EnOcean Link

16.04.2020

REVISION HISTORY

The following major modifications and improvements have been made to the first version of this document:

No	Major Changes
1.0	Initial version
1.1.	Update to new format and links

Published by EnOcean GmbH, Kolpingring 18a, 82041 Oberhaching, Germany
www.enocean.com, info@enocean.com, phone +49 (89) 6734 6890

Important!

This information describes the type of component and shall not be considered as assured characteristics. No responsibility is assumed for possible omissions or inaccuracies. Circuitry and specifications are subject to change without notice. For the latest product specifications, refer to the EnOcean website: http://www.enocean.com.
As far as patents or other rights of third parties are concerned, liability is only assumed for modules, not for the described applications, processes and circuits.
EnOcean does not assume responsibility for use of modules described and limits its liability to the replacement of modules determined to be defective due to workmanship. Devices or systems containing RF components must meet the essential requirements of the local legal authorities.
The modules must not be used in any relation with equipment that supports, directly or indirectly, human health or life or with applications that can result in danger for people, animals or real value.
Components of the modules are considered and should be disposed of as hazardous waste. Local government regulations are to be observed.
Packing: Please use the recycling operators known to you. By agreement we will take packing material back if it is sorted. You must bear the costs of transport. For packing material that is returned to us unsorted or that we are not obliged to accept, we shall have to invoice you for any costs incurred.

TABLE OF CONTENT

1 GENERAL DESCRIPTION

With EnOcean Link EnOcean has launched the first middleware for energy harvesting wireless technology. OEMs can now integrate EnOcean technology easier and faster into a wide range of applications and systems, such as those in smart homes. The software provides a universal interface for wireless communication and automatically interprets information from EnOcean telegrams. As a result, sensor data such as moisture or temperature is prepared so that different devices, servers and even cloud services can process it immediately. OEMs can obtain EnOcean Link directly from EnOcean as licensed software.

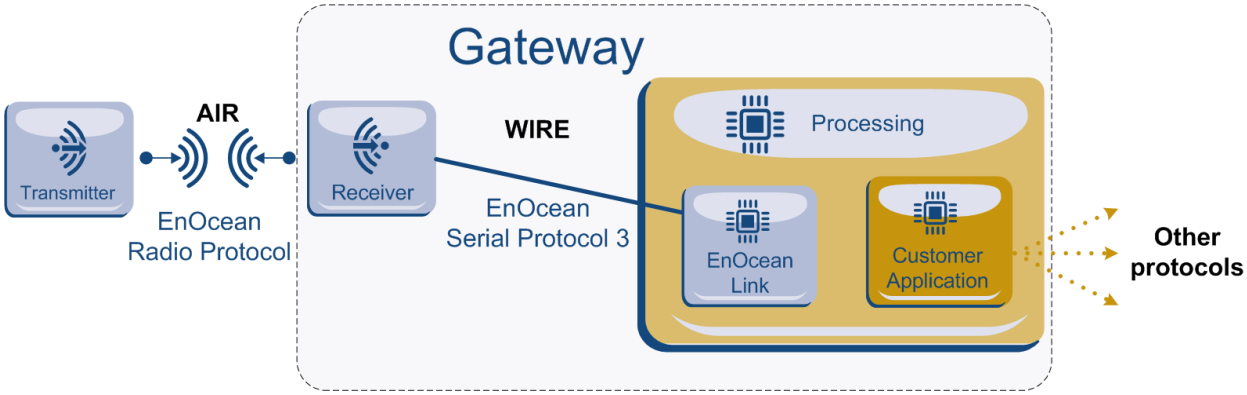
1.1 Basic functionality

EnOcean Link converts the bits and bytes of an EnOcean telegram directly into data values. In doing so, the middleware automatically takes into account all specifications of the EnOcean protocol stack and the EnOcean Equipment Profiles (EEPs) and Generic Profiles (GP) of the EnOcean Alliance as well as encryption mechanisms. This means EnOcean Link provides all wireless applications with a ready-made key to the energy harvesting wireless world. Since the software interprets all data, it also ensures the interoperability of equipment from different manufacturers.

EnOcean Link is a software project. Target hardware are external micro controllers – not provided by EnOcean. It is indented for embedded devices that are processing EnOcean telegrams provided by an ESP 3 (EnOcean Serial Protocol 3) EnOcean Gateway (i.e. USB 300, TCM 310 or TCM 515). Basic scenario is in the figure below.

EnOcean

EnOcean Link



1.2 Key benefits

- **Faster development**

The use of EnOcean Link saves OEMs valuable development resources, which significantly quickens the time-to-market for energy harvesting wireless applications. Instead of developing their own software for a particular application in order to interpret the communication protocol of the energy harvesting wireless technology, OEMs can use the finished middleware for this purpose immediately. The same applies to any application that integrates or further processes the energy harvesting wireless technology as well as to each EnOcean radio frequency.

Manufacturers can now focus all their attention on the application and their core activities. At the same time, EnOcean Link makes it easier to implement powerful energy harvesting wireless networks, which are required by comprehensive smart home and M2M systems.

- **Continuous updates**

EnOcean will update EnOcean Link continuously. With the purchase of the licensed software, OEMs will receive all updates within the first year free of charge. Afterwards, they can extend this service, including the related support, by entering into a maintenance agreement.

- **Fully prepared data**

In a smart home application, for example, a gateway can use EnOcean Link to immediately interpret the information from energy harvesting wireless sensors, including temperature, occupancy and light intensity, and forward this information to a central building control system. It does not matter if the system is networked with BACnet, KNX, Z-Wave, ZigBee, Bluetooth low energy or other automation protocols. For cloud services, working with GSM or WiFi, the data speed needed for automated control is also increased. If wireless telegrams are encrypted on the air interface, EnOcean Link decodes them before they are further processed by an external controller.

