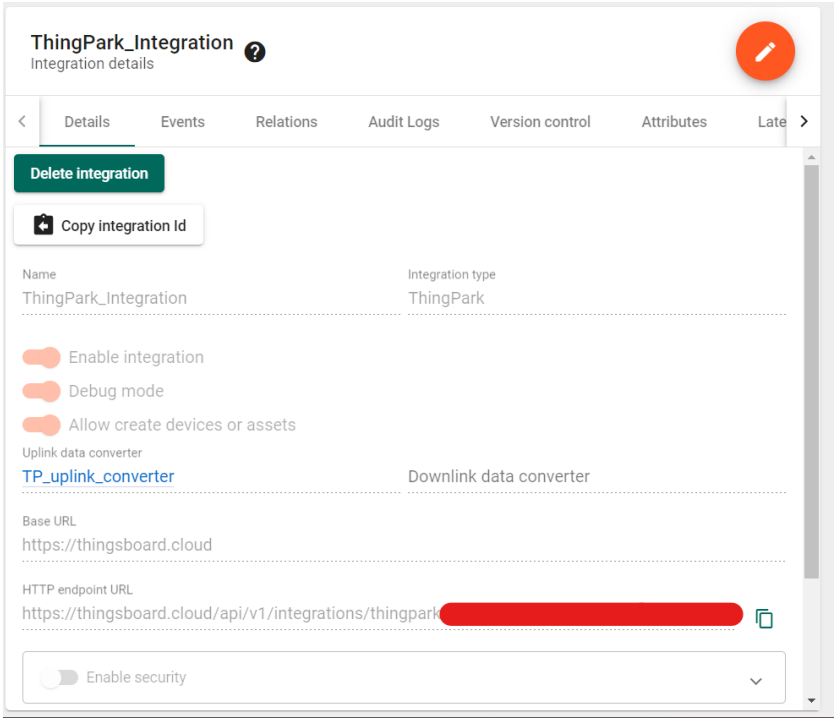

FIGURE 29. The created ThingPark Integration

4.4.1 Uplink converter

The created JavaScript code described in this chapter can be found in GitHub (see Appendix 1).

The uplink converter created for the Integration was a JavaScript code that converts the upcoming data into telemetry for ThingsBoard dashboards. The data coming from Thingpark was in JSON-format and the message body is shown in Figure 27. The key information in the message body was the 'payload_hex' property that holds the actual measurements from the Carlo Gavazzi EM24 meter in a hexadecimal form. To decode the hex encoded values a decoder function needed to be created. The decoder was written in JavaScript, and it transforms the hex values to human readable values. For this thesis, the UWPA was made to send two different kinds of packets with specific variables listed

earlier (see Figure 18). The code was made to read only these specific packages. Figures 30-37 show how the decoding of the uplink message is handled in the code. In the code first it is checked that an 'DevEUI_uplink' was received (see figure 30) and when it is verified then the hex values are taken and turned into bytes (see Figure 31). Then the code checks what kind of hex packet was sent (see Figure 34). There were 4 different main types of packets the UWPA could send as seen in the Figures 33-34. As mentioned earlier the UWPA was made to send two different packets of type 1P, this is handled in the 'getpackets()' function shown in Figure 34. The actual decoding of each value is described in Figures 35 and 36. The code also handles any possible error messages derived from the EM24 meter (see figure 33 and 37).

```
/**
 * Start decoder
 * @type {any | undefined}
 */
var payLoadJson = decodeToJson(payload); //turns payload to JSON form if not already
var result = setPayload(); // variable the decoder function returns
var payload_hex;
var payloadByte = [];
var telemetryC;
var devEuiLink;


/** Helper functions **/
/**
 * Parsing metadata and payload_hex
 * @param payLoadJson
 * @returns {null|{payload: {deviceType: string, telemetry: string, deviceName: string}}}
 */
function setPayload() {
    if (payLoadJson !== null) {
        if (payLoadJson !== null && payLoadJson.length !== 0) {
            if (payLoadJson.hasOwnProperty('DevEUI_uplink')) {
                devEuiLink = payLoadJson.DevEUI_uplink;
            } else if (payLoadJson.hasOwnProperty('DevEUI_downlink_Sent')) {
                devEuiLink = payLoadJson.DevEUI_downlink_Sent;
            }
            if (devEuiLink !== null && devEuiLink.length !== 0) {
                var payloadResult = getPayload();
                if (devEuiLink.hasOwnProperty('payload_hex')) {
                    payload_hex = devEuiLink['payload_hex'];
                    if (payload_hex !== null && payload_hex.length !== 0) {
                        payload_hex = payload_hex.trim();
                        payloadByte = hexStringToBytes(payload_hex);            // Convert the hex values to bytes
                        payloadResult.telemetry = (payLoadJson.hasOwnProperty('DevEUI_uplink')) ? getTelemetryUp() : getTelemetryDown();
                    }
                }
                else{
                    //payloadResult.telemetry =getTelemetryDown();       // if downlinks would be used
                }
                payloadResult.deviceName = devEuiLink['DevEUI'];
                return payloadResult;
            }
        }
    }
    return null;
}
```

FIGURE 30. The main function in the converter is the "setPayload" function.

```
function hexStringToBytes(str) {
    var array = str.match(/.{1,2}/g);
    var a = [];
    array.forEach(function (element) {
        a.push(parseInt(element, 16));
    });
    return a;
}
```

FIGURE 31. Function that turns the hex string to array of bytes

```
    /**
     * Metadata
     * @returns Headers payload: {deviceType: string, telemetry: string, deviceName: string}
     */
    function getPayload() {
        return {
            'deviceName': "",
            'deviceType': payLoadJson.DevEUI_uplink.CustomerData.name,
            'Timestamp': payLoadJson.DevEUI_uplink.Time,
            'telemetry': ""
        }
    }
```

FIGURE 32. The form the decoder returns the payload

```
var constants = {
        'commonMessageHeaderSpr':
        {
            'type':                          //Application headers for different packets
            {
                01: "Application 3P",
                02: "Application 1P",
                03: "Application VMUMC-OC",
                255: "Application Error"
            },

        },
        'Error_message': {
            1: "Autoscan in progress",
            2: "No meter connected",
            3: "Invalid meter",
            4: "Invalid meter setup"
        },
    };
```

FIGURE 33. Application headers and error messages

```
/**
 * Receiving telemetry by value message_type
 * @param payload_hex
 * @param payLoadJson
 * @returns {null|Telemetry }
 */
function getTelemetryUp() {
    var message_type = payloadByte[0];          //Checking the application header
    switch (message_type) {
        case 01:                                // "Application 3P"
            return getTelemetryCarlo_3P();
        case 02:                                //"Application 1P"
            return getpackets();
        case 03:                                // "Application VMUMC-OC"
            return null;
        case 255:                               // Application Error
            return getErrorMessage();
        default:
            return null;
    }
}
function getpackets(){
    switch(payloadByte[1]){
        case 97:                                // if UWPA packet 2 is sent
            return getTelemetryCarlo_1P61();
        case 99:                                //if UWPA packet 1 is sent
            return getTelemetryCarlo_1P63();
        default:
        return null;
    }
}

