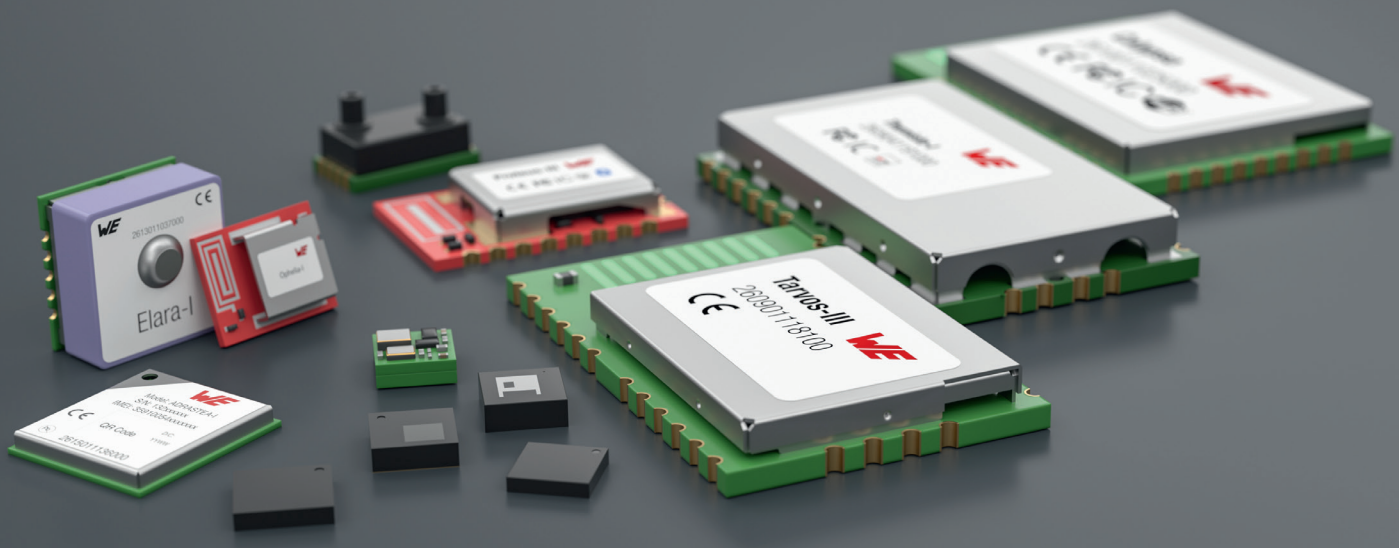

ANR008

Wireless Connectivity SDK

Version 1.9

April 3, 2023

WÜRTH ELEKTRONIK MORE THAN YOU EXPECT

Revision history

Manual version	Version for STM32	Version for Raspb. Pi	Notes	Date
1.0	-	3.0.0	• Initial version of this document	April 2019
1.2	-	3.0.0	• Updated file name to new AppNote name structure. Updated important notes, legal notice & license terms chapters.	June 2019
1.3	-	3.1.0	• Updated supported modules	September 2019
1.4	-	3.2.0	• Updated supported modules	May 2020
1.5	-	3.3.0	• Updated supported modules • Updated supported libraries • Updated installation instructions • Added description for SPI interfaces	February 2021
1.6	-	3.4.0	• Fixed wrong links • Updated installation description of FTDI drivers • Renamed USB radio sticks variants	August 2021

1.7	1.0.0	3.4.0	• Added new driver for STM32 microcontrollers (see chapter 2). • Restructured chapters for better overview	August 2021
1.8	1.2.0	3.4.0	• 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 The Raspberry Pi version has been archived.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 for STM32)	May 2022
1.9	1.5.0	3.4.0	• Updated SDK history of STM32 version (chapter 2.2)	April 2023

⋆ For SDK version history see chapter 2.2 (for STM32) and chapter 3.2 (for Raspberry Pi).

Abbreviations

Abbreviation	Name	Description
CS	Checksum	
DC	Duty cycle	Active transmission time per hour expressed as percentage. 1% means, channel is occupied for 36 seconds per hour.
FSE	Field Sales Engineer	Support and sales contact person responsible for limited sales area
0xhh [HEX]	Hexadecimal	The prefix 0x indicates hexadecimal values. All other numbers are decimal values.
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 module's interface with the host processor of the embedded application via UART can be operated using a command interface.