- **EnOcean Certified**

EnOcean certifies EnOcean Link according all valid certification standards from the EnOcean Alliance and ensures all quality measures. All communication profiles have been tested and thus provide best baseline for interoperability.

1.3 Features

- Serial Commands Interface ESP3
- Interprets EnOcean Equipment Profile 2.5 and Generic Profiles data Handling
- Set and Get Values as float or integer(depending on the datatype).
- Handle the LRN Process of UTE, EEP, GP, Smart Acknowledge and Security.
- Flexible and extendable software architecture with layers corresponding to the EnOcean Protocol Stack
- Full support for Linux based systems, portable to other architectures
- Security encryption and decryption
- Sending and Receiving of telegrams
- Simulating devices for all Profiles
- Storing and loading of configurations in an human readable Version
- Message Chaining
- Remote Management support
- Smart Acknwoledge support
- Different Software filters for telegrams
- You can implement your own Filter

1.4 References

1. EnOcean Alliance Standard - https://www.enocean-alliance.org/de/ueber-uns/spezifikationen/
2. Developer User Manual and Specification – available only with Trial or Full License
3. EnOcean Link Product page - http://www.enocean.com/en/enocean-software/enocean-link/

Useful Links for developers:

GatewayController Firmware User manual - TCM 310 -
https://www.enocean.com/en/enocean_modules/tcm-310/
GatewayController Firmware User manual - USB 300
https://www.enocean.com/en/enocean_modules/usb-300-oem/

TCM 515 https://www.enocean.com/en/products/enocean_modules/tcm-515/

1.5 EnOcean Link Packages

Full Version Deliverables:

- Complete Source Code – to cross compile for your platform
- Complete User Manual und Source Code Documentation (.chm file)
- DolphinView script to emulate EnOcean Devices (.do file)

Evaluation Version Deliverables:

- Feature limited Source Code – to cross compile for your platform
- Complete User Manual und Source Code Documentation (.chm file)
- DolphinView script to emulate EnOcean Devices (.do file)

1.6 About this document

This document is not the full user manual and specification that is needed by a developer for the usage of the library. This document is rather a technical presentation of EnOcean Link which describes main uses cases, architecture of the framework and simple examples. After reading this document, you should be aware of the potential of EnOcean Link for your project and what it‘s constrains are.

2 USE CASE DESCRIPTION

EnOcean Link can be summarized into these characteristics:

- a piece of software
- a telegram interpreter – middleware for EnOcean
- intended to run on separate µC – not EnOcean Hardware
- an easy entry point to EnOcean World / Stack

Based on these points please refer to the scenarios below:

- a smart home (residential area), use case scenario below in Figure 1.
- an office building scenario (commercial area), use case in Figure 2.

Provided use case descriptions are there to picture the basic use scenarios. EnOcean Link is not limited to these usages only.

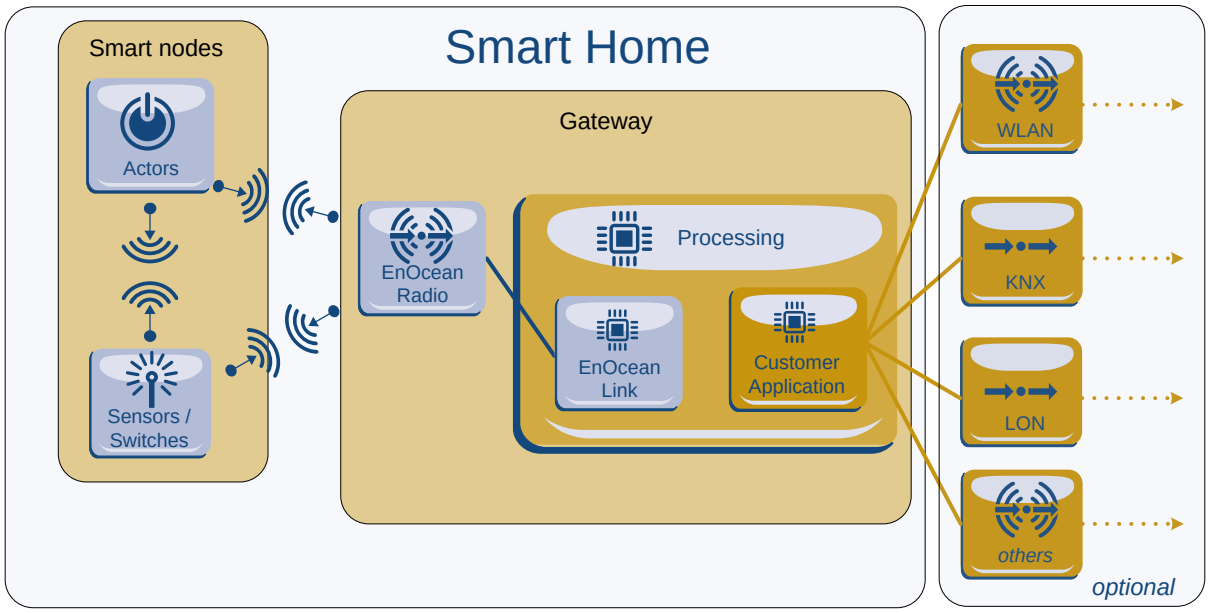


Figure 1 Smart Home scenario

In the Smart Home scenario in Figure 1 we can see that EnOcean Link is part of the Smart Home box. Typically, this is a central unit - one per household. The smart home box holds the intelligence of the applications and in most cases has connectivity to the cloud via a third party protocol. EnOcean Link runs with the "customer" application on the same hardware and EnOcean Link provides an interface to the EnOcean world. Most probably, the information is not carried in EnOcean protocol format beyond EnOcean Link.

