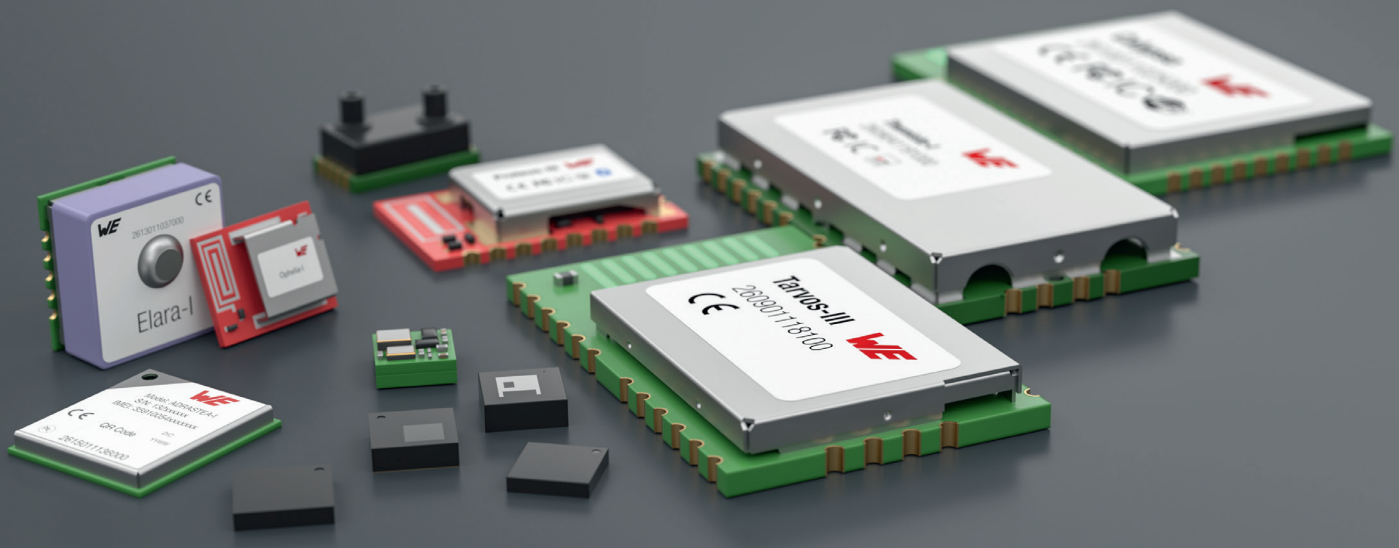

ANR023

Secure cloud connectivity using Calypso Wi-Fi module

Version 1.1

July 19, 2023

WÜRTH ELEKTRONIK MORE THAN YOU EXPECT

Revision history

Document version	Notes	Date
1.0	• Initial version	June 2021
1.1	• Updated `Important notes`, meta data and document style	July 2023

Contents

1 IoT application

1.1 Introduction

The Internet of Things (IoT) can be broadly defined as an umbrella term for a range of technologies that enable devices to connect and interact with each other. Interconnected devices generating data provide for a range of new applications. Industrial automation, healthcare, smart home, smart cities, smart grids and smart farming are some of the areas in which IoT provides substantial benefits.

Dubbed the "fourth industrial revolution" or Industry 4.0, the Industrial IoT (IIoT) is the digitization of industrial assets and processes that connects products, machines, services, locations/sites to workers, managers, suppliers, and partners. Closer networking of the digital world with the physical world of machines helps achieve higher productivity, safety, efficiency and sustainability.

The core task of any IoT solution is to get data from the field to the cloud where analysis of the same generates the desired value addition for the application. This application note aims to propose an elegant solution to achieve this task based on Calypso Wi-Fi module.

This chapter begins by describing the parts of a typical IoT application. Further, an application scenario is discussed with an example. Finally, a sensor-to-cloud IoT solution is presented using tools from Würth Elektronik eiSos. Specifically, a step-by-step description is presented to get sensor data over Wi-Fi into the most popularly used cloud platforms like Amazon AWS and the Microsoft Azure.

1.2 The IoT stack

Irrespective of the area of application, an end-to-end IoT solution consists of the following components (figure 1).