```

FIGURE 34. Functions to handle different type packets

```
function getTelemetryCarlo_1P63(){
    var tele =get_1P_Telemetry(); // Load the telemetry model
    var mes =hexStringToBytes(payload_hex); // Convert the hex message to bytes

    if(mes[1]==99){ // Check the first byte after the application header byte
            var x = mes.slice(2,6); // Check the length of the variable from UWPA datasheet,
                                    //but remove the headerbyte --> kw L1 = 4 bytes
             for (var j = 0; j < x.length; j++) {
                    tele['kW L1']+=x[j]<<(8*(j)); // Bitwise shift operation
                }
               tele['kW L1']=tele['kW L1']*0.000001; // The UWP multiplier 0.000001
            mes.splice(1,5); //Remove the variable bytes from the original message

    }
     if(mes[1]==100){
            var x1 = mes.slice(2,6); // kVa = 4 bytes
             for (var j = 0; j < x1.length; j++) {
                    tele['kVA L1']+=x1[j]<<(8*(j));
                }
                tele['kVA L1']=tele['kVA L1']*0.000001; // The UWP multiplier 0.000001
            mes.splice(1,5); //Remove the variable bytes from the original message

    }
     if(mes[1]==101){
            var x2 = mes.slice(2,6); // kvar = 4 bytes
             for (var j = 0; j < x2.length; j++) {
                    tele['kvar L1']+=x2[j]<<(8*(j));
                }
                tele['kvar L1']=tele['kvar L1']*0.000001; // The UWP multiplier 0.000001
            mes.splice(1,5); //Remove the variable bytes from the original message

    }
     if(mes[1]==60){
            var x2 = mes.slice(2,10); // kWh = 8 bytes
             for (var j = 0; j < x2.length; j++) {
                    tele['kWh (+) TOT']+=x2[j]<<(8*(j));
                }
                tele['kWh (+) TOT']=tele['kWh (+) TOT']*0.1; // The UWP multiplier 0.1
            mes.splice(1,9); //Remove the variable bytes from the original message
    }
     if(mes[1]==62){
            var x2 = mes.slice(2,10); // kvarh = 8 bytes
             for (var j = 0; j < x2.length; j++) {
                    tele['kvarh (+) TOT']+=x2[j]<<(8*(j));
                }
                tele['kvarh (+) TOT']=tele['kvarh (+) TOT']*0.1; // The UWP multiplier 0.1
            mes.splice(1,9); //Remove the variable bytes from the original message
    }
     if(mes[1]==255){
            var x = mes.slice(2,8); // Timestamp = 6 bytes
            var d = new Date(...x); // returns 2023 as 0123
            tele['TimestampEM24(UTC)']=d;
    }

    clean(tele);//Unused telemetry headers are not returned
    return tele;
}
```

FIGURE 35. Function that decodes the hex values in first package

```
function getTelemetryCarlo_1P61(){
    var tele =get_1P_Telemetry();
    var mes =hexStringToBytes(payload_hex);
    if(mes[1]==97){
            var x = mes.slice(2,6);
             for (var j = 0; j < x.length; j++) {
                    tele['V L1-N']+=x[j]<<(8*(j));
                }
                tele['V L1-N']=tele['V L1-N']*0.1; // The UWP multiplier 0.1
            mes.splice(1,5); //Remove

    }
     if(mes[1]==98){
            var x1 = mes.slice(2,6);
             for (var j = 0; j < x1.length; j++) {
                    tele['A L1']+=x1[j]<<(8*(j));
                }
                tele['A L1']=tele['A L1']*0.001;    // The UWP multiplier 0.001
            mes.splice(1,5);    //Remove
    }
     if(mes[1]==102){
            var x2 = mes.slice(2,4);
             for (var j = 0; j < x2.length; j++) {
                    tele['PF L1']+=x2[j]<<(8*(j));
                }
                tele['PF L1']=tele['PF L1']*0.01;   // The UWP multiplier 0.01
            mes.splice(1,3);    //Remove

    }
     if(mes[1]==103){
            var x2 = mes.slice(2,4);
             for (var j = 0; j < x2.length; j++) {
                    tele['Hz']+=x2[j]<<(8*(j));
                }
                tele['Hz']=tele['Hz']*0.1;          // The UWP multiplier 0.1
            mes.splice(1,3);    //Remove
    }
     if(mes[1]==255){
            var x = mes.slice(2,8);
            var d = new Date(...x);
            tele['TimestampEM24(UTC)']=d;
    }


    clean(tele); //Unused telemetry headers are not returned
    return tele;
}
```

FIGURE 36. Function that decodes the second package

```
function getErrorMessage(){
    var error_message = payloadByte[1];
    var err;
    switch (error_message){
        case 01:
            err=constants.Error_message[1];
            break;
        case 02:
            err=constants.Error_message[2];
            break;
        case 03:
            err=constants.Error_message[3];
            break;
        case 04:
            err=constants.Error_message[4];
            break;
        default:
            err="Unknown error";
            break;
    }
    return{
        'Error' : err
    };
}
```

FIGURE 37. Function 'getErrorMessage()' handles possible error messages from the meter.

4.4.2 Dashboard

After the creation of the uplink converter code, a dashboard needed to be created. ThingsBoard dashboards are a way to visualize the telemetry data. Figure 38 shows the created dashboard and data coming to it. Dashboards are created by adding widgets to it that are linked to certain telemetry value. In Figure 38 there are six different dashboard widgets created that show the meter values on a specific time window. From the dashboard figure it is notable that the timestamp of the EM24 (UTC), which is the actual measurement time, is about a minute behind the ThingsBoard own timestamp (UTC+2). This is due to the short time it takes the data to go through the whole IoT-system. The measurement data can be viewed as a graph over time and the time window to be shown can easily be modified to get for example monthly reports of consumption.

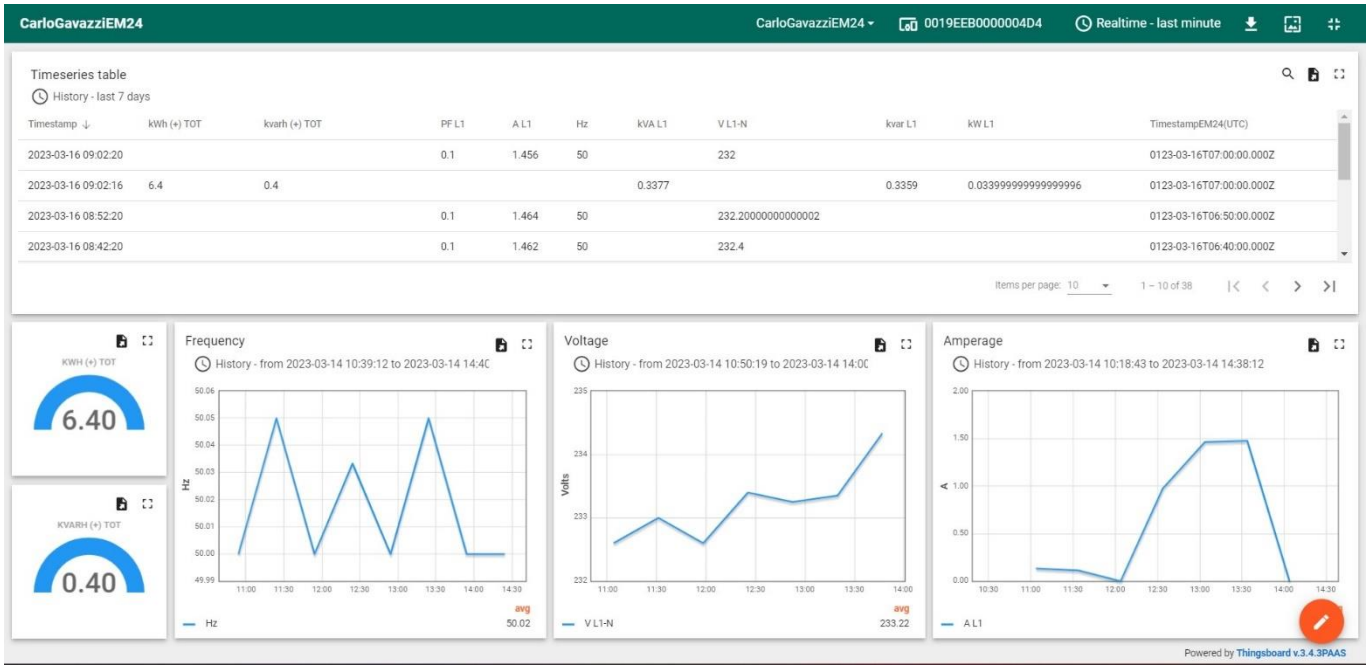

FIGURE 38. ThingsBoard dashboard created for the Carlo Gavazzi meter

4.5 Azure configuration

For the Azure connection an Azure account and subscription was needed. The subscription used for this thesis was the Azure for students' subscription, that gives students with valid email addresses free credits up to 100USD and access to Azures features. For this thesis it was decided to use the IoT Central Application as the destination for the measurement data. The steps of the dataflow from the electricity meter to Azure are shown in Figure 39.