On the Building scenario in Figure 2 EnOcean Link is used on a central device – control server. It is used to control the whole building, holds the application intelligence and does not have to be physically located in the building. There are several EnOcean Gateways that gather the radio telegrams and are also used to send telegrams out. These Gateways are

connected with the control server by a backbone, which does not have to be EnOcean Radio based or wireless. For example, these can be an EnOcean/IP Gateway. EnOcean Link interprets the telegrams coming in from all EnOcean Gateways.

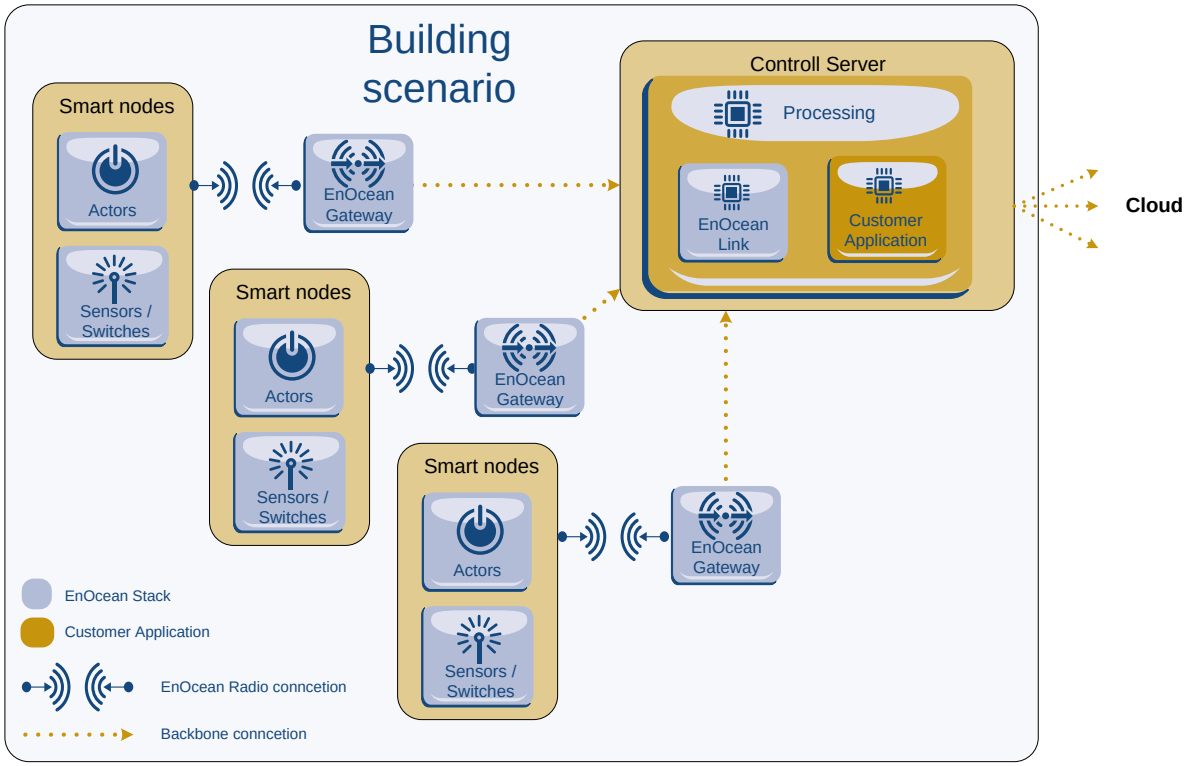


Figure 2 Building scenario

3 ENOCEAN LINK PROCESS ARCHITECTURE

EnOcean Link provides several service layers to operate an EnOcean Radio based sensor network. These service layers are connected together to provide a clear user interface. A very simple overview of the most important layers, actions and interfaces can be found in Figure 3.

On the **physical layer** EnOcean Link receives the UART Data stream from the EnOcean gateway. This signal can come directly from an EnOcean Gateway (e.g. TCM 310 – Smart Home Scenario) or can be provided by a backbone which was optionally "tunnelled" before (e.g. EnOcean IP Gateways – Office Building Scenario).

The **EnOcean Serial Protocol encoder** is located over the physical layer. This layer handles all necessary operations to prepare telegrams data fields into a suitable form for further processing. Most important data fields are:

- payload
- sender ID

The **EnOcean Profile interpreter,** based on the stored profile, interprets the telegram payload into human readable values. (e.g. temperature, humidity). These values are presented as the devices channels.

The devices channels are handled by the application trough the **API interface**. This interface can be called directly or through an "adapter" that encapsulates the EnOcean Link library (e.g. JSON Webservice). The adapter is not part of EnOcean Link.

