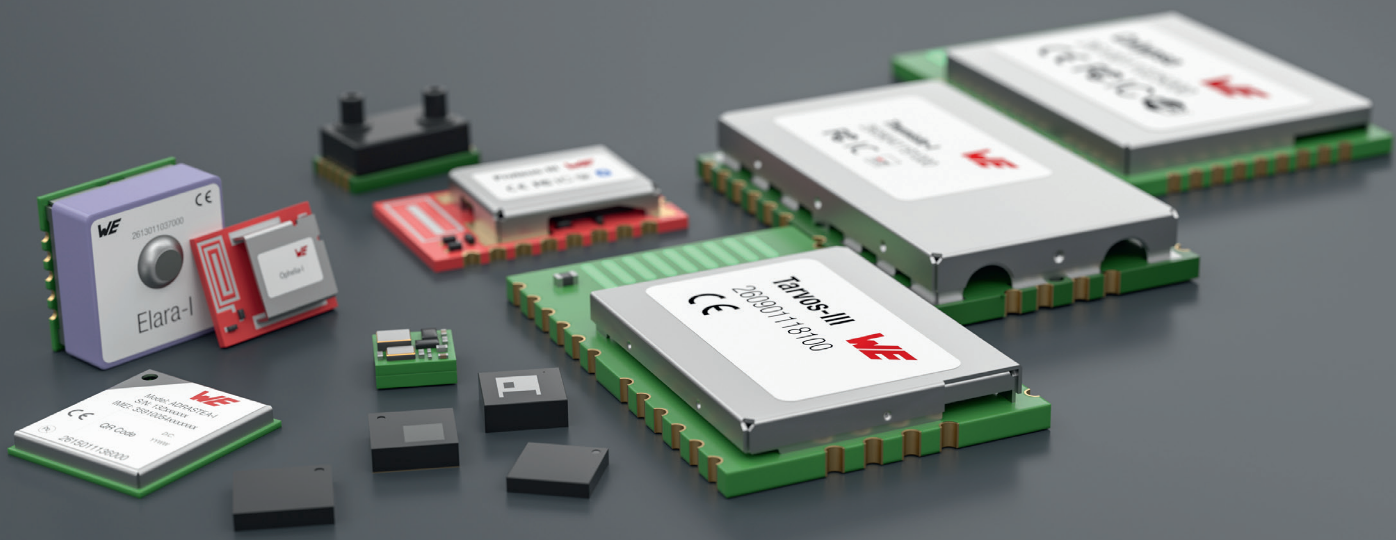

ANR008

Wireless Connectivity SDK

Version 1.15

March 17, 2025

WÜRTH ELEKTRONIK MORE THAN YOU EXPECT

Revision history

Manual version	Version for STM32	Notes	Date
1.0	-	• Initial version of this document	April 2019
1.2	-	• Updated file name to new application note name structure. Updated important notes, legal notice & license terms chapters.	June 2019
1.3	-	• Updated supported modules	September 2019
1.4	-	• Updated supported modules	May 2020
1.5	-	• Updated supported modules • Updated supported libraries • Updated installation instructions • Added description for SPI interfaces	February 2021
1.6	-	• Fixed wrong links • Updated installation description of FTDI drivers • Renamed USB radio sticks variants	August 2021
1.7	1.0.0	• Added new driver for STM32 micro controllers (see chapter 2). • Restructured chapters for better overview	August 2021

1.8	1.2.0	• The STM32 SDK is now the primary version of the SDK. The Raspberry Pi version has been archived. • Updated supported modules • Updated chapter 2 (Wireless Connectivity SDK)	May 2022
1.9	1.5.0	• Updated SDK history of STM32 version (chapter 2.3)	April 2023
1.10	1.7.2	• Wireless Connectivity SDK for Raspberry has been archived • Updated `Important notes` and meta data	July 2023
1.11	2.0.0	• Removed documentation Wireless Connectivity SDK for Raspberry Pi (legacy version) • Explained new structure of Wireless Connectivity SDK for STM32 of version 2.0.0	January 2024
1.12	2.1.0	• Updated chapter `Version history`	June 2024
1.13	2.2.0	• Updated chapter `Version history`	September 2024
1.14	2.3.0	• Updated chapter `Version history`	January 2025