The Wireless Connectivity SDK is a set of software tools that enable quick software integration of Würth Elektronik eiSos wireless modules into any of the most commonly used host processors. It consists of drivers and examples in C-code that use the UART, SPI or USB peripheral of the underlying platform to communicate with the attached radio device.

The Wireless Connectivity SDK has been developed for the STM32 (see chapter 2) and Raspberry Pi 3 (see chapter 3) platforms.

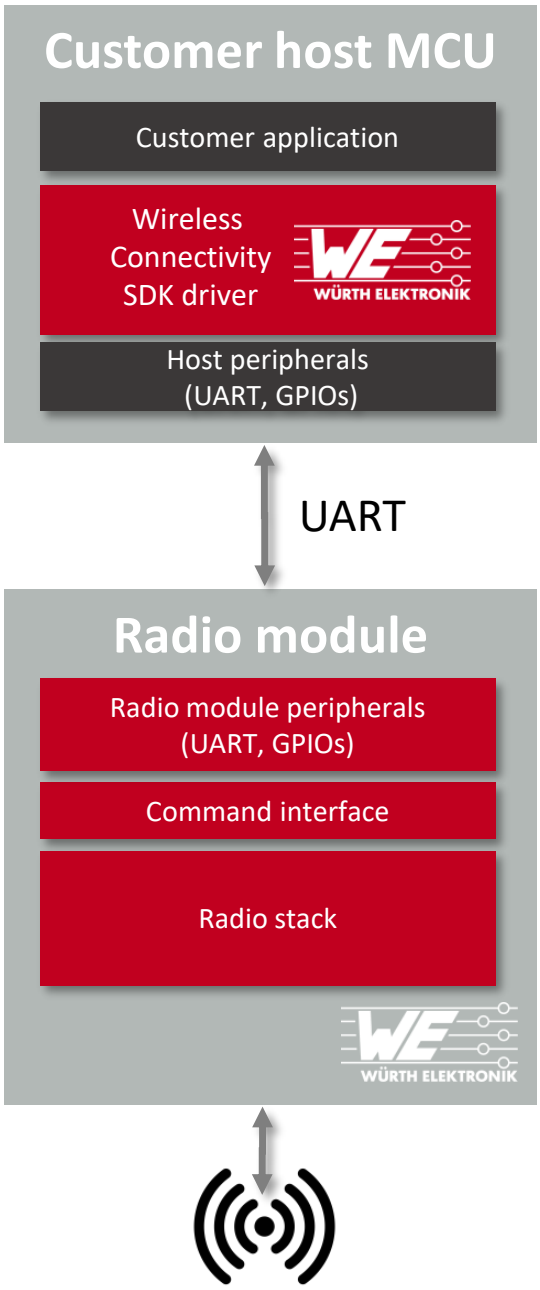

Figure 1: Wireless connectivity SDK driver as part of the end product

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 port the corresponding C-code to his host processor. This significantly reduces the time needed for developing the software interface to the radio module.

Würth Elektronik eiSos products, like the 868MHz proprietary radio module Tarvos-III, use a so called **command interface** for configuration and operation tasks. This interface provides up to 30 commands that accomplish tasks like updating various device settings, transmit/receive data and putting the module into one of various low power modes.

There are Würth Elektronik eiSos wireless modules that can operate in transparent mode in addition to the standard command mode. When using the transparent mode, the device does not interpret the commands sent via UART. Please make sure that the connected radio device runs in command mode to use the Wireless Connectivity SDK.

The commands of such an 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 the reset will be performed 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 was received via radio.

The commands itself 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

Sending "Hello World!"

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 and this is exactly the reason why Würth Elektronik eiSos offers the Wireless Connectivity SDK.

The steps for porting are explained in more detail in chapter 2.3 and 3.3.

2 Wireless Connectivity SDK for STM32

2.1 Content

The radio modules supported by the latest version of the Wireless Connectivity SDK for STM32 are:

SDK version	Radio standard	Radio module & USB dongle
1.2.0	Bluetooth® LE	Proteus-III, Proteus-e
	Proprietary 868 MHz	Tarvos-III, Thebe-II
	Proprietary 915 MHz	Telesto-III, Themisto-I
	Proprietary 2.4 GHz	Thyone-I
	Wi-Fi / WLAN	Calypso
	Wireless M-BUS	Shared driver for Metis-II, Metis-I, Mimas-I