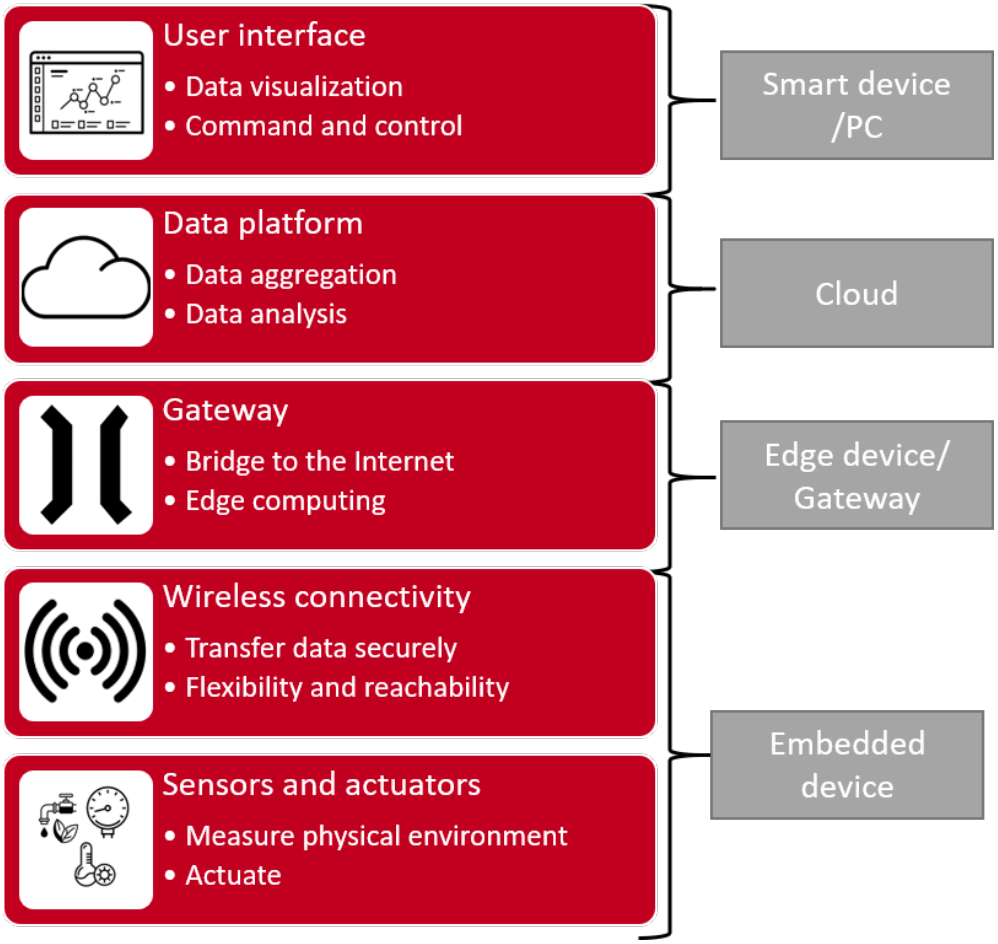

Figure 1: IoT application stack

- **Sensors and actuators:** This is a part of the system that directly interfaces with the physical environment. Sensors measure the state of the environment and interpret the same as analog or digital data. On the other hand, actuators activate a physical change in the measured environment. Advances in the field of electronics in general and semiconductors in particular has led to availability of a wide range of sensors and actuators which are highly efficient and yet very compact.
- **Wireless connectivity:** Sensors and actuators are typically installed in devices with limited access to the digital world. Consider, for example, a temperature sensor mounted inside an industrial boiler. Wireless connectivity provides in addition to many advantages the reachability necessary for such applications. A wide variety of standards as well proprietary wireless connectivity solutions are available today. A number of factors including range, throughput, spectrum regulations, local statutory requirements and power budget determine the choice of wireless connectivity solution.

  Modern embedded designs usually combine the above components into a single device (node) interacting with a gateway.
- **Gateway device:** A gateway device acts as a bridge between the physical and the digital worlds. It interprets the multitude of wireless connectivity protocols, collects the data and

forwards the same in a format understood by entities above. In certain applications, the gateway device also performs basic analytics like threshold detection.

- **Data platform:** This is the platform where the data is finally stored and presented for further analysis. Options here can range from a local database to a cloud server with redundancies. A typical platform consists of the components as shown in figure 2. The data platform enables the use of technologies like Artificial Intelligence (AI) and Machine Learning (ML) to perform advanced data analytics that generates value additions to the application.

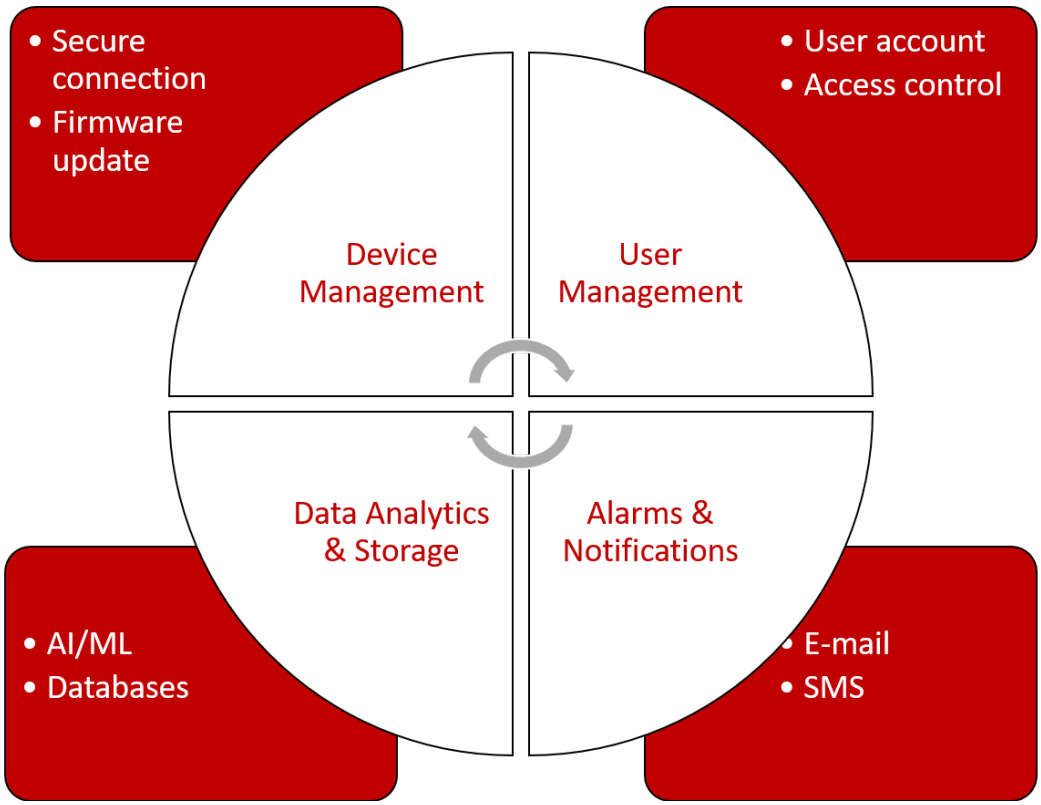

Figure 2: Components of a data platform

- **User interface:** This is the interface between the human users and the digital world. Here the status of the observed environment is presented in a human readable format. The user can take the necessary actions by interacting with this application

1.3 IIoT application example

The IIoT creates a universe of sensors that enables an accelerated deep learning of existing operations that allows rapid contextualization, automatic pattern, and trend detection. This leads to true quantitative capture of qualitative operations resulting in better quality, efficiency, higher safety, lower costs and several other benefits. This has led to the use of IoT in several application use cases.

Remote monitoring and control of production/storage environment (temperature, humidity, pressure etc.) is one of the essential tasks in the industry. Maintaining optimal conditions and

automating this process has been a challenge. This use case is considered in this application note and an IoT based approach to solve this problem is presented here.
The above-mentioned task requires sensors/actuators to interact with the environment, a wireless connectivity method, a gateway to collect the data, a cloud platform to store data and a user interface to enable human interaction. The architecture of such a system is as shown in figure 3.