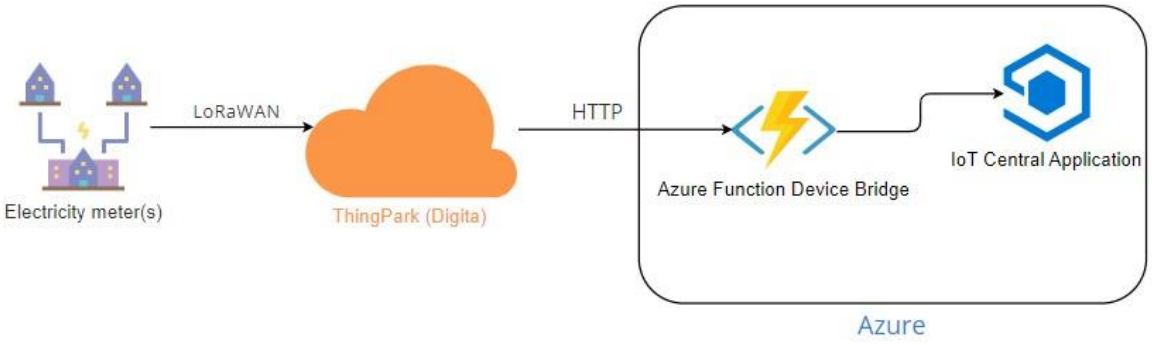

FIGURE 39. Dataflow from meter to Azure (Kinnunen 2023b)

4.5.1 IoT Central Application and The IoT Central device bridge

First step in Azure was the creation of a new resource group and then the IoT Central application in Azure portal. A resource group in Azure is a container for all the resources needed for a solution and it helps to manage the costs of the solution more easily. All the azure resources that were used in this thesis were put in the same resource group. The IoT Central application was created with the Standard Pricing 2, shown in Figure 40 as the Carlo Gavazzi UWPA sends data every ten minutes. The application was named as "carlogavazzi-lora monitor" as shown in Figure 41.

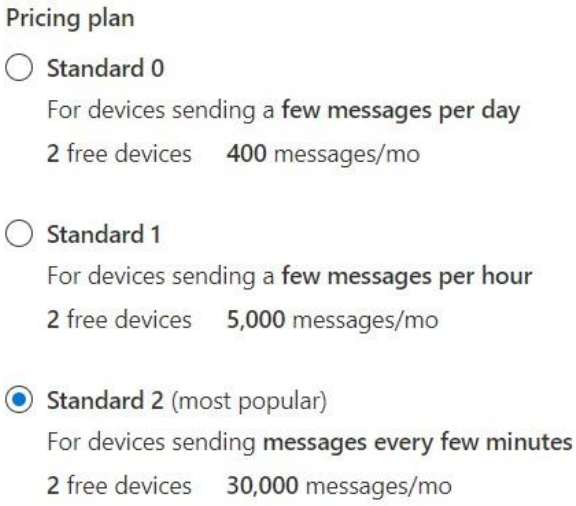

FIGURE 40. IoT Central application pricing plans

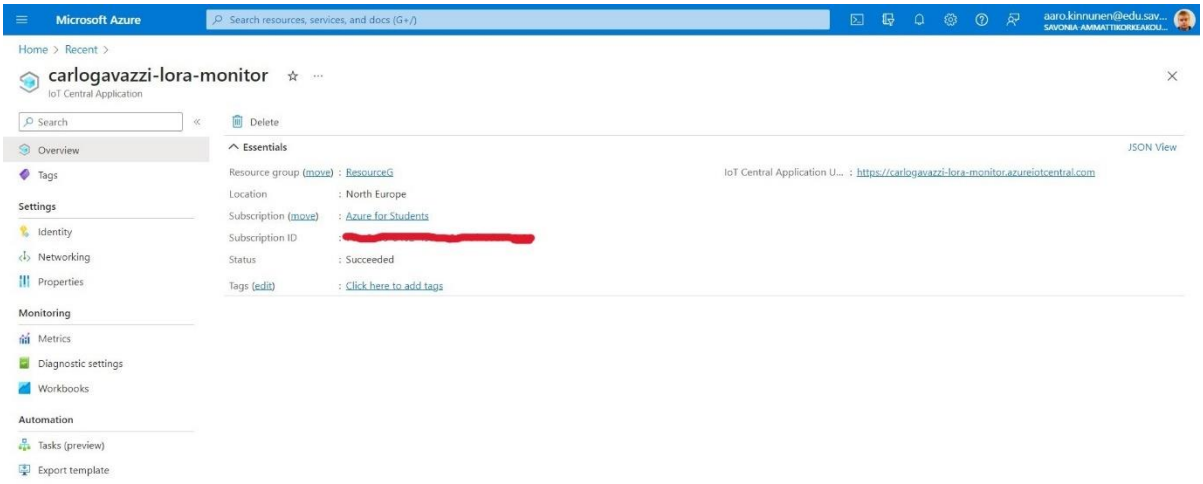

FIGURE 41. The created application in Azure portal

After creation the application was accessed through the URL shown in Figure 41. The application could also be accessed from the IoT Central page shown in Figure 42. The application was created with custom template and as shown in Figure 43 the application outlook has been modified and a picture of the Carlo Gavazzi UWPA has been added as the thumbnail of the application. The applications theme and colors were modified also.