Table 1: Wireless Connectivity SDK for STM32

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

```
/
└─ WCON_SDK ..... Root directory of the SDK
   ├─ STM32F4xx ..... STM32F4 driver (HAL, peripherals, ...) and project file
   ├─ STM32L0xx ..... STM32L0 driver (HAL, peripherals, ...) and project file
   ├─ WCON_Drivers ..... Wireless module drivers and example code
   └─ Readme.md ..... Readme file
```

The root directory of the Wireless Connectivity SDK (**WCON_SDK**) contains driver directories for each supported family of STM32 microcontrollers (STM32F4xx and STM32L0xx) as well as the **WCON_Drivers** directory, which contains all code related to the Würth Elektronik eiSos wireless modules. Within this directory, each supported radio module has its own subdirectory containing the implementation of its command interface and a main file with a short example application.
Besides the module-specific directories, the subdirectory **global** contains all the shared functions as well as definitions related to the serial communication and GPIO interfaces of the underlying host.
The wired communication interface currently supported by the Wireless Connectivity SDK for STM32 is UART. The SPI interface (STM32 role Master) is currently not supported.

```
/
├── WCON_Drivers ..... Contains the code to be ported to custom hosts
│   ├── global
│   │   ├── global_types.h ..... Declares global types used by the drivers
│   │   ├── global.h ..... Declares functions to be defined on custom hosts
│   │   ├── global.c ..... Implements functions to be defined on custom hosts
│   │   ├── global_F4xx.c ..... STM32F4-specific implementations of global functions
│   │   └── global_L0xx.c ..... STM32L0-specific implementations of global functions
│   ├── ...
│   ├── ProteusIII ..... Command interface of the Proteus-III module
│   │   ├── ProteusIII.h
│   │   ├── ProteusIII.c
│   │   └── main.c ..... Example of the Proteus-III module
│   ├── TarvosIII ..... Command interface of the Tarvos-III module
│   │   ├── TarvosIII.h
│   │   ├── TarvosIII.c
│   │   └── main.c ..... Example of the Tarvos-III module
│   └── ...
└── ...
```

2.2 Software history

Version 1.0.0 "Release"

- Initial version of the SDK

Version 1.1.0 "Release"

- Added driver for the Calypso Wi-Fi 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

2.3 Host integration

As described in chapter 2.1, the Wireless Connectivity SDK provides implementations for the STM32L0 and STM32F4 platforms. In order to use all the features of the radio module with a different type of host microcontroller, the module's drivers in the SDK have to be ported to the new custom platform.

In the following example, the steps required to port the drivers of the Bluetooth® LE 5.1 radio module Proteus-III to a custom platform are described.

- The directories **WCON_Drivers/global** and **WCON_Drivers/ProteusIII** have to be integrated into the custom project.
- The file **global_types.h** declares global types used by the drivers. It might be necessary to modify some parts of this file, such as port field of the WE_Pin_t structure.
- The file **global.h** declares the shared functions that deal with the serial interface as well as the usage of GPIOs for pin-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 Initialize and start the UART.
 *
 * @param[in] baudrate Baud rate of the serial interface
 * @param[in] flowControl Enable/disable flow control
 * @param[in] par Parity bit configuration
 * @param[in] dma Enables DMA for receiving data
 */
extern void WE_UART_Init(uint32_t baudrate,
                         WE_FlowControl_t flowControl,
                         WE_Parity_t par,
                         bool dma);
```

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.

- Note that for reasons of clarity, part of the platform-specific functionality for the supported STM32 microcontroller families is implemented in the **global_L0xx.c** and **global_F4xx.c** files.
- The pins used by the module driver are defined in the driver's main initialization function (here: **ProteusIII_Init()** in file **ProteusIII.c**). When integrating the Wireless Connectivity SDK into a custom host, this implementation may be adapted.

```
ProteusIII_pins[ProteusIII_Pin_Reset].port = GPIOA;
ProteusIII_pins[ProteusIII_Pin_Reset].pin = GPIO_PIN_10;
```