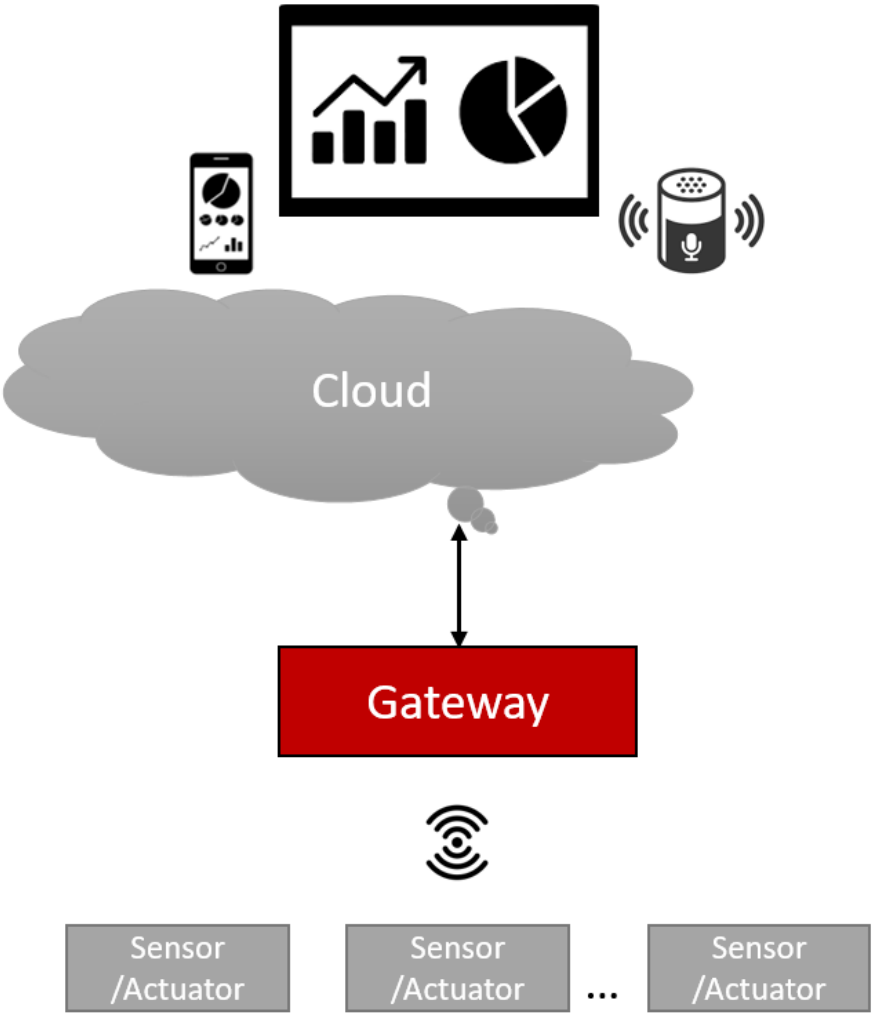

Figure 3: System architecture

1.4 System design using Calypso and cloud platforms

Figure 4 illustrates the design that realizes the architecture described in 1.3. In this example, sensor data is transferred to the cloud platforms using the Wi-Fi connectivity where it is stored and processed.

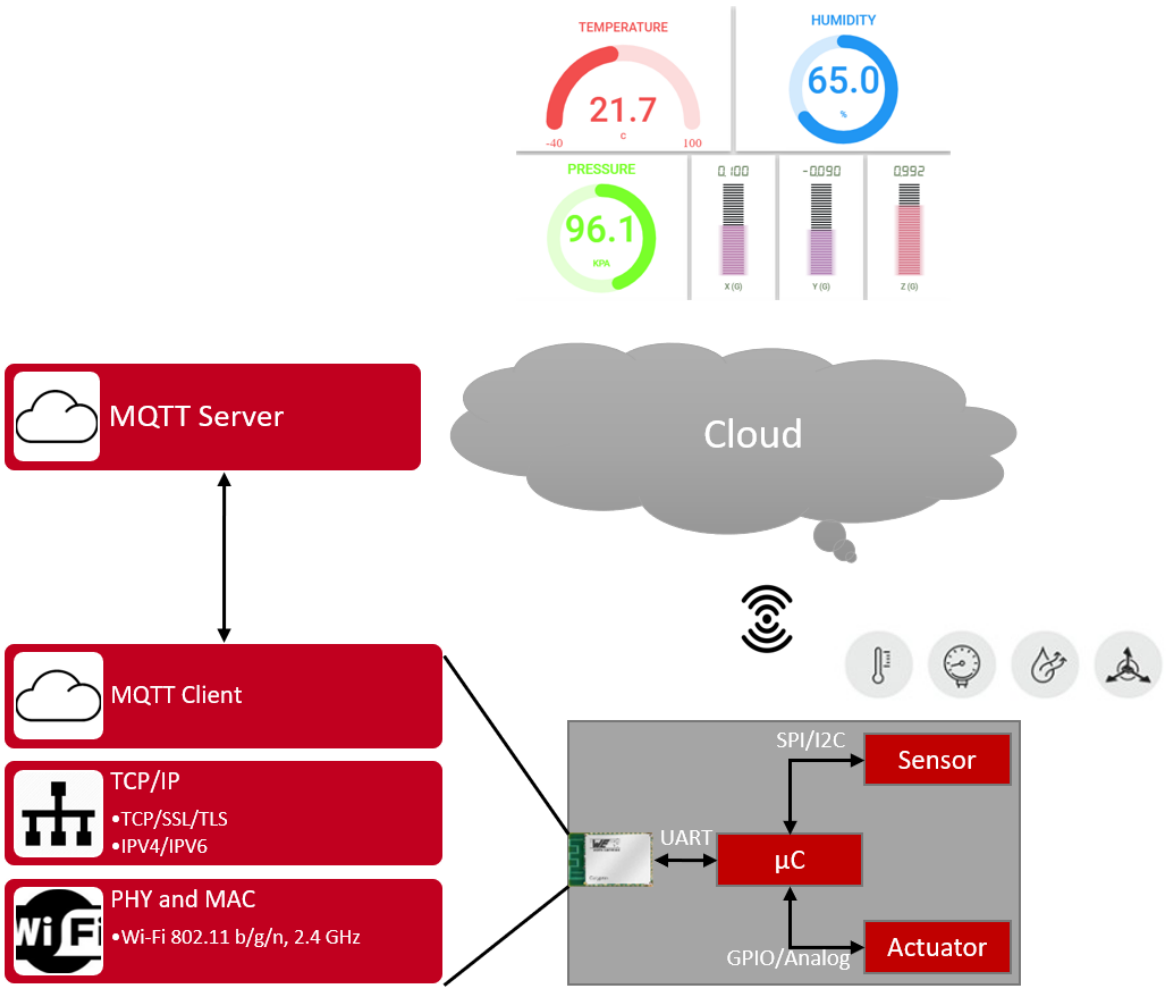

Figure 4: System design

1.4.1 Embedded design

The Calypso is a compact Wi-Fi radio module based on IEEE 802.11 b/g/n (2.4 GHz). With an on-board TCP/IP stack and MQTT protocol implemented out-of-the-box, Calypso acts as the perfect building block for an IoT application.
At the hardware level, a host microcontroller connects and controls sensors/actuators over standard interfaces like I2C, SPI, GPIO or analog. The Calypso uses UART as the primary interface to the Host. Being Wi-Fi compliant, the module can be configured to connect to the local infrastructure network (via an Access Point).

Further, the MQTT-client implemented on-board the Calypso can be configured to connect to a broker/server running in the network. Most of the commonly used cloud platforms today use a type of secure MQTT protocol to connect devices and exchange data. Hence, Calypso provides a direct communication link to the cloud without the need of a gateway device in between.
Here are a few advantages provided by this design approach:

- **Easy hardware integration:** Calypso's compact form factor, edge castellated connection and smart antenna selection allows easy integration into any hardware design. The smart antenna configuration enables the user to use either the on-board PCB antenna or an

external antenna. With UART as the standard interface, the Calypso can be interfaced with most of the standard host microcontrollers.