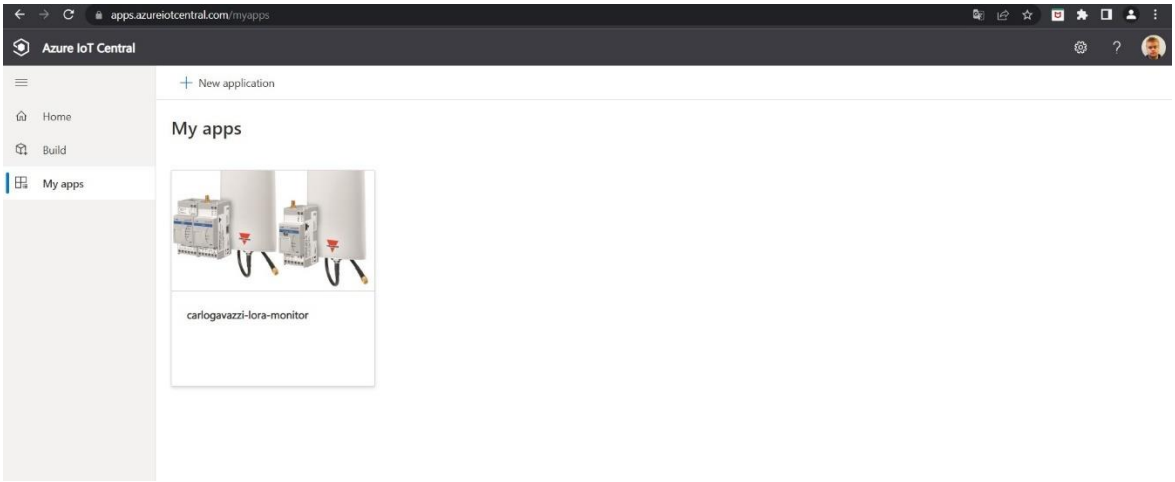

FIGURE 42. Accessing the app through Azure IoT Central page

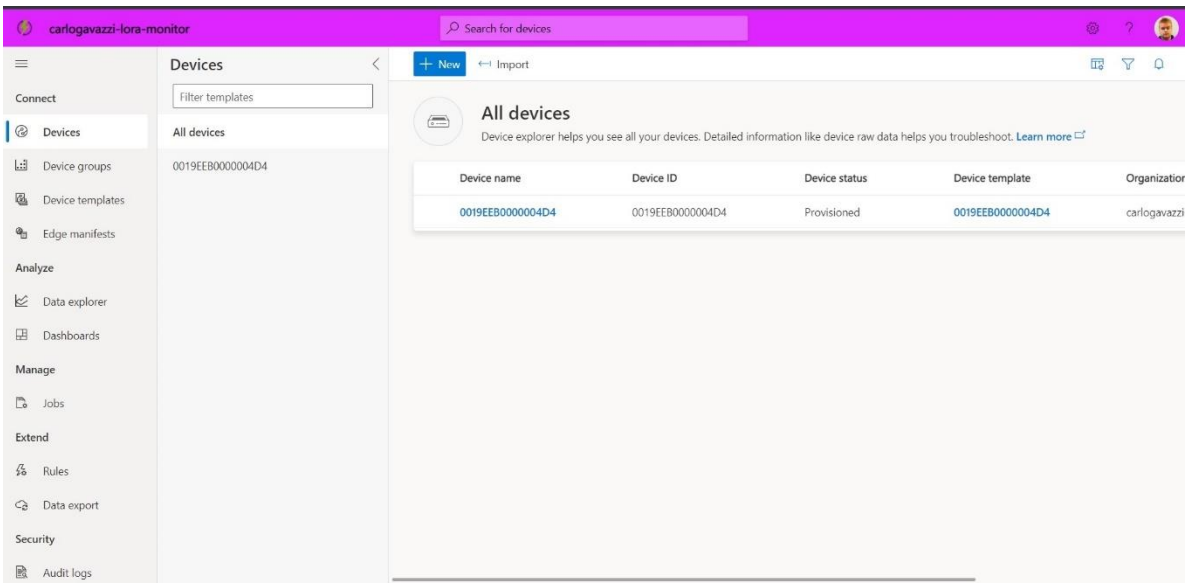

FIGURE 43. Overview of the devices in the application

To get the data to flow from Thingpark to the created IoT Central application, a kind of device bridge was needed. This was because direct HTTP connection between two cloud platforms was not supported directly by IoT Central application. The IoT Central device bridge is an open-source Azure solution that allows dataflow from different IoT clouds to IoT Central applications. The communication between clouds happens via HTTP. The device bridge solution creates several Azure resources that are automatically linked together to allow automatic dataflow from the devices to the IoT Central Application (Microsoft 2022). The device bridge consists of Azure Function App, Azure Key Vault and an Azure Storage account shown in Figure 44. These resources could also be created separately and connected manually, but The Device bridge solution does this automatically. The dataflow with the device bridge only works in one direction, meaning that the data flows only to the Central application and there is no possibility to send downlinks back to Thingpark and to the device from the application itself. The device bridge connects to the IoT central application with the Azure shared access key policy. The keys can be found from permissions page under device connection groups in the IoT central application. To deploy the device bridge, it asked the ID-scope and the primary key shown in Figure 45.

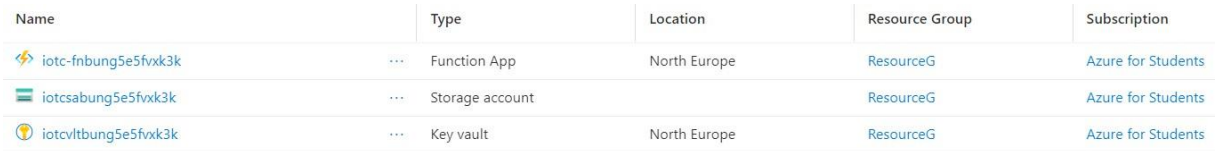

FIGURE 44. The IoT Central device bridge creates three different Azure resources.

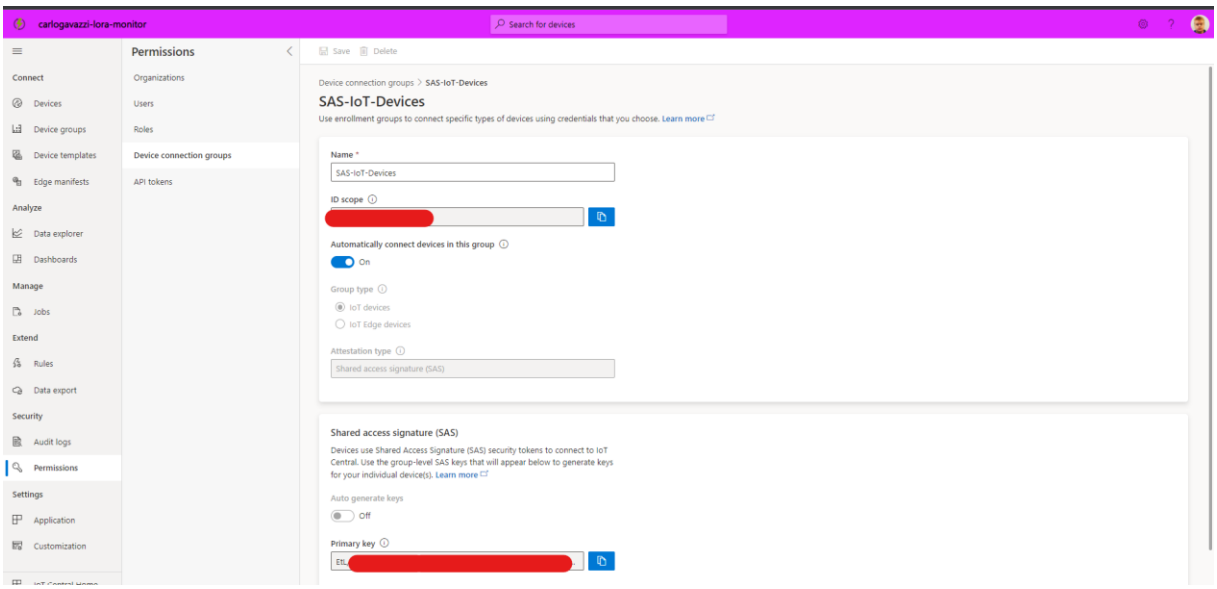

FIGURE 45. Permissions needed to link the bridge to the application

4.5.2 Function App and the HTTP trigger

The created JavaScript code described in this chapter can be found in GitHub (see Appendix 2)

The measurement data from Thingpark was sent to the created function app that has a HTTP-trigger function named "IoTCIntegration", shown in Figure 46, that handles the upcoming JSON message which Thingpark sends shown in Figure 27 earlier. The URL-endpoint of the HTTP-trigger function was added to the application server in Thingpark to allow the connection. After its deployment the function app needed NPM packages installed to it. Installing of the packets was done using Azures CLI, which is Azures own command-line tool (see Figure 47). After the packages were installed using the CLI of the function app then the HTTP triggers JavaScript code was ready to be modified. The HTTP trigger code needed to be modified a bit to convert the upcoming message to a form acceptable for the IoT central application. The Central application requires upcoming messages to come in certain JSON-body as shown in Figure 48.

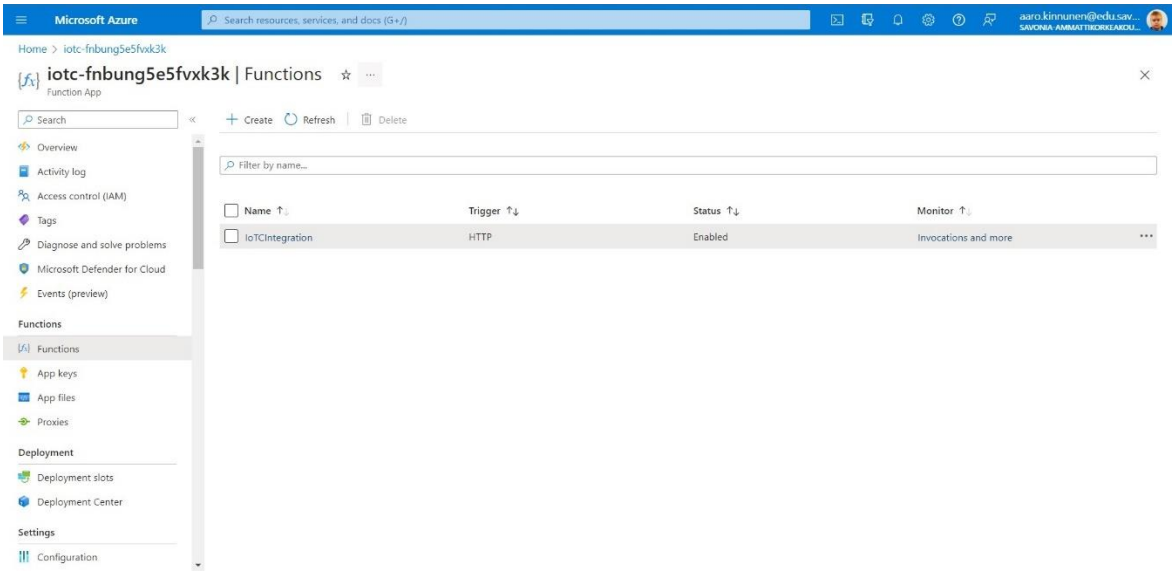

FIGURE 46. The HTTP trigger can be found inside the Function app.