```
ProteusIII_pins[ProteusIII_Pin_Reset].type = WE_Pin_Type_Output;
    ...
ProteusIII_pins[ProteusIII_Pin_Busy].port = GPIOB;
ProteusIII_pins[ProteusIII_Pin_Busy].pin = GPIO_PIN_8;
ProteusIII_pins[ProteusIII_Pin_Busy].type = WE_Pin_Type_Input;
    ...
if (false == WE_InitPins(ProteusIII_pins, ProteusIII_Pin_Count))
{
    /* error */
    return false;
}
WE_SetPin(ProteusIII_pins[ProteusIII_Pin_Boot], WE_Pin_Level_High);
WE_SetPin(ProteusIII_pins[ProteusIII_Pin_SleepWakeUp], WE_Pin_Level_High);
```

Code 2: Pin initialization in ProteusIII.c (snippet)

After dealing with these steps the driver is functional. The corresponding demo project can be considered as a basis for application development on the custom platform.

2.4 Running sample applications on the STM32 Nucleo board

2.4.1 Hardware connections

The Wireless Connectivity SDK for STM32 has been developed on the STM32 Nucleo-L073RZ [5] and Nucleo-F401RE [4] development boards using STM32CubeIDE [6]. 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 (typically mounted on the corresponding evaluation board). Besides the VCC, GND and the UART pins, other product-specific pins such as the Reset pin should 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 2 show a standard set of GPIOs used by most of the Würth Elektronik eiSos wireless modules. Depending on the used 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

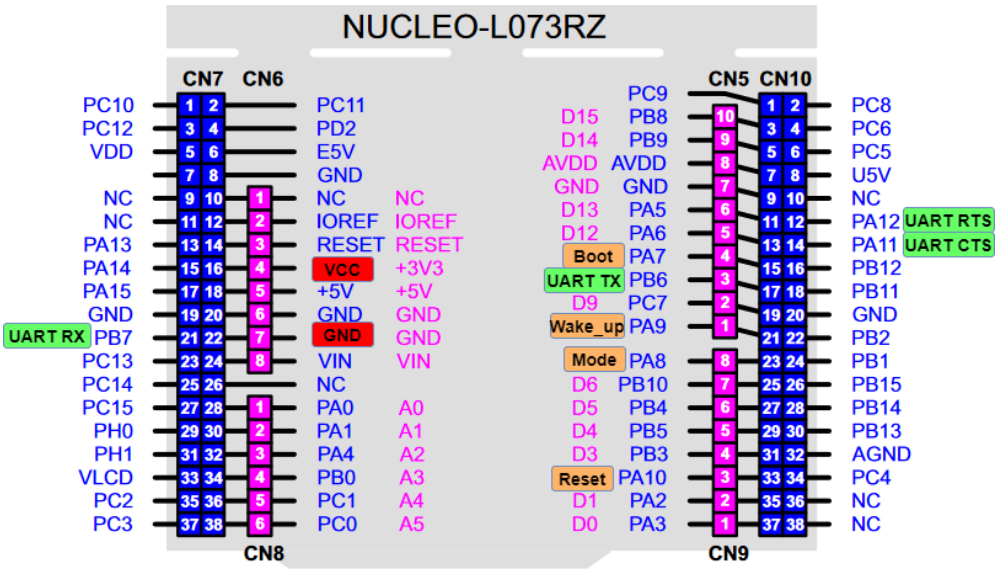

Figure 2: Layout for STM32 NUCLEO-L073RZ (source: UM1724 [3])

2.4.2 Run the project with STM32CubeIDE

First download the Wireless Connectivity SDK for STM32 [9] and save it on your computer. Then install and start the STM32CubeIDE [6] (recommended version 1.9.0 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.

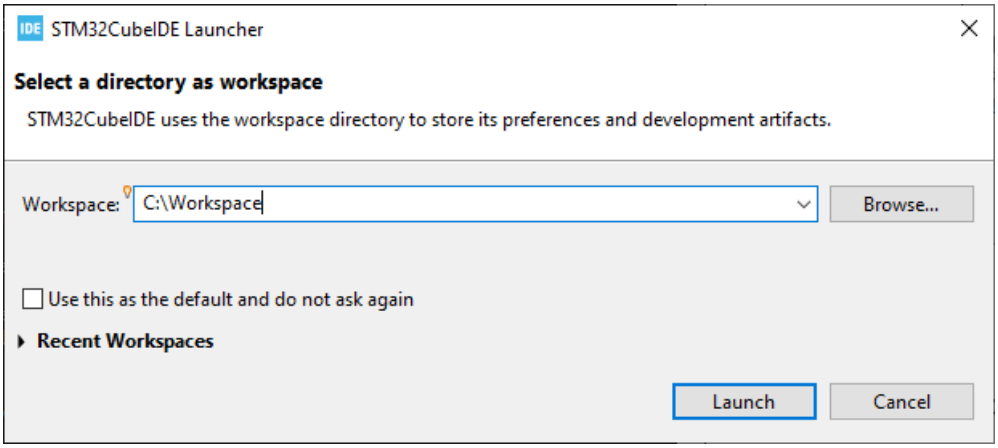

Figure 3: Choose workspace

After the STM32CubeIDE started, go to "File→Open Projects from File System" and select the path where you saved the Wireless Connectivity SDK for STM32.

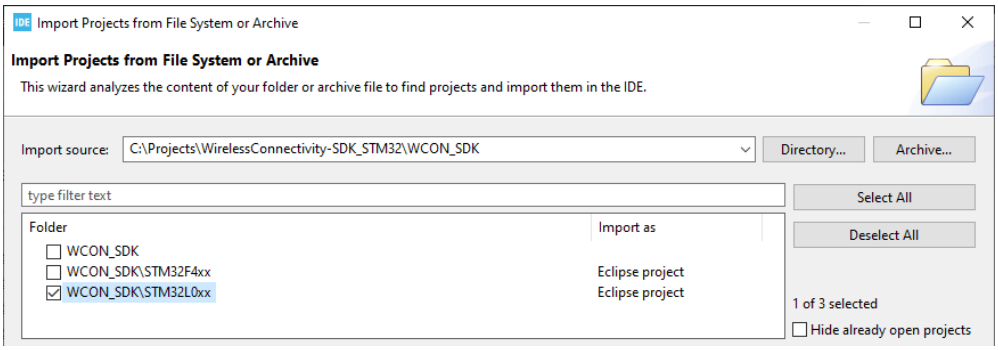

Figure 4: Open project

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 for STM32 includes the examples and drivers for all supported products in one STM32CubeIDE project. Thus to evaluate a single product, the right radio module files have to be selected first.