- **Connection to existing network:** Wi-Fi is one of the most commonly used wireless connectivity technologies. Today, most homes already have Wi-Fi infrastructure in use. Hence, integration into the existing wireless network is easier and does not require an extra bridge device.
- **Provisioning into the network:** Any device on deployment needs to be configured to connect to the local network. The Calypso offers a provisioning mode, which allows the user to connect via a smartphone/tablet and configure the device via web pages running on-board.
- **Easy software integration:** Calypso comes with an intuitive AT-style command interface over UART. This allows the host microcontroller to send ASCII based commands to the module to initiate the necessary actions. Additionally, a fully featured TCP/IP stack with several network applications implemented allows the host controller to delegate the complete network handling to the Calypso module.
- **Low-power operation:** Often, the devices deployed in IoT applications are battery operated. Calypso's low power mode allows the module to consume very less current (< 10 μA) when the device is not in use. However, features like fast connect and auto connect ensure that the module is up and running in a very short time after wake-up.
- **Security:** One of the major challenges in designing any IoT application is security. Calypso deals with this challenge at several levels. The module itself has a secure bootloader implemented to detect firmware tampering. Wi-Fi compliance ensures conformity to standard Wi-Fi security feature like WPA2 and WPA2 enterprise. Further, at the transport layer, TLS allows authentication as well as end-to-end encryption of data. Any modern application requires secure storage to store sensitive information like credentials, certificate-key pairs etc. Calypso also has an encrypted file system with limited space to enable secure storage. Thus, Calypso provides a very good basis for secure IoT application development.

1.4.2 Cloud platform design

Hundreds of cloud platforms exists in the market today. In this example, the two of the most popular platforms, Amazon Web Services (AWS) and Microsoft Azure are considered. Most cloud platforms are made up of cloud services, which offer specialized functionalities like,

- Device management
- User management
- Data storage
- Data streaming
- Data visualization
- AI/ML

- Security
- Networking
- Billing and cost management

These services act as building blocks for any IoT application. The challenge here is to pick and choose the right services and combine them into a secure, flexible and scalable application that serves the desired purpose.
In this application note, two such combinations are considered keeping in mind the most frequently occurring use cases.

- **Amazon AWS:** Figure 5 shows a sample application using Amazon AWS. In this example, the devices are provisioned to connect securely to the Amazon AWS IoT core. The incoming data is than processed using the AWS lambda function. Using the thresholds stored in the Dynamo DB, the lambda function detects the thresholds and sends notifications to the user using the notification service (Amazon SNS). Further, the data is streamed to an S3 bucket using the Kinesis Firehose streaming service. The stored data can be used for visualization using Amazon Quicksight service.

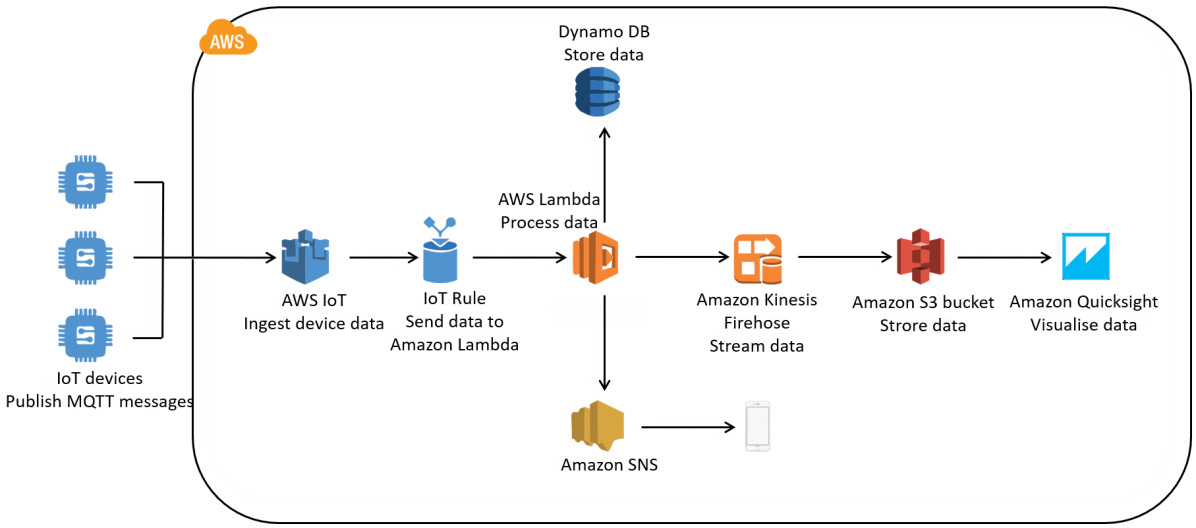

Figure 5: AWS example

- **Microsoft Azure:** Microsoft's Azure IoT hub provides the device management necessary to securely connect IoT devices to the cloud. Once the data is in the cloud it can follow one or a combination of the following paths,
  - Streamed into PowerBI for analytics using the stream analytics service
  - Visualized using a web application
  - Stored in an SQL database and processed using Excel

  Figure 6 shows a sample application using Microsoft Azure.