1.15	2.4.0	• Updated chapter `Version history` and `Host integration`	March 2025

⋆ For SDK version history see chapter `Version history`

Abbreviations

Abbreviation	Name	Description
BDM	Business Development Engineer	Support and sales contact person responsible for limited sales area
CS	Check sum	
DC	Duty cycle	Active transmission time per hour expressed as percentage. 1% means, channel is occupied for 36 seconds per hour.
0xhh [HEX]	Hexadecimal	The prefix 0x indicates hexadecimal values. All other numbers are decimal values.
HAL	Hardware Abstraction Layer	
HIGH	High signal level	
LOW	Low signal level	
LPM	Low power mode	Operation mode with reduced energy consumption.
LRM	Long range mode	Tx mode increasing the RX sensitivity by using spreading and forward error correction
LSB	Least significant bit	
MSB	Most significant bit	
PL	Payload	The real, non-redundant information in a frame/packet.
RF	Radio frequency	Describes everything relating to the wireless transmission.
SDK	Software development kit	Software code that implements the command interface of various Würth Elektronik eiSos products
UART		Universal Asynchronous Receiver Transmitter - a serial data transmission interface
VDD	Supply voltage	

Contents

1 Introduction

The Würth Elektronik eiSos wireless modules provide an easy to use radio interface to any embedded application. The host processor of the embedded application can operate the module by sending commands via UART to the module's command interface.

The Wireless Connectivity SDK is a set of software tools that enable quick software integration of Würth Elektronik eiSos wireless modules into external host processors. It consists of examples and platform independent module drivers in C-code that use the UART to communicate with the attached radio device.

The Wireless Connectivity SDK has been developed and tested on the STM32 platform, but is designed in a way that it can be easily ported to any other host processor platform. The C-code of examples, command interface definition and platform dependant functions are separated from each other.

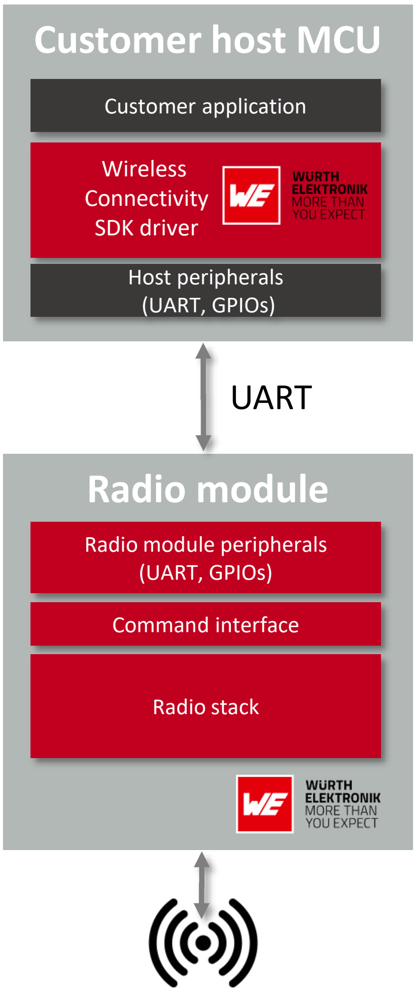

Figure 1: Wireless connectivity SDK driver as part of the end product

1.1 Motivation

The aim of the Wireless Connectivity SDK is to minimize the effort required on customer side to enable his host MCU to communicate with Würth Elektronik eiSos radio modules. It contains the implementation of all available commands in pure C-code. In order to integrate any Würth Elektronik eiSos wireless module, the user has to simply copy the corresponding C-code and adapt the platform specific parts to his host processor (HAL).