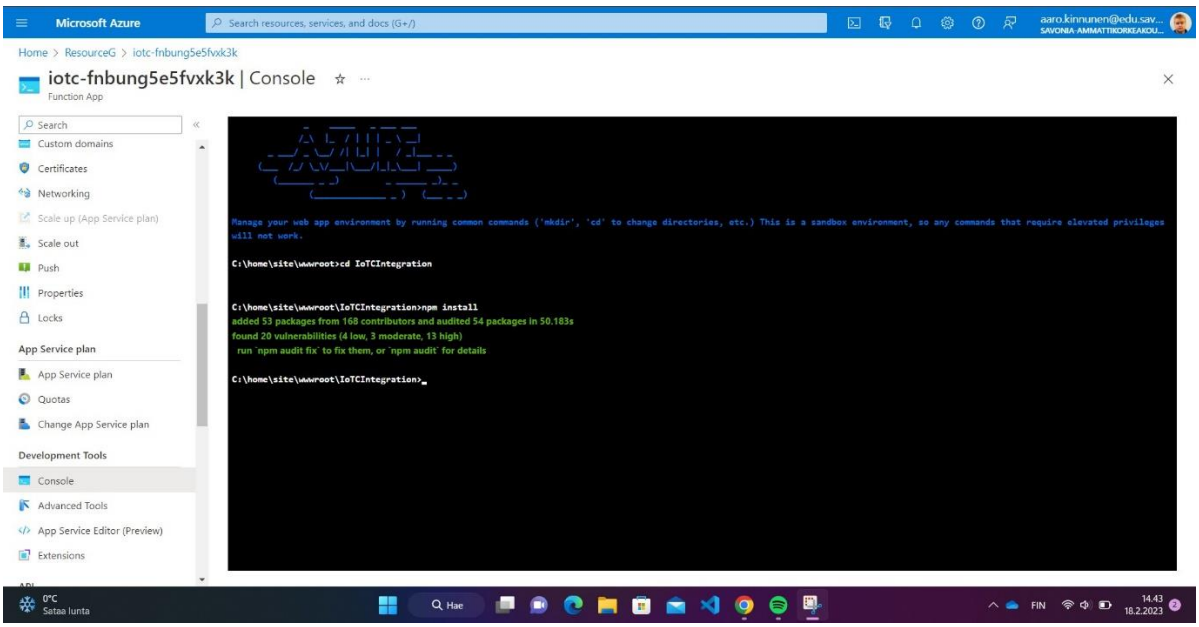

FIGURE 47.Installing packages using Azure CLI

```
req.body = {
  device: {
    deviceId: //{DeviceName or DevEUI]
  },
  measurements: // {the measurements from device}
};
```

FIGURE 48. JSON-body IoT Central application requires

The decoding of the hex message was also handled in the JavaScript code inside the HTTP trigger. The functions used to decode the hex encoded values were similar that were used in ThingsBoards uplink converter. The decoder function made for Azure was also made to handle the JSON-body change as shown in Figure 49, meaning that the decoder function returns the message in that form. The data transfer to the IoT central application is also shown in Figure 49 below.

```
489 module.exports = async function (context, req) {
490     var bod = decoder(req.body);// call the decoder function,
491                                 //(returns Headers payload: {device: {deviceId: string (DevEUI)}, measurements: {(decoded values)}})
492     try {
493         req.body = bod; // change the message body to required form
494         await handleMessage({ ...parameters, log: context.log, getSecret: getKeyVaultSecret }, req.body.device, req.body.measurements, req.body.timestamp)
495     } catch (e) {
496         context.log('[ERROR]', e.message);
497
498         context.res = {
499             status: e.statusCode ? e.statusCode : 500,
500             body: e.message
501         };
502     }
503 }
```

FIGURE 49. Part that handles the data transfer to IoT Central.

4.5.3 Dashboard in IoT Central application

When the HTTP trigger function had executed it created a device instance in the IoT Central application as seen in Figure 43. The device telemetry can be accessed from the device list. Figure 50 shows the raw telemetry data that is sent from the function app. From the raw data the telemetry needed to be modeled to a device template (see Figure 51). The device template is a model that

matches the telemetry data to IoT Centrals own variables or so-called capabilities. When the device template was created and assigned to the device then the incoming telemetry was automatically assigned to its capability. This can also be seen in the raw data page in Figure 50 where the data is already modeled.

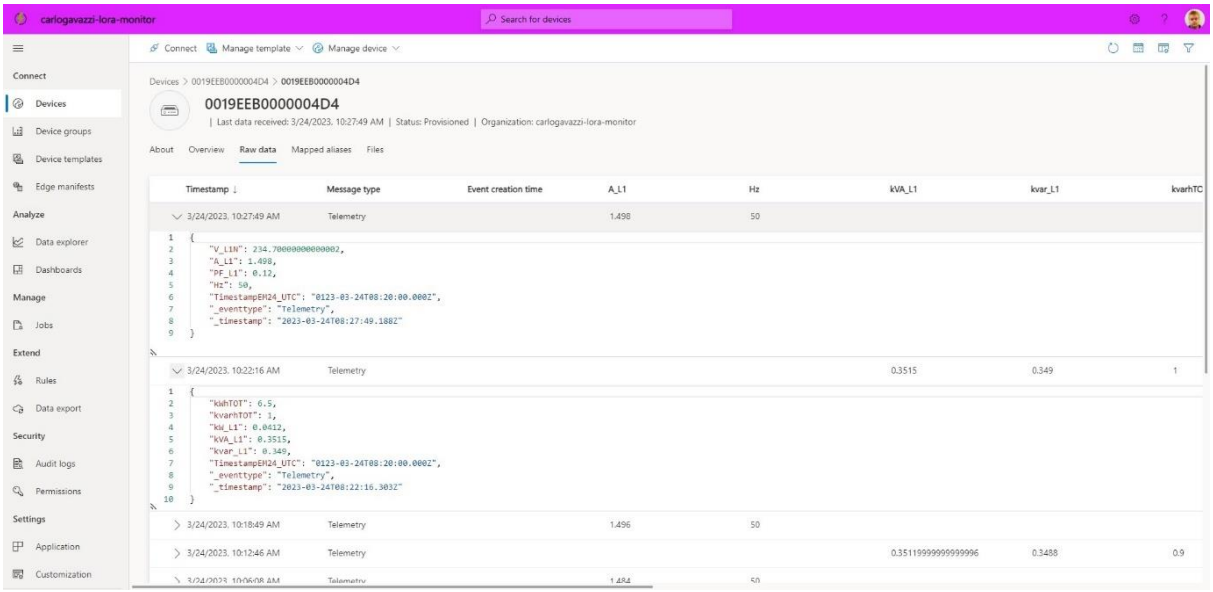

FIGURE 50. Raw data view of the device

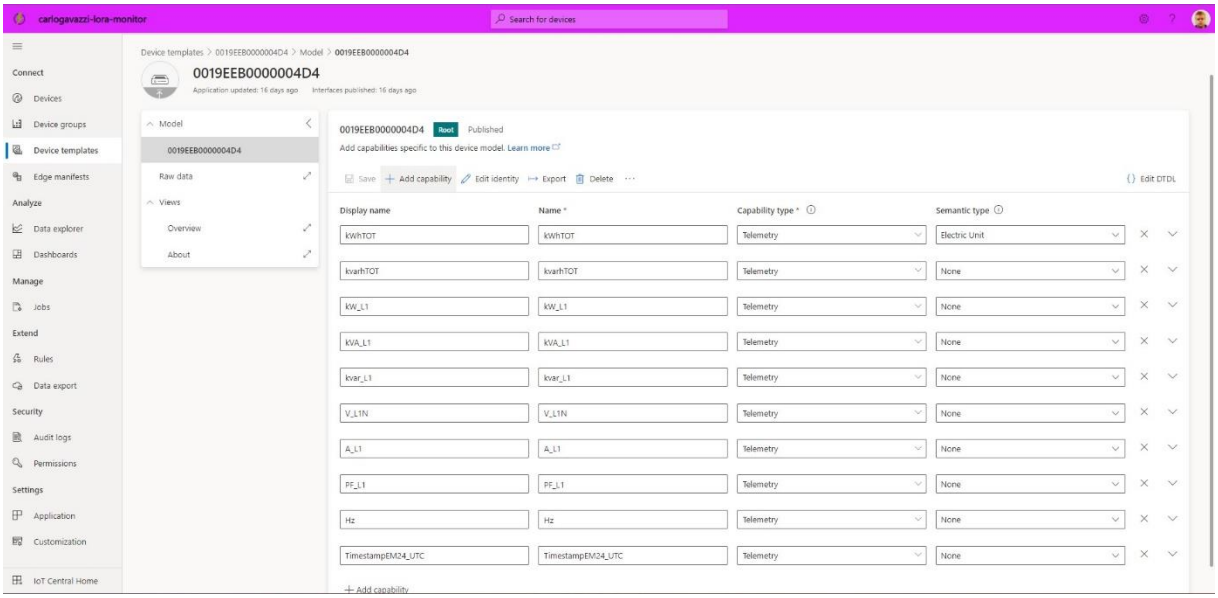

FIGURE 51. The device template created

Creating a custom dashboard for the Carlo Gavazzi meter was done from the Dashboards section of the application. The creation of dashboard tiles was similar as the creation of ThingsBoards dashboard widgets, as each tile was assigned to a capability. IoT Central application had a variety of tiles to choose from as shown in Figure 52. From the edit button of the dashboard the tiles and their capabilities could later be modified if needed. From the edit page it was also possible to add rules to tiles and their capabilities. As an example, a rule was created for voltage measurements tile where an alert is shown when voltage from the meter goes above 230 volts (see Figures 53 and 54). These kinds of rules are good way to visualize any anomalies or alerts in the meter readings.