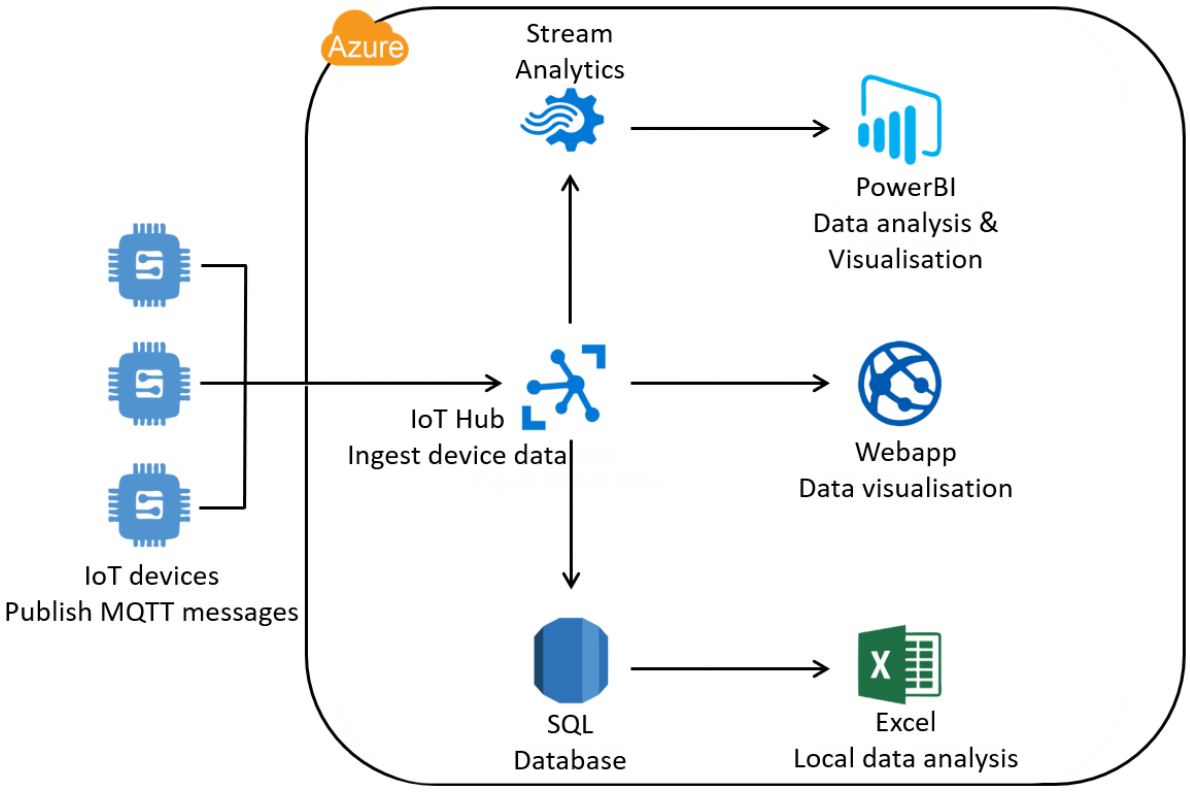

Figure 6: Azure example

1.5 Building a prototype application

Rest of this application note is intended to describe the process of building a prototype for an end-to-end IIoT solution (from sensor to cloud) using tools from Würth Elektronik eiSos. Chapter 2 gives a brief introduction to the MQTT protocol and chapter 3 provides a step-by-step description for building a prototype IoT application including the hardware, firmware and the cloud components.

2 MQTT

MQTT - Message Queuing Telemetry Transport is a lightweight application layer protocol based on a publish/subscribe messaging mechanism. This protocol was designed for resource constrained and unreliable networks with limited bandwidth and high latency. These characteristics make MQTT suitable for low-power, low-bandwidth IoT applications. Inherently, the MQTT protocol offers some degree of assurance of delivery thereby offering the robustness necessary for industrial machine-to-machine communication.

MQTT was originally developed by IBM and the version 3.1.1 is an OASIS standard that is open and royalty-free [3]. It is based on client-server architecture where every client connects to a server (broker) over TCP resulting in a star topology. Once connected, the clients send and receive messages using the publish/subscribe mechanism.

2.1 Publish/Subscribe

Data transfer in MQTT takes place based on publish/subscribe mechanism. The clients connected to the broker can publish messages under certain topics. The clients can also subscribe to topics that are of interest to them. When a client publishes a message on a topic, the broker forwards the message to any client that has subscribed to the topic. This mechanism enables bi-directional communication with an extremely low overhead (2-byte header).

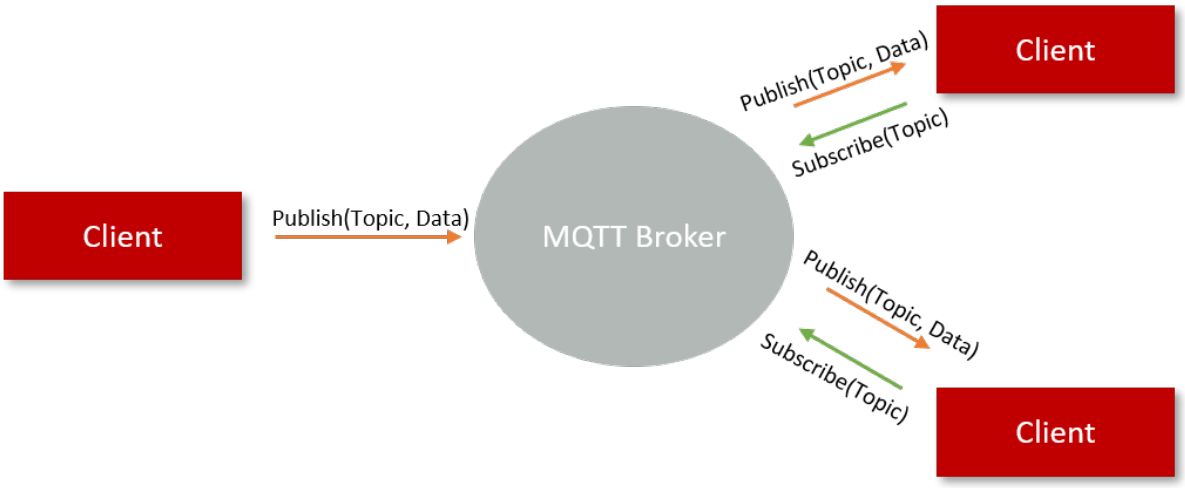

Figure 7: Publish/Subscribe mechanism

2.2 Topics/Subscriptions

Messages in MQTT are always published on topics. A hierarchy can be created in topics using the '/' character. For example, the status of smart light in the living room can have a topic "home/lighting/living_room/ligth_index".

Clients can create subscription on a topic by explicitly mentioning the topics or by using wildcard characters. There are two wildcards available, '+' and '#'. '+' matches any topics on a

single hierarchical level whereas '#' matches any number of levels. For example, subscribing to "home/lighting/+/light_index" would result in getting status change messages of all lights with "light_index" from all rooms of the house. On the other hand subscribing to "home/lighting/#" results in messages with all lights (all light indices) from all rooms.

This feature makes MQTT modular and highly scalable. Inserting a new node to an existing network does not require a lot of configuration.

2.3 QoS

Based on the requirement of the application one of the following levels of Quality-of-Service (QoS) can be chosen.

- **QoS level 0:** The broker/client delivers the message only once without acknowledgement of reception. The reliability in this case is same as that of the underlying TCP.
- **QoS level 1:** The broker/client delivers the message at least once. In this case an acknowledgement of received packet is done. This case however does not handle duplicate packets.
- **QoS level 2:** The broker/client delivers the message exactly once using a four-step handshake. This in turn offers higher reliability at the cost of lower throughput.

2.4 Message persistence

The publisher can specify if a message published to a topic has to be retained. If marked as retained, the broker retains the message and sends it to all new subscriptions. This acts as the "last known good" mechanism where nodes that come into network do not have to wait long to get the first message.

2.5 Last Will and Testament

This mechanism enables the client to publish one last message to all subscribed clients when it abruptly disconnects from the network. Clients can send a last will and testament message to the broker at any point. If the broker detects that a client has gone offline without a disconnect message, it sends the LWT message on the specified topic. This mechanism is very helpful to detect node failures in due to battery failure or networks outages.

2.6 Security

MQTT offers basic authentication where the client has to send a username and password with the connect message. The broker can authenticate the connection and allow or disallow a client. However, the user has to run MQTT over TLS/SSL to enable end-to-end encryption and advanced client authentication.

2.7 MQTT on cloud platforms

Most cloud platforms support the MQTT protocol although with subtle variations. Both Amazon AWS and Microsoft Azure support only the secure version of MQTT with authentication and end-to-end encryption built in. Please refer to [1] and [2] for more details.