Würth Elektronik eiSos products, like the 868 MHz proprietary radio module Tarvos-III, use a so called **command interface**[1] for configuration and operation tasks. This interface provides up to 30 commands that accomplish tasks like updating various device settings, transmitting/receiving data and putting the module into one of various low power modes.
The commands of the interface can be divided into 3 categories:

1. Requests: The host requests the module to trigger an action, e.g. in case of the request `CMD_RESET_REQ` the host asks the module to perform a reset.
2. Confirmations: On each request the module answers with a confirmation message as a feedback on the requested operation status. In case of a `CMD_RESET_REQ`, the module answers with a `CMD_RESET_CNF` to tell the host whether it is ready to run the reset process or not.
3. Indications and Responses: In case of special events, the module indicates the same spontaneously to the host. The `CMD_DATAEX_IND` indicates for example that data has been received via radio.

All command messages (requests, confirmations, events) have the following format:

Start byte	Command	Length	Payload	CS
0x02	1 Byte	1 Byte	Length Bytes	1 Byte

Example: CMD_DATA_REQ of the Tarvos-III
The `CMD_DATA_REQ` has the command number 0x00. It provides a simple data transfer. The length field indicates the number of bytes to be transmitted via radio.

Format:

Start byte	Command	Length	Payload	CS
0x02	0x00	1 Byte	Length Bytes	1 Byte

Example: Sending "Hello World!"

[1] There are Würth Elektronik eiSos wireless modules that support a second operation mode, the so called transparent mode. When using the transparent mode the device does not accept commands sent via UART. Please make sure that the connected radio device runs in command mode as specified in the respective radio module user manual, when using the Wireless Connectivity SDK.

Start byte	Command	Length	Payload	CS
0x02	0x00	0x0C	0x48 0x65 0x6C 0x6C 0x6F 0x20 0x57 0x6F 0x72 0x6C 0x64 0x21	0x0F

With the above command, we are sending 12 bytes (0x0C), corresponding to the ASCII string "Hello World!" (0x48 0x65 0x6C 0x6C 0x6F 0x20 0x57 0x6F 0x72 0x6C 0x64 0x21) and the resulting checksum is 0x0F.

To use the complete feature set of such a radio device, all available commands of the corresponding command interface have to be implemented on the custom host processor. This involves considerable effort for the user. To avoid that effort Würth Elektronik eiSos offers the Wireless Connectivity SDK, which provides the implementation of the command interface.

2 Wireless Connectivity SDK

2.1 Content and structure

The radio modules supported by the latest version of the Wireless Connectivity SDK are:

SDK version	Radio standard	Radio module & USB dongle
2.4.0	Bluetooth® / Bluetooth® LE combo	Skoll-I
	Bluetooth® LE	Proteus-II, Proteus-III, Proteus-e
	Cellular	Adrastea-I
	LoRa WAN	Daphnis-I
	Proprietary 868 MHz	Tarvos-III, Thebe-II, Thebe-II-IND, Tarvos-e
	Proprietary 915 MHz	Telesto-III, Themisto-I
	Proprietary 2.4 GHz	Thyone-I, Thyone-e
	WiFi / WLAN	Calypso, Cordelia-I
	WiFi / Bluetooth® LE combo	Stephano-I
	Wireless M-BUS	Metis-e, Metis-II, Metis-I, Mimas-I

Table 1: Wireless Connectivity SDK for STM32

The driver is designed in a way, that several modules of different type can be run at the same time. See also the "multi module" examples present in the Wireless Connectivity SDK.

At the topmost level, the Wireless Connectivity SDK directory structure looks as follows.

```
/
└─ WCON_SDK ........................................ Root directory of the SDK
   ├─ Examples ................................ Wireless modules' example codes
   ├─ global .............................................. STM32 shared functions
   ├─ STM32F4xx .................. STM32F4 driver (HAL, peripherals, ...) and project file
   ├─ STM32L0xx .................. STM32L0 driver (HAL, peripherals, ...) and project file
   ├─ WCON_Drivers ...................... Platform independent wireless module drivers
   └─ Readme.md .................................................... Readme file
```