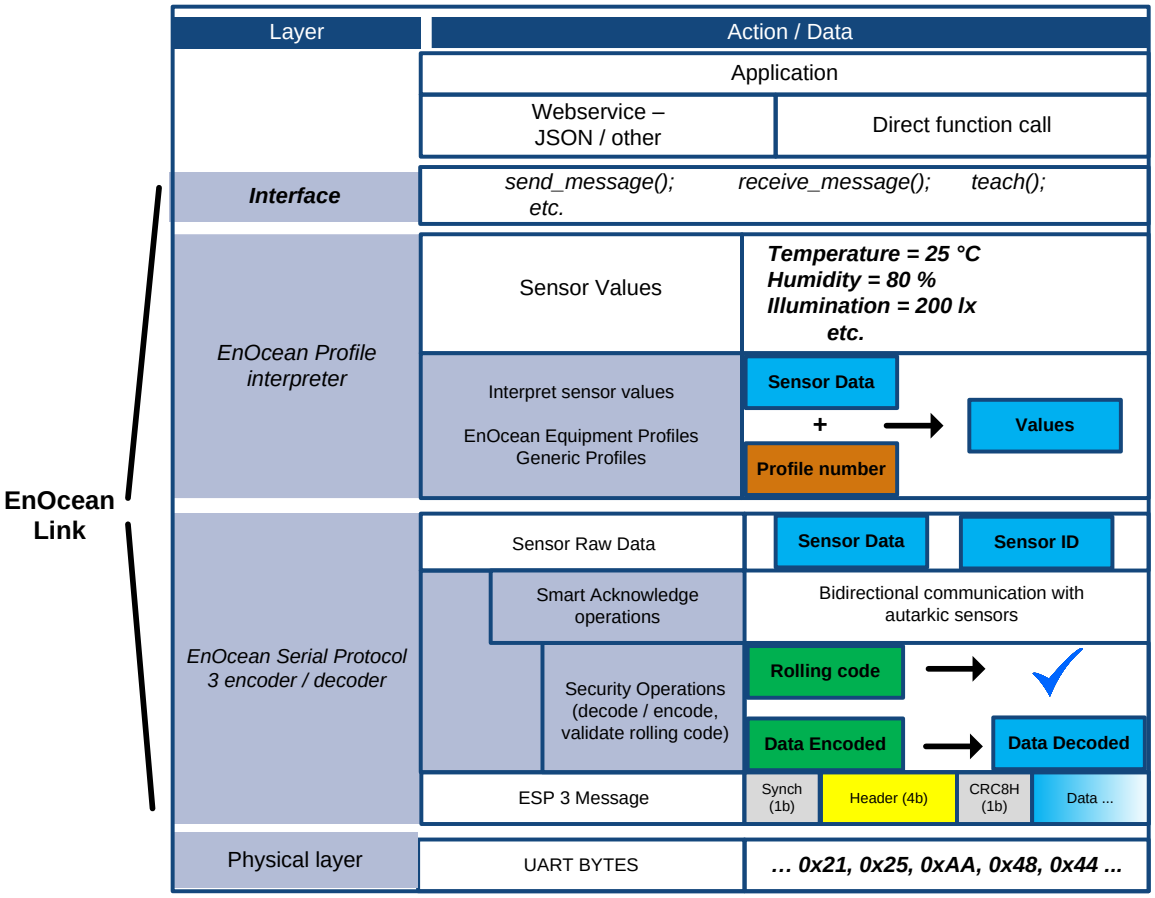


Figure 3 Layers, actions and interface overview

EnOcean Link is a library without an internal process running in it. From this perspective it is a library that offers services of several layers. The application / developer decides how to call them and thus gains control of the:

- execution order
- inbound and outbound communications interfaces (lower layer - UART, upper layer – application)
- storage

3.1 Function Calls

EnOcean Link enables its functionality through API functions. Detailed specification of the interface functions can be found in Reference 1. For EnOcean Link these three main functionalities there are:

1. Receiving and decoding incoming communication according to the profile standard
2. Executing Device binding / TeachIn and store device information
3. Encoding outgoing communication according to the profile standard and sending out

To demonstrate the nature of the calls in EnOcean Link and to gain a basic overview, please refer to the following two figures. The used function names in the figure do not respond to the actual function names in the source code.

In Figure 4 you can see the sending sequence diagram. After the application executes the send-call, EnOcean Link then codes the information and sends a serial request to the EnOcean Gateway (in diagram TCM 310). The gateway performs this operation and responds with a return code on the serial interface. EnOcean Link waits for this response and provides the result as part of the send-call back to the application.