In the example shown in figure 5 the radio module Tarvos-III is selected. In case Proteus-III shall be chosen instead of Tarvos-III, first right click on the Tarvos-III directory, then select "Properties" and check the box "Exclude resource from build". Then do the same for Proteus-III, but uncheck the "Exclude resource from build" box.

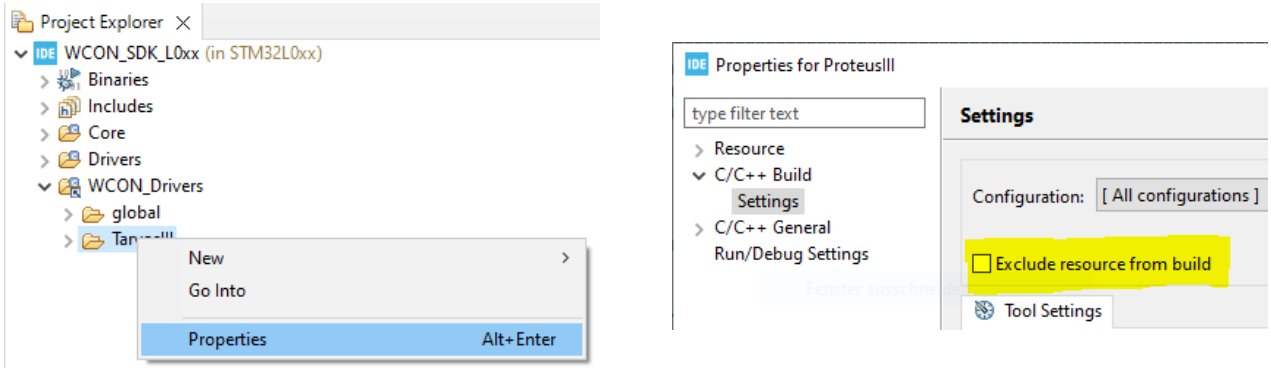

Figure 5: Select module

Then click on "Build" and check whether the project builds without errors. In case of success, click on "Debug" to run the project on the STM32 Nucleo board.

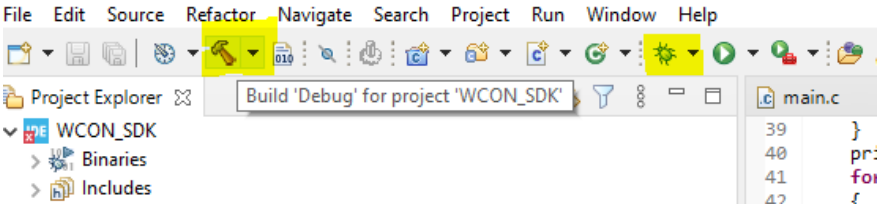

Figure 6: Build and run

3 Wireless Connectivity SDK for Raspberry Pi

The Wireless Connectivity SDK for Raspberry Pi has been archived and will not be developed any further. Usage of the SDK for new products is discouraged. The final version of the SDK is 3.4.0.

3.1 Content

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

SDK version	Radio standard	Radio module & USB dongle
3.4.0	Bluetooth® LE	Proteus-I, Proteus-II, Proteus-II USB radio stick, Proteus-III, Proteus-III USB radio stick, Proteus-III-SPI
	Proprietary 169 MHz	Titania
	Proprietary 434 MHz	Thadeus
	Proprietary 868 MHz	Tarvos-I, Tarvos-I USB radio stick, Tarvos-II, Tarvos-II USB radio stick, Tarvos-III, Tarvos-III USB radio stick, Thebe-I, Thebe-II
	Proprietary 915 MHz	Telesto-I, Telesto-II, Telesto-III, Telesto-III USB radio stick, Themisto-I
	Proprietary 2.4 GHz	Triton, Thalassa, Thalassa USB radio stick. Thyone-I, Thyone-I USB radio stick
	Wi-Fi / WLAN	Calypso
	Wireless M-BUS	Metis-II, Metis-II USB radio stick, Metis-I, Metis-I USB radio stick, Mimas-I, Mimas-I USB radio stick

Table 3: Wireless Connectivity SDK for Raspberry Pi

Besides the various sample projects, there is a directory named **drivers**. Within the **driver** directory, each supported radio module has its own directory containing the implementation of its command interface. Additionally to the definition of the commands, a thread is defined that checks for the confirmation and indication messages that are transmitted from the radio module to the host. Furthermore, functions that use specific radio module pins, like a pin reset or pin wake-up, are defined here.

Besides the module-specific directories, the subdirectory **global** contains all the shared functions as well as the definitions of the serial communication and GPIO interfaces of the underlying host.