Figure 2: Folder structure

The root directory of the Wireless Connectivity SDK (**WCON_SDK**) contains the STM32 shared functions and HAL driver directories for each supported family of STM32 micro controllers (STM32F4xx and STM32L0xx) as well as the **WCON_Drivers** and **Examples** directory. Within the **WCON_Drivers** directory, each supported radio module has its own subdirectory containing the **platform independent** implementation of its command interface. In the **Examples** directory you can find many application examples for all of the radio modules, separated in module specific subdirectories as well.

```
/
├─ WCON_Drivers ........................ Platform independent module drivers
│  ├─ ProteusIII ..................... Command interface of the Proteus-III module
│  │  ├─ ProteusIII.h
│  │  └─ ProteusIII.c
│  ├─ TarvosIII ...................... Command interface of the Tarvos-III module
│  │  ├─ TarvosIII.h
│  │  └─ TarvosIII.c
│  └─ ...
├─ Examples
│  ├─ main.c ............ main file linking to the respective examples of each module
│  ├─ ProteusIII
│  │  └─ ProteusIII_Examples.c ............ Example of the Proteus-III module
│  ├─ TarvosIII
│  │  └─ TarvosIII_Examples.c ............. Example of the Tarvos-III module
│  └─ ...
├─ global ....................... Contains the code to be ported to custom hosts
└─ ...
```

Figure 3: Folder structure

2.2 Host integration

As described in chapter 2.1, the Wireless Connectivity SDK provides implementations for the STM32L0 and STM32F4 platforms. In order to use the drivers with a different type of host micro controller, they have to be ported to the new platform. This is shown in the following example, on base of the Proteus-III drivers and example.

1. The directories **WCON_Drivers/global** and **WCON_Drivers/ProteusIII** have to be copied to the custom project.

2. The file **global_types.h** declares global types used by the drivers. The file **global.h** declares the shared functions that deal with delay- and GPIO-related functions.

```
/**
* @brief Switch pin to output high/low
*
* @param[in] pin Output pin to be set
* @param[in] out Output level to be set
* @return true if request succeeded, false otherwise
*/
extern bool WE_SetPin(WE_Pin_t pin, WE_Pin_Level_t out);

/**
* @brief Sleep function.
*
* Note that WE_MICROSECOND_TICK needs to be defined to enable microsecond timer
    resolution.
*
* @param[in] sleepForUsec Delay in microseconds
*/
extern void WE_DelayMicroseconds(uint32_t sleepForUsec);
```

Code 1: Code snippet of the file global.h

When integrating the Wireless Connectivity SDK into a custom host, the implementation of the functions defined in **global.h** must be done in the **global.c** file.

3. Furthermore, the files **global_L0xx.h/c** and **global_F4xx.h/c** contain the UART implementation of the respective STM32 platform. For the custom host a source and header file containing the respective UART functions must be be created.

```
extern void WE_SystemClock_Config(void);

/**
* @brief Initialize and start the UART.
*
* @param[in] baudrate Baud rate of the serial interface
* @param[in] flowControl Enable/disable flow control
* @param[in] parity Parity bit configuration
* @param[in] rxByteHandlerP Pointer to the handle rx byte function inside the
    driver
*/
extern bool WE_UART1_Init(uint32_t baudrate,
WE_FlowControl_t flowControl,
WE_Parity_t parity,
WE_UART_HandleRxByte_t *rxByteHandlerP);
```

```
/**
* @brief Deinitialize and stop the UART.
*/
extern bool WE_UART1_DeInit();

/**
* @brief Transmit data via UART.
*
* @param[in] data Pointer to data buffer (data to be sent)
* @param[in] length Number of bytes to be sent
*/
extern bool WE_UART1_Transmit(const uint8_t *data, uint16_t length);
```