3 Sensor-to-cloud application prototyping

In this chapter, a prototype of a sensor-to-cloud IoT application is presented using the Würth Elektronik eiSos's prototyping tools. The intention here is to go through the process step-by-step to enable the user to replicate the same without much effort. Connection to the two most commonly used cloud platforms, Amazon AWS and Microsoft Azure are presented. However, the same concept can be extended to work with any cloud platform that supports MQTT.

This chapter begins with a short introduction to the Würth Elektronik eiSos FeatherWing line of evaluation boards, which will be used for building this prototype application followed by the detailed description of the prototype application.

3.1 WE FeatherWings

Würth Elektronik eiSos's FeatherWings are a range of development boards that are open source and fully compatible with the Feather form factor. Through these development boards, WE brings a range of wireless connectivity modules, sensors and power modules to the Feather ecosystem.

Adafruit Feather is a complete line of development boards from Adafruit and other developers that are both standalone and stackable. They can be powered by LiPo batteries for on-the-go use or by their micro-USB plugs for stationary projects. Feathers are flexible, portable, and as light as their namesake.

FeatherWings are stacking boards and add functionality and room for prototyping. At its core, the Adafruit Feather is a complete ecosystem of products - and the best way to get your project flying by supercharging your prototyping.

In addition to the hardware, Würth Elektronik eiSos provides a software development kit (SDK) with examples to support all the WE FeatherWings. Here are the salient features of the WE FeatherWing SDK.

- The SDK is open-source and well documented.
- It uses a popular open-source tool chain including an IDE.
- The examples are written in Arduino-styled C/C++ for quick prototyping.
- The core components of the SDK are written in pure C to enable easy porting to any microcontroller platform.
- Development platform independent (Windows, Linux or MAC).
- Modular structure of the software stack makes it easy to integrate into any project.

The SDK can be accessed on Github at *https://github.com/WurthElektronik/FeatherWings*. More information regarding WE FeatherWings can be found under,
https://www.we-online.com/featherwings
In the following section, a prototype sensor-to-cloud solution is built using some of these FeatherWings.

3.2 Prerequisites

The following hardware is necessary to go through this demo.

1. Adafruit Feather M0 express development board.
2. WE Sensors FeatherWing, WE Calypso Wi-Fi FeatherWing.
3. An IEEE 802.11 b/g/n compatible access point with internet access.
4. A computer to build and modify the code. For the following example, a Windows10 PC is considered.

3.3 Set-up

The main objective here is to create a quick prototype of the system designed in section 1.4. The prototype consists of the following components,

- **Host Microcontroller:** For this prototype the Adafruit Feather M0 express to interface with the sensors, Wi-Fi module and the cloud. At the Feather M0's heart is an AT-SAMD21G18 ARM Cortex M0 processor, clocked at 48 MHz and at 3.3 V logic, the same one used in the new Arduino Zero. This chip has 256K of FLASH 32K of RAM (16x as much). This chip comes with built in USB so it has USB-to-Serial program and debug capability built in with no need for an FTDI-like chip. To make it easy to use for portable projects, a connector is added for any 3.7V Lithium polymer battery and built in battery charging.

- **Sensors:** The Würth Elektronik eiSos Sensor FeatherWing development board is used to monitor the environment. It consists of the following four sensors,
  - WSEN-PADS - Absolute pressure sensor (2511020213301)
  - WSEN-ITDS - 3-axis acceleration sensor (2533020201601)
  - WSEN-TIDS - Temperature sensor (2521020222501)
  - WSEN-HIDS - Humidity sensor (2525020210001)

  All four sensors are connected over the shared I^2C bus and hence can be connected to any of the Feather microcontroller boards.

- **Wireless connectivity and gateway:** Calypso FeatherWing provides Wi-Fi connectivity as well as MQTT client for cloud connection. The on-board Calypso radio module offers Wi-Fi connectivity based on IEEE 802.11 b/g/n with a fully featured TCP/IP (IPv4 and IPv6) stack. With out-of-the-box support to commonly used network applications like SNTP, HTTP(S), MQTT(S) Calypso offers an easy and secure solution to any IoT application.

  It has an AT-style command interface on the standard UART and hence can be connected to any of the Feather microcontroller boards. Advanced security features such as secure boot, secure storage and secure sockets makes this a good basis for secure IoT application.

- **Cloud data platform:** In this example, one of the two of the most commonly used cloud platforms are supported.
  - **Amazon AWS**
  - **Microsoft Azure**

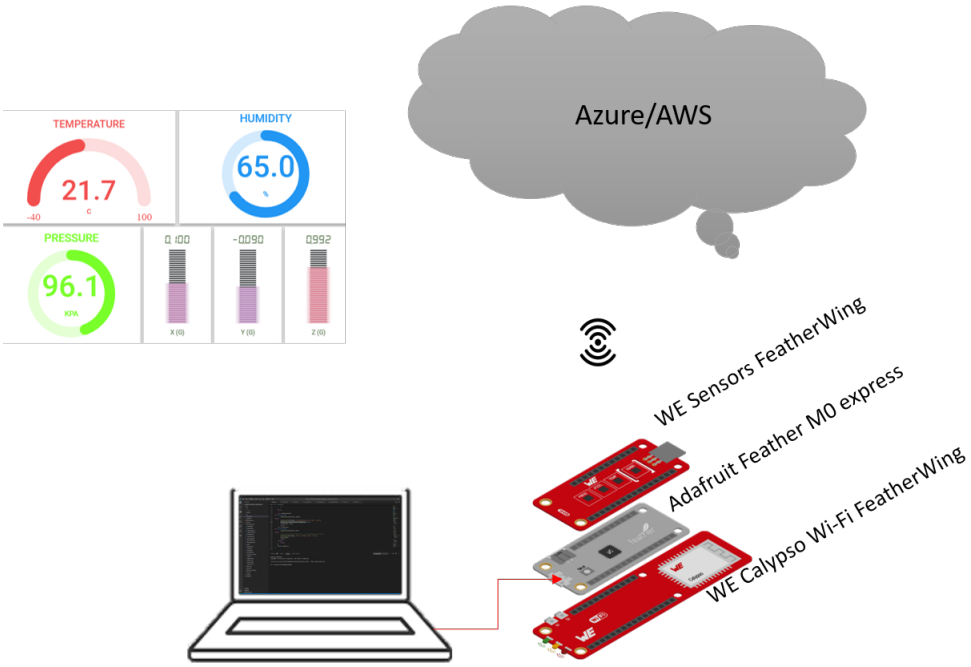

Figure 8: System set-up

3.4 Installation of tools

- Install Visual Studio Code on the computer of your choice following the instructions under *https://code.visualstudio.com/docs*
- Follow the instructions under *https://platformio.org/platformio-ide* to install PlatformIO IDE extension.
- Download or clone the sensor-to-cloud example from *https://github.com/WurthElektronik/FeatherWings*
- Follow the instructions in the respective links to configure the cloud platform
  - **Amazon AWS:** *https://github.com/WurthElektronik/FeatherWings/Sensor2CloudConnectivity/aws*
  - **Microsoft Azure:** *https://github.com/WurthElektronik/FeatherWings/Sensor2CloudConnectivity/azure*