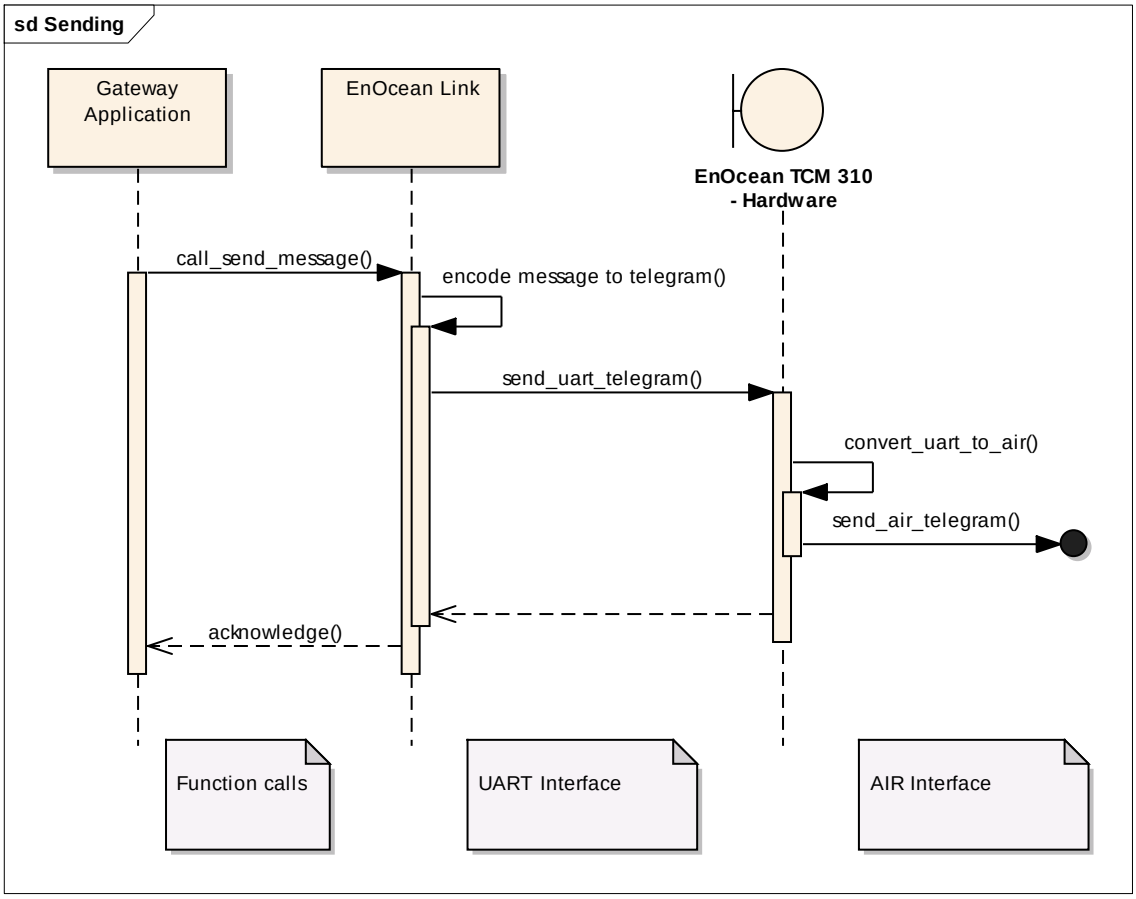


Figure 4 Sending sequence diagram

In Figure 5 the sending sequence diagram is shown. Received telegrams are passed by the EnOcean Gateway (in Diagram TCM 310). The UART Stream is asynchronous, so the UART driver of the device (e.g. Smart Home box) buffers the information. The application fetches the telegram by executing the receive-call. EnOcean Link gets received UART bytes from the

UART driver and validates the serial message. Then it passes the information according to the serial specification and passes it to the application.

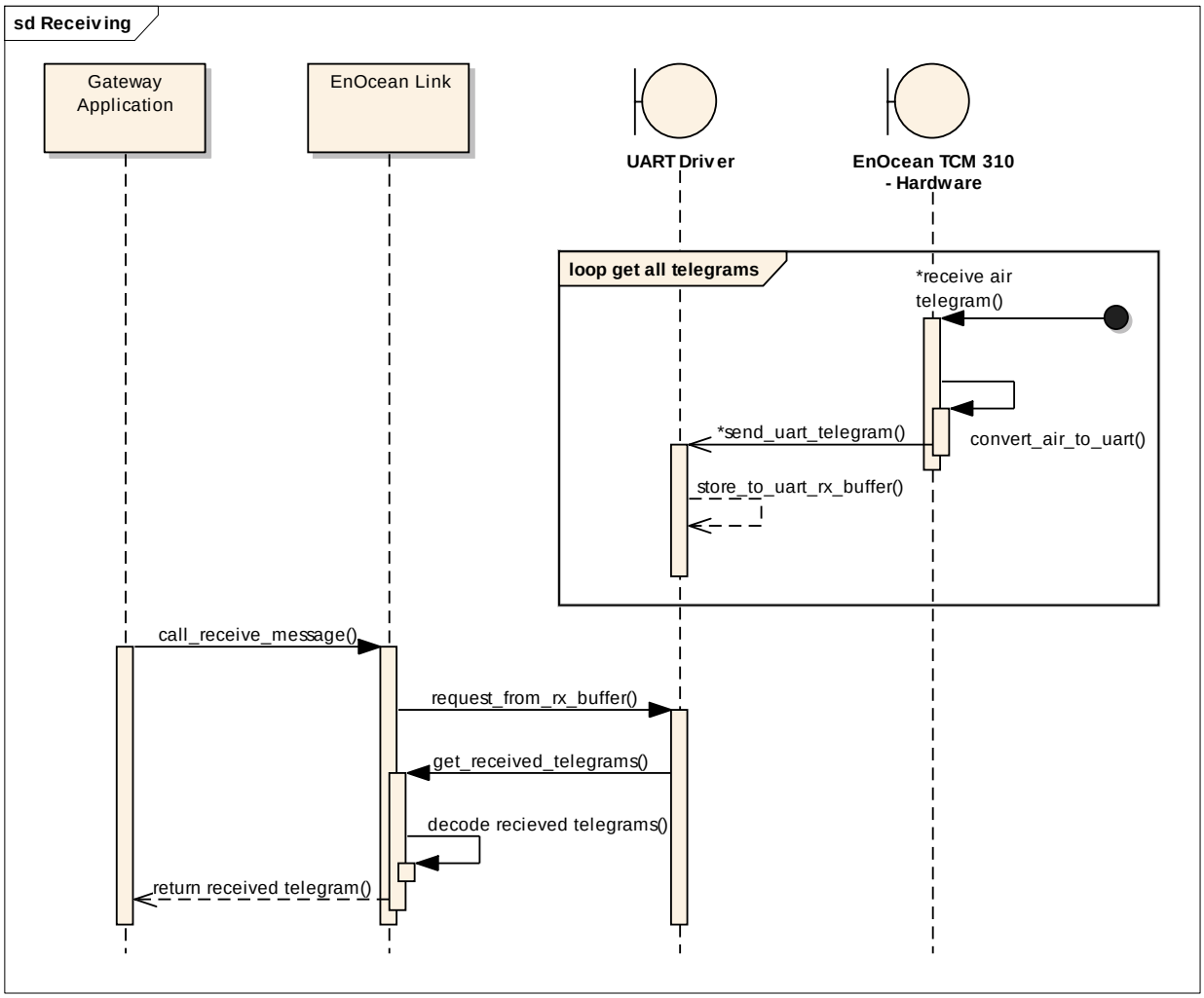


Figure 5 Receiving sequnece diagram

3.2 Interpreting the received data - example

To summarize the process architecture please see the following Figure.