Code 2: Functions needed for custom host

4. Last, in the example code the GPIO pins used by the module driver are defined, before the module's init function is called (here: **ProteusIII_Examples()** in file **ProteusIII_Examples.c**).

```
static ProteusIII_Pins_t ProteusIII_pins = {
    .ProteusIII_Pin_Reset = WE_PIN((void*)&WE_STM32_PIN(GPIOA, GPIO_PIN_10)),
    .ProteusIII_Pin_SleepWakeUp = WE_PIN((void*)&WE_STM32_PIN(GPIOA, GPIO_PIN_9)),
    .ProteusIII_Pin_Boot = WE_PIN((void*)&WE_STM32_PIN(GPIOA, GPIO_PIN_7)),
    .ProteusIII_Pin_Mode = WE_PIN((void*)&WE_STM32_PIN(GPIOA, GPIO_PIN_8)),
    .ProteusIII_Pin_Busy = WE_PIN((void*)&WE_STM32_PIN(GPIOB, GPIO_PIN_8)),
    .ProteusIII_Pin_StatusLed2 = WE_PIN((void*)&WE_STM32_PIN(GPIOB, GPIO_PIN_9)),
};

/**
* @brief Definition of the uart
*/
static WE_UART_t ProteusIII_uart;

/**
* @brief The application's main function.
*/
void ProteusIII_Examples(void)
{
    ProteusIII_uart.baudrate = PROTEUSIII_DEFAULT_BAUDRATE;
    ProteusIII_uart.flowControl = WE_FlowControl_NoFlowControl;
    ProteusIII_uart.parity = WE_Parity_None;
    ProteusIII_uart.uartInit = WE_UART1_Init;
    ProteusIII_uart.uartDeinit = WE_UART1_DeInit;
    ProteusIII_uart.uartTransmit = WE_UART1_Transmit;
ProteusIII_Init(&ProteusIII_uart, &ProteusIII_pins,
    ProteusIII_OperationMode_CommandMode, callbackConfig);
```

Code 3: Selection of GPIO and UART

After dealing with these steps the driver is functional.

2.3 Version history

Version 1.0.0 "Release"

- Initial version of the SDK

Version 1.1.0 "Release"

- Added driver for the Calypso WiFi radio module

Version 1.2.0 "Release"

- Added driver for the Proteus-e Bluetooth LE module
- Updated Proteus-III driver

Version 1.3.0 "Release"

- Improvement of the Thyone-I driver

Version 1.4.0 "Release"

- Improvement of the Calypso driver

Version 1.5.0 "Release"

- Improvement of the Proteus-III driver
- Added Proteus-II driver
- Restructured AT-command codes for coming radio modules

Version 1.7.0 "Release"

- Added driver for Adrastea-I

Version 2.0.0 "Release" January 2024

- Added driver for Daphnis-I, Stephano-I
- Changed driver structure to support multiple modules at the same time
- Implemented a second UART interface on STM32F4xxxx and STM32L0xxxx
- Added example for multi module support

Version 2.1.0 "Release" June 2024

- Added functions of firmware version 3.3.0 of radio modules Tarvos-III, Telesto-III, Thebe-II, Themisto-I
- Bug fix in writing user settings of Proteus and Thyone variants
- Bug fix in Stephano-I driver
- Improvements in UART definition and debug interface for STM32

Version 2.2.0 "Release" September 2024

- Added products Thebe-II-IND and Metis-e
- Final release Thyone-e
- Updated base64 implementation
- Bugfix Adrastea-I MQTT commands

- Bugfix Stephano-I Bluetooth® LE command

Version 2.3.0 "Release" January 2025

- Added products Cordelia-I, Tarvos-e and Skoll-I
- Added support for Daphnis-I firmware version 1.4.0
- Cleaned debug and release configuration
- Restructured directories to make drivers platform independent
- Code improvements
  - Moved Adrastea-I, Stephano-I and Calypso event definitions from RAM to flash
  - Bugfix in Adrastea-I driver dealing with empty strings
  - Bugfix in Calypso-I file transfer driver
  - Metis-e is no longer resetted when driver initialized. This needs to be done in the application.
  - Bugfix in wake-up function of Tarvos-III/Telesto-III/Thebe-II/Themisto-I

Version 2.4.0 "Release" March 2025

- Set default Daphnis-I support to firmware version 1.4.0
- Generalized definition of MCU pins
- Applied newest code style guidelines

3 Running sample applications on the STM32 Nucleo board

3.1 Hardware connections

The Wireless Connectivity SDK has been developed on the STM32 Nucleo-L073RZ [1] and Nucleo-F401RE [2] development boards using STM32CubeIDE [3]. To run the examples provided by the SDK on one of these boards, first connect the respective pins of the board to the radio module[2].
Besides the *VCC*, *GND* and the *UART* pins, other product-specific pins such as the */Reset* pin need to be connected. Consult the Wireless Connectivity SDK Readme file and/or the user manual of the used wireless module for additional information.