3.5 Configure and run the code

1. Prerequisites for running this example:
   - A Wi-Fi access point with WPA2 personal security and a known password and SSID.
   - For Azure, the IoT-Hub has to be configured as described in the previous section.
   - For AWS, the IoT Core has to be configured as described in the previous section.
2. Configure the Wi-Fi network to be used in "main.cpp".

```
// WiFi access point parameters
#define WI_FI_SSID "AP"
#define WI_FI_PASSWORD "pw"
```

3. Select the cloud platform of choice in "main.cpp".

```
#define AZURE_CONNECTION 1
#define AWS_CONNECTION 0
```

4. For connection to Azure, the following parameters should be configured in "main.cpp".

```
#define MQTT_CLIENT_ID "we-iot-device"
#define MQTT_SERVER_ADDRESS "we-exampleHub.azure-devices.net"
#define MQTT_PORT 8883
#define MQTT_TOPIC "devices/we-iot-device/messages/events/"
#define MQTT_USER_NAME "example_for_documentation.azure-devices.net/we
    -iot-device"
#define MQTT_PASSWORD
"SharedAccessSignature␣"
"sr=we-iotHub.azure-devices.net%2Fdevices%2Fwe-iot-device&sig="
"example_for_documentation=1111111111"
```

5. For connection to AWS, the following parameters should be configured in "main.cpp".

```
/*MQTT settings - AWS*/
#define MQTT_CLIENT_ID "we-iot-device-t1"
#define MQTT_SERVER_ADDRESS "example_for_documentation.iot.amazonaws.
    com"
#define MQTT_PORT 8883
#define MQTT_TOPIC "test"
#define MQTT_CERT_PATH "cert"
#define MQTT_PRIV_KEY_PATH "key"
#define MQTT_USER_NAME "calypso"
#define MQTT_PASSWORD "secret"

#define MQTT_CERTIFICATE "-----BEGIN␣CERTIFICATE-----\n\
␣␣␣␣111111111111111111111111111111111111111111111111111111111111\n
    \
␣␣␣␣example_for_documentation\n\

␣␣␣␣-----END␣CERTIFICATE-----"
```

```
    #define MQTT_PRIV_KEY "−−−−−BEGIN␣RSA␣PRIVATE␣KEY−−−−−\n\
␣␣␣␣1111111111111111111111111111111111111111111111111111111111111111\n
    \
␣␣␣␣example_for_documentation\n\
␣␣␣␣−−−−−END␣RSA␣PRIVATE␣KEY−−−−−"
    #endif
```

6. Configure SNTP server parameters in "main.cpp".

```
    // SNTP settings
    #define SNTP_TIMEZONE "+60"
    #define SNTP_SERVER "0.de.pool.ntp.org"
```

7. Build and upload the code from the PlatformIO tab as shown in the figure 9.

8. After successful upload, click on "Monitor" in the PlatformIO extension tab to view the debug logs in the serial terminal.

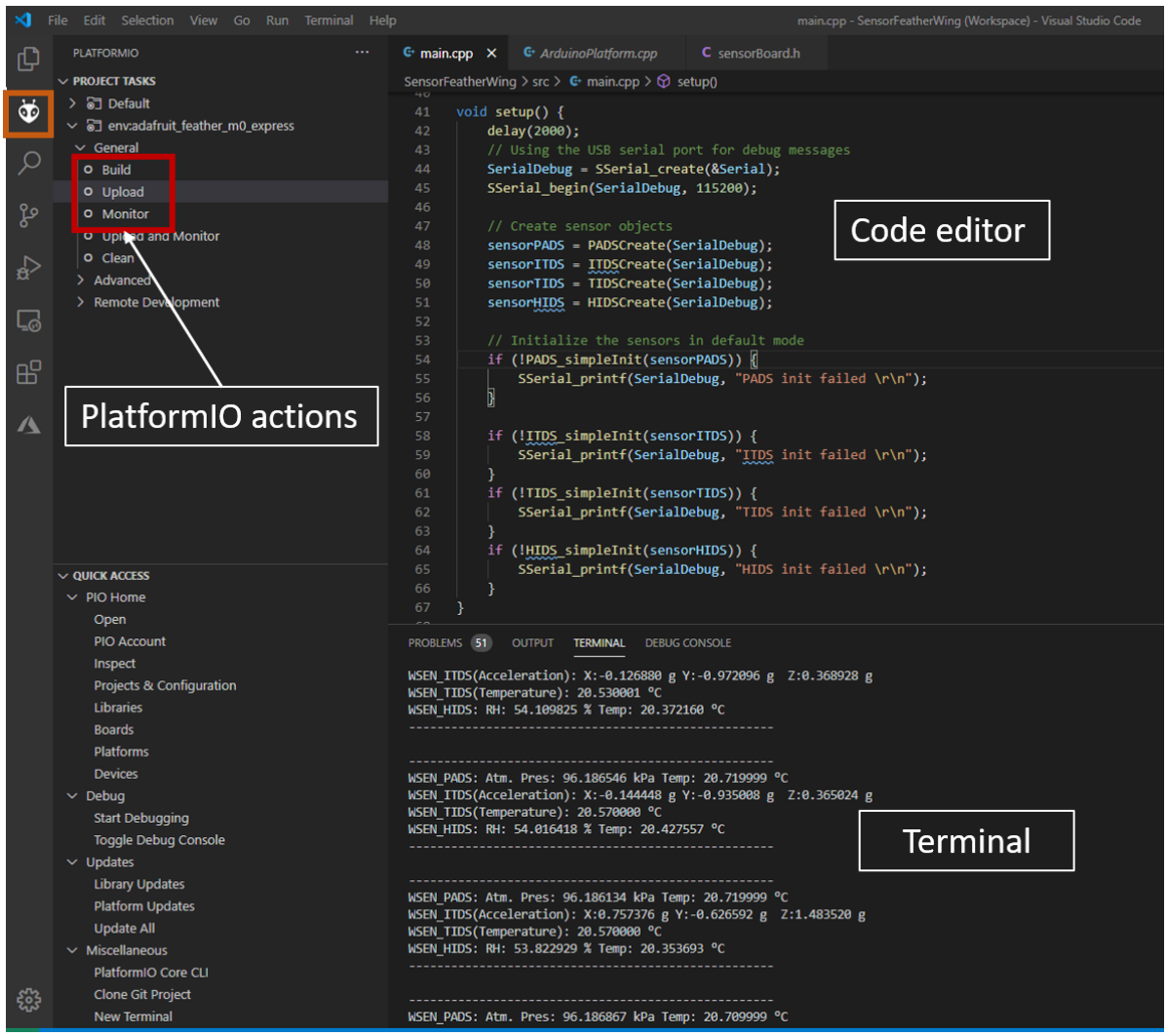

Figure 9: VS code

3.6 Sensor data in the cloud

On successful upload the following actions are performed by the application,

- The Calypso Wi-Fi module connects to the configured Wi-Fi network.
- A secure MQTT connection is made with the configured cloud.
- Sensor data is periodically read and transmitted to the cloud.