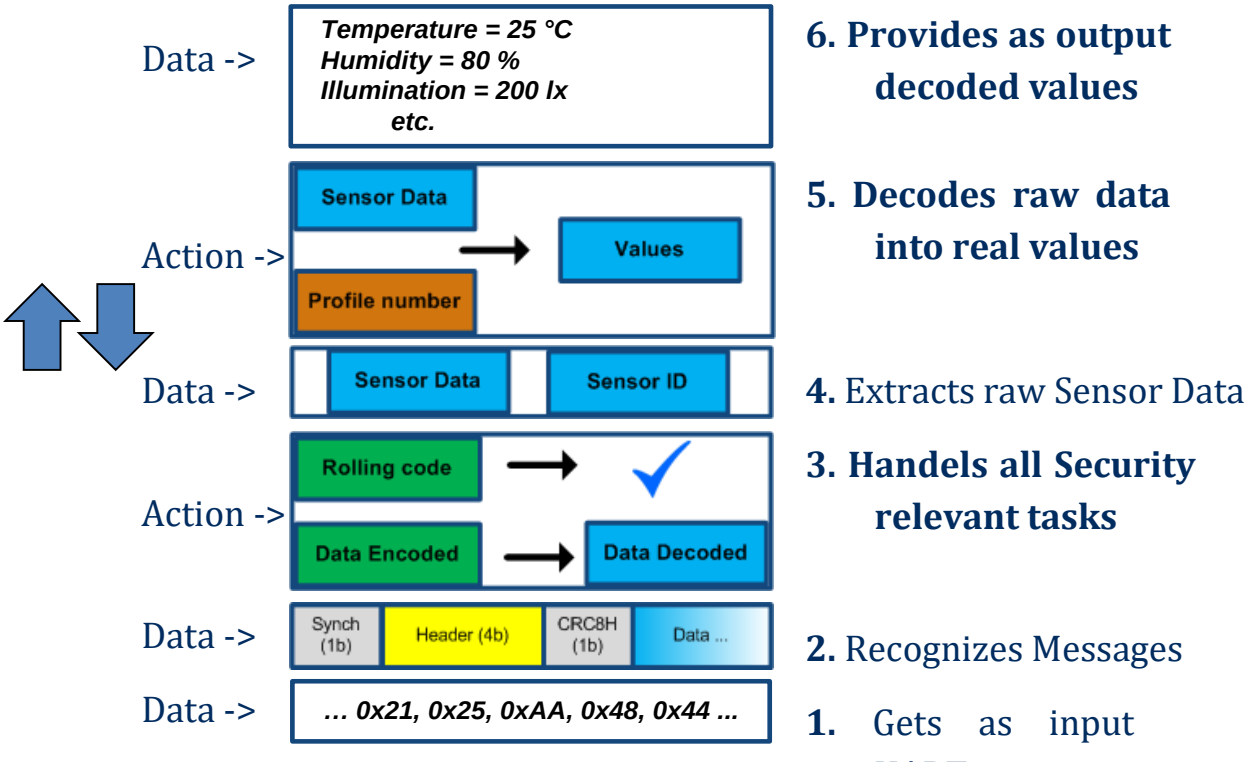


Figure 6 Interpret the received data example

The Figure 6 show how the received telegrams are processed and how the sensor values / device channels are extracted. To execute this process the application has to execute these two calls:

1. eoGateway->Receive()
2. profile->GetChannel(index/type)

For detailed examples, please look for Reference 1 or Chapter 6.

4 DEVICE CHANNELS – INTERPRETATION OF THE EEPS AND GPS

The most important parts of the Application interface of EnOcean Link are the device channels. The channels present the values a device provides in an abstract form. The device profiles – how the device transmits and codes its values - are described in the EnOcean Equipment Profiles – see reference **Error! Reference source not found.** and Generic Profiles – see reference **Error! Reference source not found.**. The structure and definition of the device channels is similar to generic profiles, but to have full semantic capabilities of device description as EEPs have, we needed to extend the semantic structures of it.

By using device channels, the application does not see a difference between using a device with EEPs or GPs. We can say that we mapped the existing EEPs to the device channels, or in other words, the EEPs are mapped to an extended Generic Profiles definition. By doing this, the EEPs are abstracted to a more generic definition and are easier to handle on the application level.

There are three device channel types:

1. **Signal**
1. These channels present the physical / measurable values. Temperature, Humidity and Illumination channels are for example signal type device channels.

2. **Flag**
   These values can only have two states. For example: ON/OFF, UP/DOWN and PRESSED/ NOT PRESSED are flag channel types.

3. **Enum**
2. Enumerations present values that show states, positions, counters or any other values that are not flags and do not have a physical unit.

In EnOcean Link, Channel types are described by a list called CHANNEL_TYPE, please see reference 1 for details. As an example, please use this list:

```
...

S_MASS    - Mass kg
S_POWER    - Power Watts in W
S_PRESSURE    - Pressure Pascals
S_RELHUM    - Relative humidity Percent
S_RESIST - Resistance Ohms
...

E_WINDOWHANDLE
E_FANSPEED

...

F_DAY_NIGHT - Day(1) / night(0)
```

```
F_DOWN - Down "-" (1 when pressed)
F_GENALARM - General alarm (1)
F_HEAT_COOL - Heat (1) / cool (0)
...
```

The complete Device Channel description includes CHANNEL_TYPE and scaling information (engineering minimum, engineering maximum, scale factor) and SUB_FLAG. SUB_FLAG is only used if Channels with same CHANNEL_TYPE are used by one device. The SUB_FLAG is then used to differ between the Channels.

4.1 Example of device channels definition

Please use following examples of channel definitions.