> Please note that table 2 and figure 4 show a standard set of GPIOs used by most of the Würth Elektronik eiSos wireless modules. Depending on the radio module and the application being developed, it may be necessary to make different connections.

Function	Pin	I/O	Function	Pin	I/O
VCC	+3V3	-	GND	GND	-
UART RX	PB7	Input	UART CTS	PA11	Input
UART TX	PB6	Output	UART RTS	PA12	Output
Reset	PA10	Output	Wake_up	PA9	Output
Boot	PA7	Output	Mode	PA8	Output

Table 2: Used pins of the STM32 Nucleo board

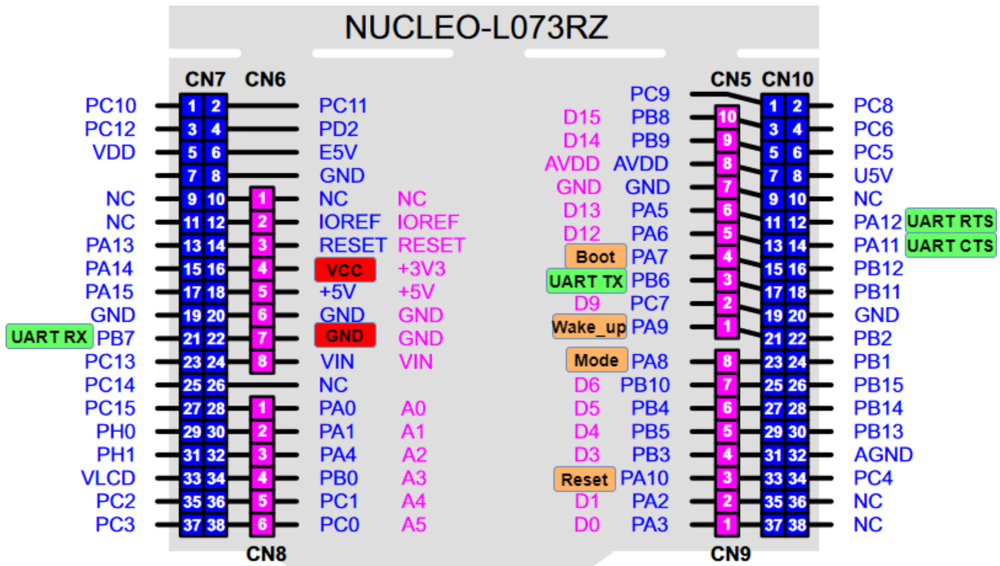

Figure 4: Layout for STM32 NUCLEO-L073RZ [4]

[2]All needed radio module pins are available for connection on the corresponding EV-Board.

3.2 Run the project with STM32CubeIDE

First download the Wireless Connectivity SDK [5] and save it on your computer. Then install and start the STM32CubeIDE [3] (recommended version 1.14.0 or higher). When starting, you will be asked for a workspace path. In case you already have a workspace from previous projects, select the desired path. In case you want to use a new workspace, create a new empty directory and select that directory.