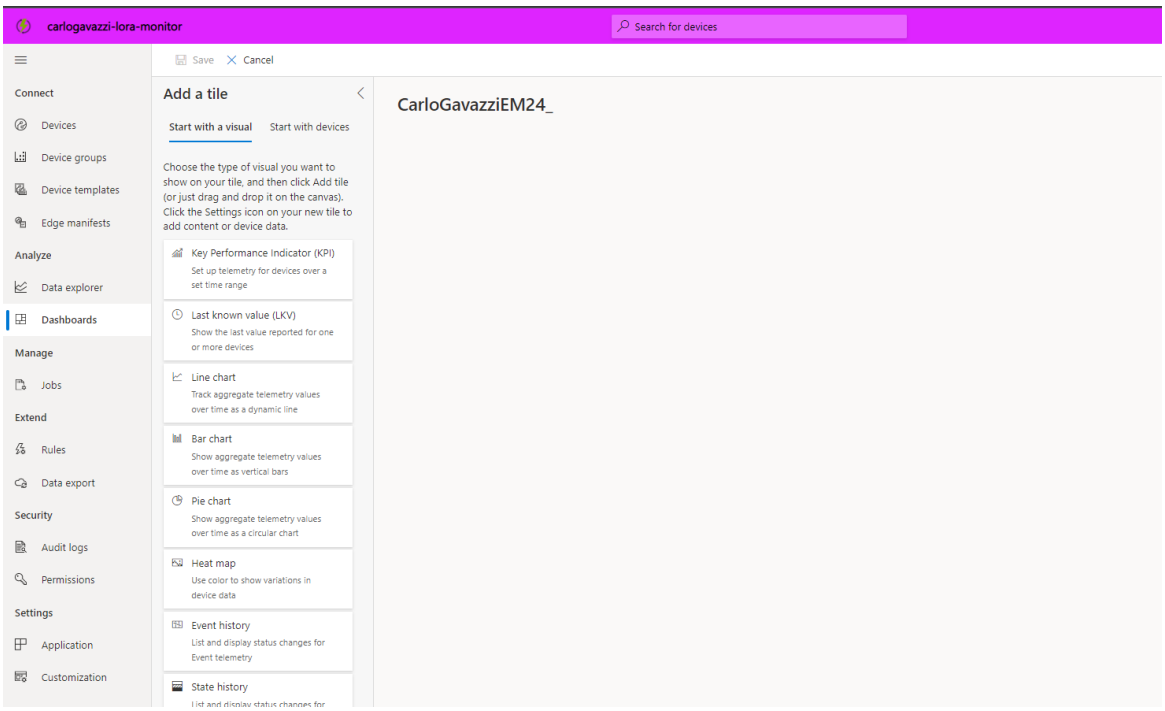

FIGURE 52. Adding tiles to a dashboard was made simple in the application.

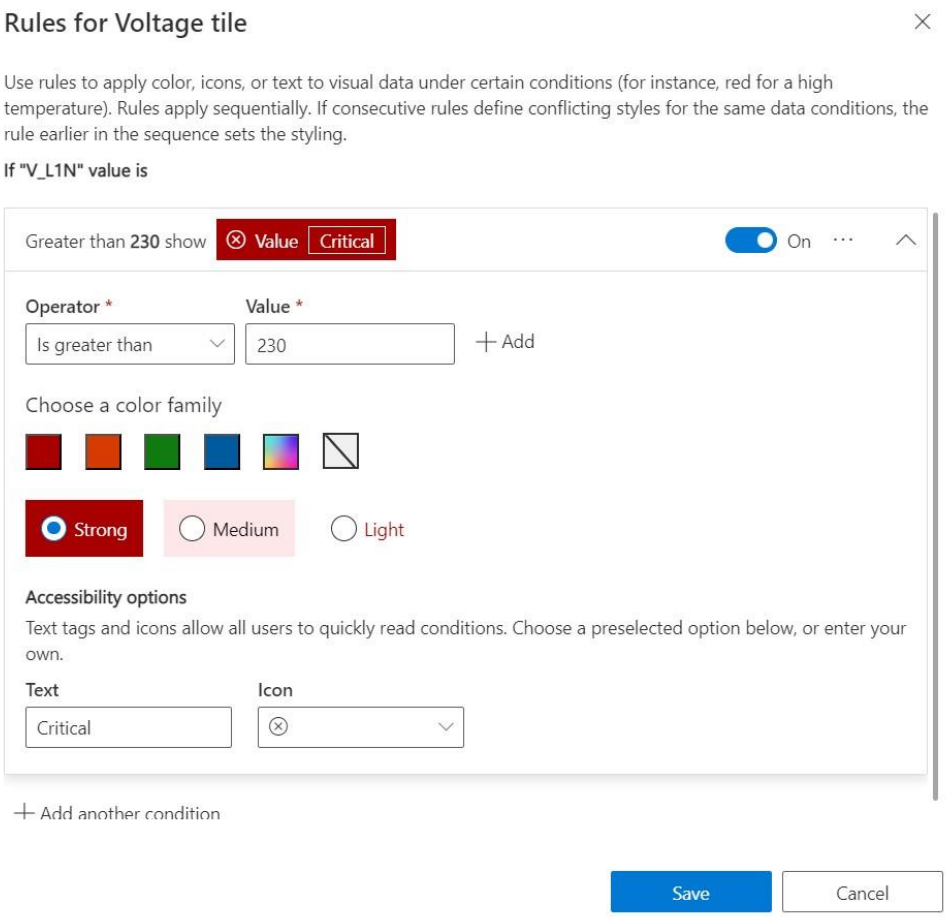

FIGURE 53. Adding example rule for voltage

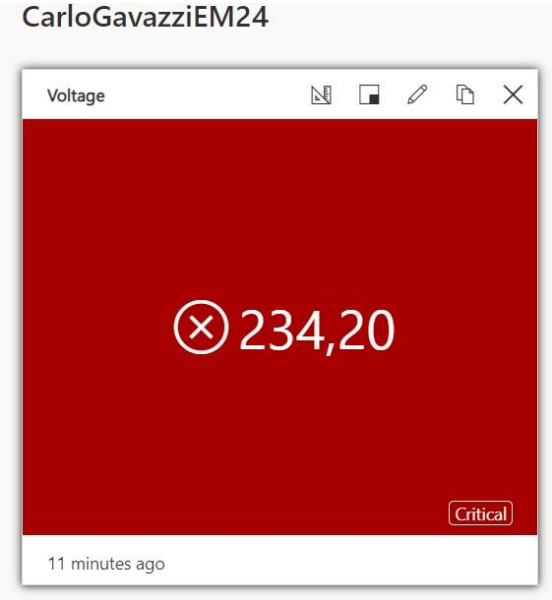

FIGURE 54. Alert in the tile if voltage is higher than wanted.

The main dashboard created for this thesis was designed to show for example the voltage and amperage in a line chart over time. The dashboard also shows all the other measurement variables in a clear way. The Carlo Gavazzi EM24 dashboard created in the application is shown below in Figure 55.