The following channels are available for profile A5-07-02 and A5-07-03:

Channel Index	Channel Type	Type
0	**S_VOLTAGE**	float
1	**S_LUMINANCE**	float
2	**F_OCCUPIED**	uint8_t

The following channels are available for Profiles A5-08-01, A5-08-02, A5-08-03:

Channel Index	Channel Type	Type
0	**S_VOLTAGE**	float
1	**S_LUMINANCE**	float
2	**S_TEMP**	float
3	**F_OCCUPIED**	uint8_t
4	**F_BTN_PRESS**	uint8_t

To see the original EEP definition of these profiles please see reference **Error! Reference source not found.**.

5 PLATFORM DEPENDENCIES

EnOCean Link was developed for a Linux based system. These specifications apply:

It uses these standard libraries:

- `std::map`
- `std::vector`
- `std::list`
- `std::string`
- `std::set`
- `std::iostream`
- `<time.h>`
- `ctime`
- `<termios.h>`
- `<fcntl.h>`
- `<stdint.h>`
- `<math.h>`

Implemented in:

- C++

Hardware (minimum requirements):

- CPU 16-bit, 24 MHz
- Non volatile data Storage per one EnOcean device – cca. 100 byte
- Non volatile memory to store rolling code (high count of write cycles) – periodically changing data aprox. After every 1-50 security telegram
- UART 57600 BAUD +/- 5%
- Timer – milliseconds (system time)
- 16 kb RAM and
- 256 kb Program memory

Compilers:

- g++
- gcc

5.1 Porting to other platforms

To port EnOcean Link to another operating system / platform you have to consider these main dependencies from the existing platform:

- File system storage – store the device information in non-volatile memory. For a porting you have to adjust the storage manager.
- UART Driver – EnOcean Link works directly with the hardware UART Interface. For a porting you have to adjust / implement the packet stream – `getByte` function.
- System timer – for protocol handling and time-outs, EnOcean Link needs an access to a `getTime` function which delivers the time with milliseconds accuracy.

To use with another programming platform (C#, JAVA) you call native interfaces or encapsulate the EnOcean Link library into a JSON Webserver.

6 APPLICATION EXAMPLE

Please find following "Hello World" tutorial of EnOcean Link.

This Tutorial will help you to understand the base concept of an **`eoGateway`**, **`eoStorageManager`** and the concept of Filter **`eoIDFilter`**, **`eodBmFilter`**.

After starting up, the example jumps to learnMode for near field devices. These learned in devices and their configuration will be stored using the **`eoStorageManager`**. To start developing an EnOcean Link application you are to either include the SourceCode or a pre-compiled library to your linker and include the `eoLink.h` file.

```
#include "./eoLink.h"
```

In our example we use some defines, to change the LearnTime, the config FileName and the device to use.

```
#define SER_PORT "/dev/ttyUSB0"//The Serial Port Device

#define SAVE_CONFIG "./learned.txt" //Where to store the configuration

#define LEARN_TIME_S 1 //The time the gateway should stay in learnmode
```

The following code parts can be either all in the main function, or an own function which returns an integer.

To receive Telegrams, we need an **`eoGateway`** and connect it via **`eoGateway::Open`** to an EnOcean Gateway Device like the USB300.

```
eoGateway myGateway;

if (myGateway.Open(SER_PORT)!=OK)

{

      printf("Failed to open USB300\n");

      return 0;

}
```

If the connection is stable, we could now start to receive Telegrams, but we only want to receive Telegrams which are in the near field range. To achieve this we add an **`eodBmFilter`** to our `eoGateway` Class.

```
eodBmFilter *  learnFilter = new eodBmFilter();

myGateway.learnFilter = learnFilter;

learnFilter->maxdBm=-0;

learnFilter->mindBm=-60;
```

As we only want to see telegrams from learned IN devices in Normal Mode, we have to add an **`eoIDFilter`**.

```
eoIDFilter * myFilter = new eoIDFilter();

myGateway.filter=myFilter;
```

Now, we're nearly ready to receive and handle the Packets, but to do so we need an `uint16_t` variable to handle the ReturnFlags of the **`eoGateway::Receive`** Function.

```
	uint16_t recv;
```

And as we want to stay only a short time in the learn mode after the startup, we've to add some variables which check the passed time. We're going to use `eoTimer::getTickCount()`, which increases the Tick amount every 10ms.

```
	uint32_t learnTime=eoTimer::GetTickCount()+LEARN_TIME_S*10000;

	uint32_t time=eoTimer::GetTickCount();
```

In this case we say we want to TeachIN devices which either send a 4BS TeachIN telegram, or if we get a RPS telegram, that we TeachIN a 2 Button Rocker.

As the RPS Profiles don't have a TeachIN message, we've to set the profile manually.

```
	myGateway.TeachInModule->SetRPS(0x02,0x01);
```

Now we only have to change from the normal mode to the learn mode.

To activate the learn mode just change the **`eoGateway::LearnMode`** to `true`, to deactivate it again, change it to `false.`

```
	myGateway.LearnMode=true;
```

Now we've prepared everything, we need to get to Receive Messages and handle TeachIN Telegrams.

```
while(learnTime>time)

{

	//updates the time, using the HAL getTickCount function
```

```
time=eoTimer::GetTickCount();

//the Gateway::Receive() Functions returns different flags,

//depending on the Packet we got

recv = myGateway.Receive();

//as we're in LearnMode currently we only want to

//process Teach_IN Telegrams(as specified in EEP)

if (recv & RECV_TEACHIN)

{

      //add the Source ID to the Normale Mode Filter

      myFilter->Add(myGateway.telegram.sourceID);

      //Print out the Message to stdout

      eoDebug::Print(myGateway.telegram);

      //If the TeachIN process was successfull and we got a Profile
      //print out the Message!

      eoProfile *profile = myGateway.device->GetProfile();

      if(profile!=NULL)

      {

            printf("Device %08X Learned-In EEP: %02X-%02X-%02X\n",
                  myGateway.device->ID,

                  profile->rorg,

                  profile->func,

                  profile->type );


            for (int i = 0; i<profile->GetChannelCount(); i++)

            {

                  printf("%s %.2f ... %.2f %s\n",

                        profile->GetChannel(i)->ToString(NAME),

                        profile->GetChannel(i)->min,
```

```
                                profile->GetChannel(i)->max,
                                profile->GetChannel(i)->ToString(UNIT));
                        }
                }
        }
}
```

When we leave this function, we want to switch to normal mode and store the current configuration of the `eoGateway` and all learned in devices!