Please refer to the instruction in the links below to process the sensor data further.

- **Amazon AWS:**
  https://github.com/WurthElektronik/FeatherWings/Sensor2CloudConnectivity/aws
- **Microsoft Azure:**
  https://github.com/WurthElektronik/FeatherWings/Sensor2CloudConnectivity/azure

A sample snippet of the sensor data is as shown below,

{{ "messageId":1},{"PADS_T":22.81,"PADS_P":94.766},{"ITDS_X":−0.08,"ITDS_Y":−1.019,"
ITDS_Z":0.158},{"TIDS_T":22.87},{"HIDS_T":23.2,"HIDS_RH":54.19}}

4 Summary

Designing an IoT solution brings with it a number of challenges. Being multifaceted, IoT applications demands a lot of competence from hardware design to UI development. Under such situations, it is prudent to take a modular approach. This means breaking down the architecture into functionally independent blocks. One such essential building-block for an IoT application is wireless connectivity. Integrating a radio module such as Calypso enables the designer to completely delegate the task of wireless connectivity, which saves a lot of effort enabling quicker time-to-market.

At the other end, it is a completely different approach necessary to arrive at a cloud solution. Flexibility and security are the key parameters that needs to be kept in mind while designing a cloud solution.

In this application note, a simple IoT technology stack that represents the principle behind any IoT solution is presented. Further, with the help of an example, a solution using the Calypso Wi-Fi module and AWS/Azure cloud platforms is described. Finally, a step-by-step prototype of the proposed solution is presented.

5 References

[1] MQTT Amazon AWS. https://docs.aws.amazon.com/iot/latest/developerguide/mqtt.html.

[2] MQTT Microsoft Azure. https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-mqtt-support.

[3] MQTT specification (V3.1.1). https://mqtt.org/mqtt-specification/.

6 Important notes

The Application Note and its containing information ("Information") is based on Würth Elektronik eiSos GmbH & Co. KG and its subsidiaries and affiliates ("WE eiSos") knowledge and experience of typical requirements concerning these areas. It serves as general guidance and shall not be construed as a commitment for the suitability for customer applications by WE eiSos. While WE eiSos has used reasonable efforts to ensure the accuracy of the Information, WE eiSos does not guarantee that the Information is error-free, nor makes any other representation, warranty or guarantee that the Information is completely accurate or up-to-date. The Information is subject to change without notice. To the extent permitted by law, the Information shall not be reproduced or copied without WE eiSos' prior written permission. In any case, the Information, in full or in parts, may not be altered, falsified or distorted nor be used for any unauthorized purpose.

WE eiSos is not liable for application assistance of any kind. Customer may use WE eiSos' assistance and product recommendations for customer's applications and design. No oral or written Information given by WE eiSos or its distributors, agents or employees will operate to create any warranty or guarantee or vary any official documentation of the product e.g. data sheets and user manuals towards customer and customer shall not rely on any provided Information. THE INFORMATION IS PROVIDED "AS IS". CUSTOMER ACKNOWLEDGES THAT WE EISOS MAKES NO REPRESENTATIONS AND WARRANTIES OF ANY KIND RELATED TO, BUT NOT LIMITED TO THE NON-INFRINGEMENT OF THIRD PARTIES' INTELLECTUAL PROPERTY RIGHTS OR THE MERCHANTABILITY OR FITNESS FOR A PURPOSE OR USAGE. WE EISOS DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT RELATING TO ANY COMBINATION, MACHINE, OR PROCESS IN WHICH WE EISOS INFORMATION IS USED. INFORMATION PUBLISHED BY WE EISOS REGARDING THIRD-PARTY PRODUCTS OR SERVICES DOES NOT CONSTITUTE A LICENSE FROM WE eiSos TO USE SUCH PRODUCTS OR SERVICES OR A WARRANTY OR ENDORSEMENT THEREOF.

The responsibility for the applicability and use of WE eiSos' components in a particular customer design is always solely within the authority of the customer. Due to this fact it is up to the customer to evaluate and investigate, where appropriate, and decide whether the device with the specific characteristics described in the specification is valid and suitable for the respective customer application or not. The technical specifications are stated in the current data sheet and user manual of the component. Therefore the customers shall use the data sheets and user manuals and are cautioned to verify that they are current. The data sheets and user manuals can be downloaded at *www.we-online.com*. Customers shall strictly observe any product-specific notes, cautions and warnings. WE eiSos reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time without notice.

WE eiSos will in no case be liable for customer's use, or the results of the use, of the components or any accompanying written materials. IT IS CUSTOMER'S RESPONSIBILITY TO VERIFY THE RESULTS OF THE USE OF THIS INFORMATION IN IT'S OWN PARTICULAR ENGINEERING AND PRODUCT ENVIRONMENT AND CUSTOMER ASSUMES THE ENTIRE RISK OF DOING SO OR FAILING TO DO SO. IN NO CASE WILL WE EISOS BE LIABLE FOR

CUSTOMER'S USE, OR THE RESULTS OF IT'S USE OF THE COMPONENTS OR ANY ACCOMPANYING WRITTEN MATERIAL IF CUSTOMER TRANSLATES, ALTERS, ARRANGES, TRANSFORMS, OR OTHERWISE MODIFIES THE INFORMATION IN ANY WAY, SHAPE OR FORM.

If customer determines that the components are valid and suitable for a particular design and wants to order the corresponding components, customer acknowledges to minimize the risk of loss and harm to individuals and bears the risk for failure leading to personal injury or death due to customers usage of the components. The components have been designed and developed for usage in general electronic equipment only. The components are not authorized for use in equipment where a higher safety standard and reliability standard is especially required or where a failure of the components is reasonably expected to cause severe personal injury or death, unless WE eiSos and customer have executed an agreement specifically governing such use. Moreover WE eiSos components are neither designed nor intended for use in areas such as military, aerospace, aviation, nuclear control, submarine, transportation, transportation signal, disaster prevention, medical, public information network etc. WE eiSos must be informed about the intent of such usage before the design-in stage. In addition, sufficient reliability evaluation checks for safety must be performed on every component which is used in electrical circuits that require high safety and reliability functions or performance. COSTUMER SHALL INDEMNIFY WE EISOS AGAINST ANY DAMAGES ARISING OUT OF THE USE OF THE COMPONENTS IN SUCH SAFETY-CRITICAL APPLICATIONS.

List of Figures

Contact
Würth Elektronik eiSos GmbH & Co. KG
Division Wireless Connectivity & Sensors

Max-Eyth-Straße 1
74638 Waldenburg
Germany

Tel.: +49 651 99355-0
Fax.: +49 651 99355-69
www.we-online.com/wireless-connectivity