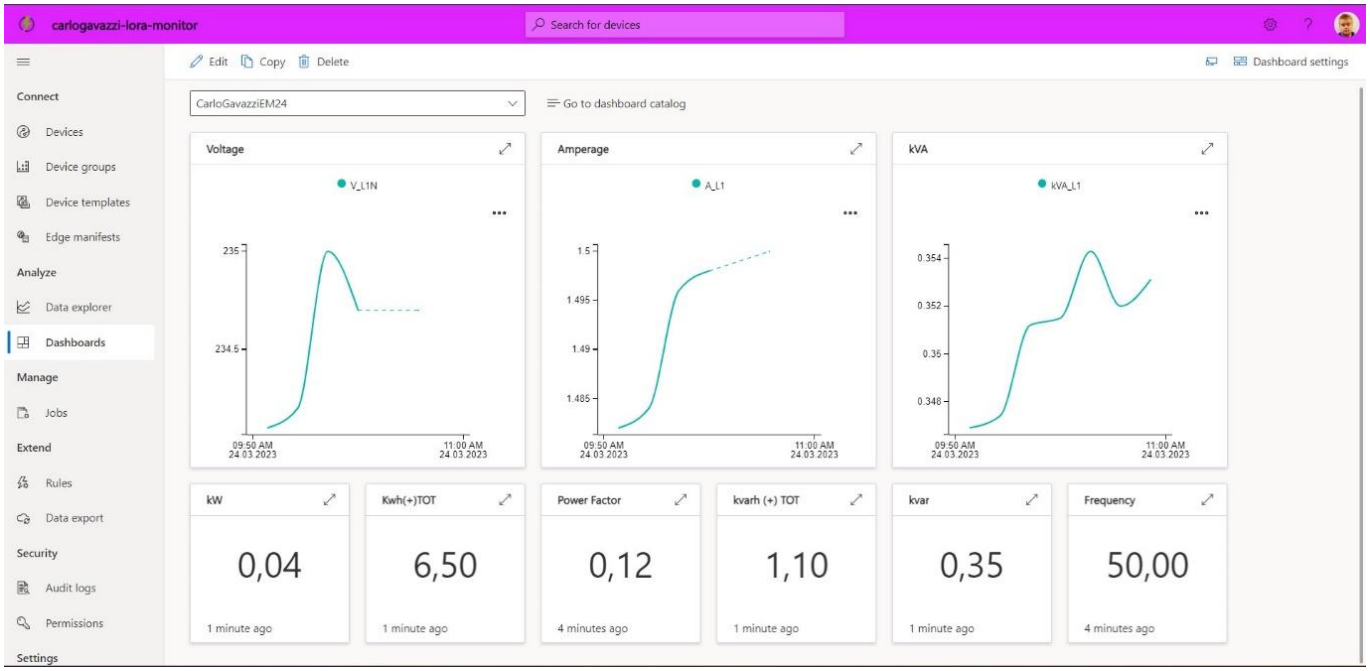

FIGURE 55. The custom Dashboard created for Carlo Gavazzi EM24.

4.6 Future implementations for the system in Azure

The commissioner of this thesis Voimatel Oy had another thesis project with Savonia student Aki Setälä, involving smart energy monitoring of underfloor heating. Setälä's thesis project uses same routing methods as described in this thesis where measurement data is sent from a device via LoRaWAN to be monitored in Azures IoT Central application. As a future implementation it would be ideal if these projects could be combined. The idea would be that data coming from the Carlo Cavazzi device and the data coming from the underfloor heating measuring device could be linked to the same IoT-central application. This would create a great and simple example of smart home

energy monitoring system. Adding both devices to the same IoT central application could be made by creating the device bridges needed for the connection to the central application with same shared access keys (see Figure 45) that lead to the same application.

In Azure there are many other resources besides the IoT Central application. For other future implementations of the systems, it would be interesting to utilize some of the resources in Azure. The Carlo Gavazzi measurement data could be exported from the central application to Azures machine learning (Azure ML) resource, where it would be possible to make prediction models of energy consumption based on the measurement data with the help of artificial intelligence (AI).

5 RESULTS AND DISCUSSION

The aim of this thesis was to study how the LoRaWAN communication could be utilized in remote electricity readings. The goal was to build and implement an electricity measuring IoT- system and design its communicational architecture that uses LoRaWAN and IoT cloud platform. As a result of this thesis an IoT-system was built that measures nine different electricity parameters from a Carlo Gavazzi EM24 meter. The system sends the measured data to Digita's LoRaWAN network from where it is then routed to IoT-platforms. Two different IoT-platforms were studied and tested as the destination for the IoT-systems data: ThingsBoard and Azure IoT Central application.

The results of this thesis show that LoRaWAN can be utilized in remote readings of electricity meters. The results also show that different IoT platforms are useful tools in the visualization of the measurement data. The results of this thesis were satisfactory, in a sense that the goal was met, and the system built worked as it was designed to. The author of this thesis would like to point out that the system was made to measure and send only certain parameters and the data communication and decoders were done to identify only these parameters. Ideally for future implementations of the system it would be better if the communicational parameters and codes were done to cover all kinds of variations of the measurement data. Also, for future implementations it would be ideal to build downlink communications from the IoT-platforms to the device to allow bi-directional dataflow, as it is becoming a standard in the field of remote meter reading. As for the comparison of the IoT platforms tested in this thesis, the author feels that the ThingsBoard cloud platform was easier to configure as it had direct option to connect to Thingpark. The dataflow to Azures IoT Central application was also a bit slower than to ThingsBoard, which was probably due to that ThingsBoard didn't need an external connector to connect to Thingpark. The author still feels that Azure would be a better solution for the system as it has more features where the data could be analyzed. For example, Azure has machine learning (Azure ML) feature that could be utilized in the future implementations of the system. The machine learning feature could allow making automated predictions based on the electricity data.

As a conclusion the author feels that, even though there were things to improve, the thesis project was a success and it helped to deepen the idea behind IoT systems.

REFERENCES

AWS. (2023). *Types of Cloud Computing*. Accessed 24. 3. 2023 at AWS website: https://aws.amazon.com/types-of-cloud-computing/

Azure, W. i. (2023). *What is Azure*. Accessed 20. 3. 2023 at Azure microsoft webpage: https://azure.microsoft.com/en-gb/resources/cloud-computing-dictionary/what-is-azure/

Azure-as-a-service. (2023). *What is IaaS?.* Accessed 24. 3. 2023 at Microsoft Azure website: https://azure.microsoft.com/en-ca/resources/cloud-computing-dictionary/what-is-iaas/

Betts, D. (2023). *What Azure technologies and services can you use to create IoT solutions?* . Accessed 24. 3. 2023 at Microsoft learn website: https://learn.microsoft.com/en-us/azure/iot-fundamentals/iot-services-and-technologies

Burges, M. (16. 2. 2018). *What is the Internet of Things? WIRED explains.* Accessed 23. 3. 2023 at Wired webpage: https://www.wired.co.uk/article/internet-of-things-what-is-explained-iot

Digita, A. (2015). *Digita Anouncement*. Accessed 10. 3. 2023 at Digita webpage: https://www.digita.fi/ajankohtaista/actility-ja-lahetysverkko-operaattori-digita-ottaneet-koekayttoon-lora-verkon/

Digita-jyväskylä. (2022). *Pamflet at Jyväskylä (Valo ja Tele av) fair.*

Digita-LoRa. (2023). *Digita website about LoRaWAN*. Accessed 10. 3. 2023 at Digita website: https://www.digita.fi/etusivu/palvelut-yrityksille/digitan-iot-palvelut/lorawan-teknologia/

GavazziEM24. (18. 11. 2020). *Datasheet for Carlo Gavazzi EM24 meter*. Accessed 10. 3. 2023 at GavazziOnline website: https://www.gavazzionline.com/pdf/EM24DINDS.pdf

GavazziUWPA. (10. 3. 2022). *Datasheet for Carlo Gavazzi UWPA*. Accessed 10. 3. 2023 at GavazziAutomation website: https://www.gavazziautomation.com/images/PIM/DATASHEET/ENG/UWPA_UWPM_DS_ENG.pdf

Gerber, A. (6. 8. 2017). *Simplify the development of your IoT solutions with IoT architectures*. Accessed 23. 3. 2023 at IBM developer website: https://developer.ibm.com/articles/iot-lp201-iot-architectures/

Haas, J. (30. 9. 2020). *NB-IoT ja LTE-M -teknologioiden hyödyt etäluennassa*. Accessed 28. 3. 2023 at Landis+Gyr website: https://eu.landisgyr.com/blog-fi/nb-iot-ja-lte-m-teknologioiden-hy%C3%B6dyt-et%C3%A4luennassa

IBM_X-Force. (2014). *Figure from an artice in IBM webpage*. Accessed 23. 3. 2023 at IBM developers webpage: https://developer.ibm.com/articles/iot-lp201-iot-architectures/

IoTCentralArchitecture. (22. 2. 2023). *Azure IoT Central architecture*. Accessed 24. 3. 2023 at Microsoft learn website: https://learn.microsoft.com/en-us/azure/iot-central/core/concepts-architecture

IoTCentralPricing. (2023). *Azure IoT Central pricing*. Accessed 24. 3. 2023 at Microsoft azure webpage: https://azure.microsoft.com/en-us/pricing/details/iot-central/

IoT-map. (2023). *Figure of digitas LoraWAN coverage*. Accessed 22. 3. 2023 at Digitas website: https://www.digita.fi/iotn-kartta/

Kinnunen, A. (2022, CC BY-NC). Photograph 8.9.2022. *Digita IoT gateway at the Jyväskylä fair.* Jyväskylä.