To store an Object, which has the `ISerialize` interface implemented, you use the **`eoStorageManager`** and add Objects to handle to the Manager.

```
        myGateway.LearnMode=false;
        eoStorageManager myStore;
        myStore.addObject("Gateway",&myGateway);
        myStore.Save(SAVE_CONFIG);
```

Now, you have a text file, containing all the learned-in devices, which you can use for other applications.

This example application now just shows the incoming data of TeachIN devices, and if the profile is supported, the Values are printed in clear Text.

```
while (1)
{
        recv = myGateway.Receive();
        if (recv & RECV_MESSAGE)
        {
                //Resend received telegrams
```

```
            myGateway.Send(myGateway.message);
            eoDebug::Print(myGateway.telegram);
      }
      //If we got a valid Profile Telegram which is
      //not a Learn IN we print the received Values
      if (recv & RECV_PROFILE)
      {
            printf("Device %08X\n", myGateway.device->ID);
            eoProfile *profile = myGateway.device->GetProfile();
            float f;
            uint8_t t;
            for (int i = 0; i<profile->GetChannelCount(); i++)
            {
                  //get the channel value if it is a float
                  if (profile->GetValue(profile->GetChannel(i)->type,f) ==OK)
                        printf("%s %.2f %s\n",
                              profile->GetChannel(i)->ToString(NAME),
                              f,
                              profile->GetChannel(i)->ToString(UNIT));
                  //get the channel value if it is an uint
                  if (profile->GetValue(profile->GetChannel(i)->type,t) ==OK)
                        printf("%s %u \n",
                              profile->GetChannel(i)->ToString(NAME),
                              t);
            }
      }
}
```

```
return 0;
```

6.1 Output example

In the following example you can find output generated with the example source code. We used 4 devices:

- window contact, profile: D5-00-01
- switch, profile: F6-02-01
- temperature and humidity sensor, profile: A5-04-01
- smart-plug outlet – D2-08-01

3.

```
Opening Connection to USB300

EnOcean-Link Gateway LearnMode

RORG: A5 Data: 10 08 31 80 SrcID: 0088E0FB Status: 00 DstID: FFFFFFFF



Device 0088E0FB Learned-In EEP: A5-04-01

Relative humidity 0.00 ... 100.00 Percent

Temperature 0.00 ... 40.00 Deg C

RORG: D4 Data: A0 FF 15 00 08 01 D2 SrcID: 0081FBA4 Status: 00 DstID: FFFFFFFF



Device 00891AA8 Learned-In EEP: D5-00-01

Open (1) / closed (0)

Temperature 0.00 ... 40.00 Deg C

RORG: D5 Data: A0 FF 15 00 08 01 D2 SrcID: 00891AA8 Status: 00 DstID: FFFFFFFF



Device 0081FBA4 Learned-In EEP: D2-01-08

Dim value 0.00 ... 7.00 N/A

Input/Output channel 0.00 ... 31.00 N/A

Percentage 0.00 ... 100.00 %

On (1) / off (0) 0.00 ... 1.00 N/A

On (1) / off (0) 0.00 ... 1.00 N/A

On (1) / off (0) 0.00 ... 1.00 N/A
```

```
On (1) / off (0) 0.00 ... 1.00 N/A

On (1) / off (0) 0.00 ... 1.00 N/A

On (1) / off (0) 0.00 ... 1.00 N/A

Power 0.00 ... 4294967296.00 Watts

Power 0.00 ... 4294967296.00 Watts

Energy 0.00 ... 4294967296.00 J - W*s


Storing the Config

EnOcean-Link Gateway FilterdMode

=======================


RORG: F6 Data: 10 SrcID: 001571A5 Status: 20 DstID: FFFFFFFF

Device 001571A5

Rocker Button A 2

Rocker Button B 2

Energybow direction 1

Multiple Buttons pressed? 0

=======================


RORG: A5 Data: 00 6E A9 0A SrcID: 0088E0FB Status: 00 DstID: FFFFFFFF

Device 0088E0FB

Relative humidity 44.00 Percent

Temperature 27.04 Deg C

=======================


RORG: D2 Data: 04 00 E4 SrcID: 0081FBA4 Status: 00 DstID: FFFFFFFF

Device 0081FBA4
```

```
On (1) / off (0) 0

Error state 0

Input/Output channel 0

On (1) / off (0) 1

Percentage 100.00 %

=======================


RORG: D2 Data: 07 60 00 00 00 25 SrcID: 0081FBA4 Status: 00 DstID: FFFFFFFF

Device 0081FBA4

Input/Output channel 0

Power 37.00 Watts

=======================


RORG: 32 Data: 01 SrcID: 00891AA8 Status: 00 DstID: FFFFFFFF

Device 00891AA8

Open (1) / closed (0) 1

=======================
```

7 FOLLOW UP

To gain more information about EnOcean Link and try the Evaluation version we recommend you to:

- get the evaluation version or full version – you can find contact information and details on the EnOcean Link Product page – please see Reference 3.
- read the full user manual and specification, please see Reference 1, where you can find more tutorials and documentation of every class