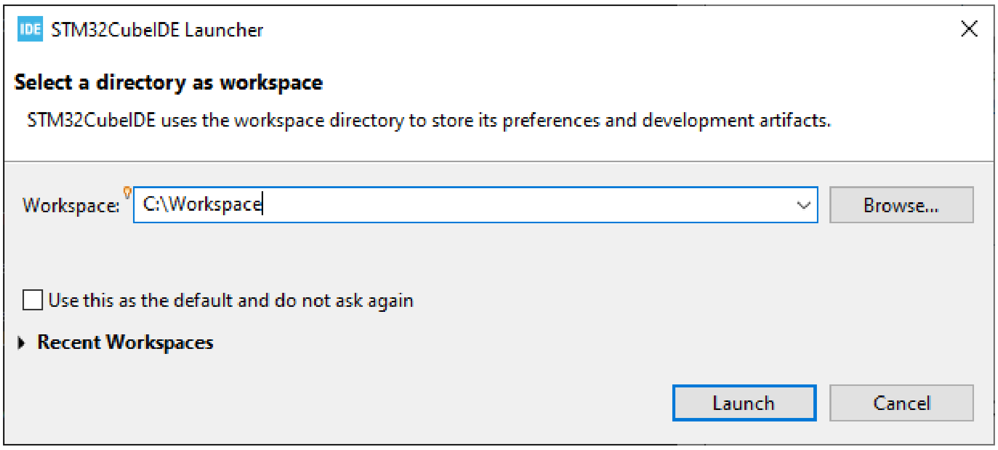

Figure 5: Choose workspace

After the STM32CubeIDE started, go to "File→Open Projects from File System" and select the path where you saved the Wireless Connectivity SDK.

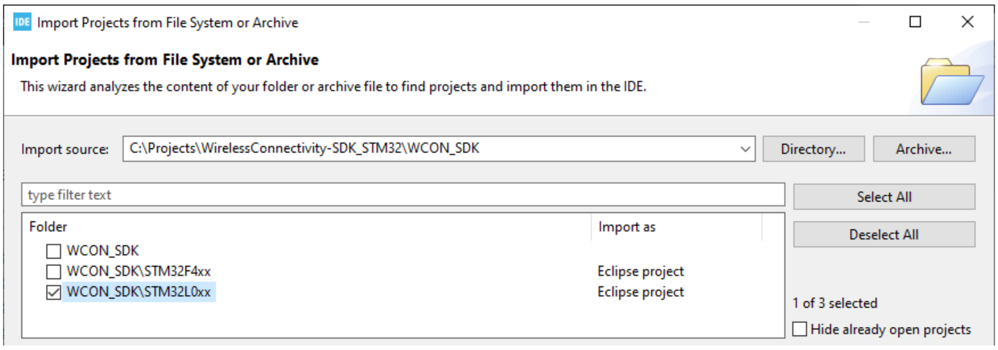

Figure 6: Open project

Select the project which you would like to import depending on the microcontroller family that you intend to use (e.g. STM32L0xx). Press the "Finish" button, so that the project is loaded into the "Project Explorer".

The Wireless Connectivity SDK includes the examples and drivers for all supported products in one STM32CubeIDE project. Thus to evaluate a single product, the right radio module example must be selected first. To do so, open the **main.c** file and comment/uncomment the example functions.

```
/* select the example to run */
```

```
    //Adrastea_Examples();
    //Calypso_Examples();
    //DaphnisI_Examples();
    //Metis_Examples();
    //ProteusE_Examples();
    //ProteusII_Examples();
    ProteusIII_Examples();
    //StephanoI_Examples();
    //ThyoneE_Examples();
    //ThyoneI_Examples();
    //TarvosIII_Examples();
    //TelestoIII_Examples();
    //ThebeII_Examples();
    //ThemistoI_Examples();
    //MultiModule_ProteusIII_TarvosIII_Examples();
```

Code 4: Select example in main.c file

Then press "Build" and "Run" to flash the firmware on the STM32 controller. If the debug output is enabled by defining the macro "WE_DEBUG", you can see the debug messages on the serial interface (115200 Baud, 8n1), that is available on the USB connector of the Nucleo board.

4 References

[1] STMicroelectronics. NucleoL073RZ. https://www.st.com/en/evaluation-tools/nucleo-l073rz.html.

[2] STMicroelectronics. NucleoF401RE. https://www.st.com/en/evaluation-tools/nucleo-f401re.html.

[3] STMicroelectronics. STM32CubeIDE. https://www.st.com/en/development-tools/stm32cubeide.html.

[4] STMicroelectronics . UM1724 - User manual - STM32 Nucleo-64 boards (MB1136). https://www.st.com/en/evaluation-tools/stm32-nucleo-boards.html.

[5] Würth Elektronik. Wireless Connectivity SDK for STM32 - Radio module drivers in C-code. https://github.com/WurthElektronik/WirelessConnectivity-SDK_STM32.

5 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

List of Tables

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

WÜRTH ELEKTRONIK MORE THAN YOU EXPECT