Kinnunen, A. (2023a, CC BY-NC). Photograph 24.3.2023. *Carlo Gavazzi measurement system.* Savonia UAS, Kuopio.

Kinnunen, A. (2023b, CC BY-NC). Illustration 1.3.2023. *Dataflow from meter to Azure IOTC.*

Kuivasmäki, U. (1. 3. 2023). *Vaasan Sähkö vaihtaa mittareita – 66 000 asiakasta saa älykkään mittarin*. (YLE) Accessed 28. 3. 2023 at YLE website: https://yle.fi/a/74-20020312

Kurunsaari, S. (3. 10. 2018). *Landis+Gyr ja Telia tuovat NB-IoT -teknologian etäluentaan*. Accessed 25. 3. 2023 at STTinfo website: https://www.sttinfo.fi/tiedote/landisgyr-ja-telia-tuovat-nb-iot--teknologian-etaluentaan?publisherId=69691516&releaseId=69843488

Laitinen, T. (18. 1. 2023). *PKS Sähkönsiirto mukana suuressa sähkömittareiden ja palveluteknologian kaupassa*. (YLE) Accessed 28. 3. 2023 at Yle website: https://yle.fi/a/74-20013522

Leskinen, M. (26. 9. 2019). *Mitä tapahtui esineiden internetille, josta piti tulla seuraava iso mullistus?* (YLE). Accessed 28. 3. 2023 at Yle website: https://yle.fi/a/3-10985702

Lora-alliance. (2023). *Lora-alliance, about LoRaWAN*. Accessed 10. 3. 2023 at Lora-alliance website: https://lora-alliance.org/about-lorawan/

Lora-developers, S. (2023). *LoRa developer portal about LoRa*. Accessed 22. 3. 2023 at Semtech.com: https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/

LoRaWorkgroup, L. A. (2018). *LoRaWAN 1.0.3 Regional Parameters*. Accessed 10. 3. 2023 at Lora alliance pdf: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_regional_parameters_v1.0.3reva_0.pdf

Lorrain, R. (21. 3. 2022). *10 Things About LoRaWAN & NB-IoT*. Accessed 29. 3. 2023 at Semtech blog website: https://blog.semtech.com/title-10-things-about-lorawan-nb-iot

Microsoft. (24. 6. 2022). *IoT Central device bridge Instructions*. Accessed 10. 3. 2023 at Micsoft webpage: https://learn.microsoft.com/en-us/azure/iot-central/core/howto-build-iotc-device-bridge

Microsoft_IoTCentral. (12. 9. 2022). *What is Azure IoT Central?* . Accessed 24. 3. 2023 at Microsoft learn website: https://learn.microsoft.com/en-us/azure/iot-central/core/overview-iot-central

Miettinen, M. (2023). *Tuntimittausta ja etäluentaa*. Accessed 25. 3. 2023 at Energia.fi website: https://energia.fi/energiasta/asiakkaat/sahkoasiakkuus/sahkon_mittaus

Mokolora. (2021). *Comparison between LoRa and other wireless technologies*. Accessed 3. 23, 2023 at Mokolora webpage: https://www.mokolora.com/lora-and-wireless-technologies/

Rana, D. (19. 5. 2022). *Top 11 Cloud Platforms for Internet of Things (IoT)*. Presentation. Accessed 24. 3. 2023 at DZone website: https://dzone.com/articles/10-cloud-platforms-for-internet-of-things-iot

Rouse, M. (12. 1. 2017). *Wireless Personal Area Network*. Accessed 24. 3. 2023 at Techopedia website: https://www.techopedia.com/definition/5109/wireless-personal-area-network-wpan

Semtech. (2023). *What is LoRaWAN*. Accessed 20. 3. 2023 at Semtech webpage: https://www.semtech.com/lora/lorawan-standard

SemtechLoRa-applications. (2023). *Semtech LoRa applications*. Accessed 10. 3. 2023 at Semtech webpage: https://www.semtech.com/lora/lora-applications

Simoneau, P. (2006). *What is OSI*. PDF. Accessed 23. 3. 2023 at Global Knowledge: https://faculty.sfcc.spokane.edu/Rudlock/files/WP_Simoneau_OSIModel.pdf

Slats, L. (8. 1. 2020). *A Brief History of LoRa®: Three Inventors Share Their Personal Story at The Things Conference*. Accessed 20. 3. 2023 at Semtech blog page: https://blog.semtech.com/a-brief-history-of-lora-three-inventors-share-their-personal-story-at-the-things-conference

TheThingsNetwork. (2023). *Device Classes*. Accessed 10. 3. 2023 at The Things network website: https://www.thethingsnetwork.org/docs/lorawan/classes/

ThingIndustries. (2023). *ABP vs OTAA*. Accessed 19. 3. 2023 at ThingIndustries webpage: https://www.thethingsindustries.com/docs/devices/abp-vs-otaa/

ThingsboardAuthors. (2023). *Thingsboard monolithic architecture*. Accessed 24. 3. 2023 at Thingsboard docs website: https://thingsboard.io/docs/reference/monolithic/

ThingsBoardDevelopers. (2023). *What is ThingsBoard*. Accessed 10. 3. 2023 at Thingsboard.io website https://thingsboard.io/docs/getting-started-guides/what-is-thingsboard/

ThingsboardIntegrations. (2023). *User guide to Integrations*. Accessed 10. 3. 2023 at Thingsboard.io website https://thingsboard.io/docs/user-guide/integrations/

ThingsBoardPricing. (2023). *Thingsboard pricing*. Accessed 10. 3. 2023 at Thingsboard webpage: https://thingsboard.io/pricing/

Thingsboard-ThingParkIntegration. (2023). *Thingpark integration*. Accessed 10. 3. 2023 at Thingsboard website https://thingsboard.io/docs/paas/user-guide/integrations/thingpark/

TurkuEnergia. (2023). *Etäluennan tiedonsiirtotekniikat ja terveysvaikutukset*. Accessed 24. 3. 2023 at Turku energia website: https://www.turkuenergia.fi/sahkoverkot/sahkoliittyma-ja-sahkon-mittaus/sahkon-mittaus/etaluennan-tiedonsiirtotekniikat-ja-terveysvaikutukset/

VantaanEnergia. (2023). *Kotiautomaatioliitäntä eli HAN-portti*. Accessed 29. 3. 2023 at vantaan energia website: https://www.vantaanenergiasahkoverkot.fi/palvelut/tietoa-mittausuudistuksesta/kotiautomaatioliitanta-eli-han-portti/

Vasanth, K. R. (17. 2. 2023). *Top 7 Cloud Platforms for IoT*. Accessed 24. 3. 2023 at CloudThat website: https://www.cloudthat.com/resources/blog/top-7-cloud-platforms-for-iot

Veeru. (2021). *Lora & Lorawan explained*. Accessed 10. 3. 2023 at ElectronicsInnovation webpage: https://electronicsinnovation.com/lora-lorawan-explained-what-is-lora-history-of-lora-who-invented-lora-how-lora-works/

Verma, E. P. (23. 2. 2016). *Types of Wireless networking technology and Comparison*. Accessed 24. 3. 2023 at Yuvayana, tech & graft website: https://er.yuvayana.org/types-of-wireless-networking-technology-and-comparison/

WirelessIoT. (2016). *Wireless IoT connectivity technologies.* Accessed 24. 3. 2023 at Scirp website: https://www.scirp.org/journal/paperinformation.aspx?paperid=65802

Zhang, M. (1. 1. 2023).Article: *Top 10 Cloud Service Providers Globally in 2023*. Accessed 24. 3. 2023 at DgtlInfra website: https://dgtlinfra.com/top-10-cloud-service-providers-2022/

APPENDIX 1: THINGSBOARD UPLINK CONVERTER CODE

The ThingsBoard code can be found on GitHub: https://github.com/AaroKinnunen/Things-Board_ThingPark_uplinkConverter

APPENDIX 2: AZURE IOTC FUNCTION APP CODE

The code for IoT Central application device bridge can be found on GitHub: https://github.com/AaroKinnunen/Azure-IoTCentralDeviceBridge-HttpTrigger