```
/
├── drivers ..... Contains the code to be ported to custom hosts
│   ├── global
│   │   ├── global.h ..... Declares all functions to be defined on custom hosts
│   │   ├── global.c ..... Implements shared functions
│   │   ├── global_ftdi.c ..... UART and GPIO of the FTDI USB driver for RPi
│   │   ├── global_serial.c ..... UART of the RPi using termios
│   │   ├── global_serialWiringPi.c ..... UART of the RPi using wiringPi library
│   │   ├── global_spi.c ..... SPI interface of the RPi using ioctl
│   │   ├── global_pin.c ..... GPIO interface of the RPi using the libgpiod library
│   │   └── global_pinWiringPi.c ..... GPIO interface of the RPi using the wiringPi library
│   ├── ...
│   ├── Triton ..... Command interface of the Triton module
│   │   ├── Triton.h
│   │   └── Triton.c
│   ├── ...
│   ├── ThebeI ..... Command interface of the Thebe-I module
│   │   ├── ThebeI.h
│   │   └── ThebeI.c
│   └── ...
├── Example_Triton ..... Demo project using Triton module
│   ├── main.c
│   └── Example_Triton.cbp
├── ...
├── Example_ThebeI ..... Demo project using Thebe-I module
│   ├── main.c
│   └── Example_ThebeI.cbp
└── ...
```

The **global** directory contains various implementations of the serial and GPIO interface. Even though they are implemented with the Raspberry Pi platform in mind they can be used as a starting point for other platforms or even used on various Linux-based systems.
Within the Code::Blocks projects the implementations can be selected by means of the build target.

Build target	Interface	API/Library	Comments
Debug Release	UART	termios	For modules using UART
	GPIO	libgpiod	For modules using UART
DebugWiringPi ReleaseWiringPi	UART	wiringPi	For modules using UART
	GPIO	wiringPi	For modules using UART
Debug Release	SPI	spi_dev	For modules using SPI
	GPIO	libgpiod	For modules using SPI
Debug_WiringPi Release_WiringPi	SPI	spi_dev	For modules using SPI
	GPIO	wiringPi	For modules using SPI
Debug Release	UART	FTDI	For USB radio stick
	GPIO	None	For USB radio stick. Uses FTDI CBUS pins instead of GPIOs of the Raspberry Pi.

Table 4: Implementations of serial and GPIO interfaces in Wireless Connectivity SDK

WiringPi is deprecated by the author and no longer under development. WiringPi shall not be used anymore.

The definition of the serial and USB interfaces of the Raspberry Pi is included.

3.2 Software history

Version 1.0.0 "Engineering"

- Initial version of the SDK

Version 1.2.0 "Release"

- Added new products
- Updated driver structure to easily switch between serial and USB interface on Raspberry

Version 1.6.3/2.0.0 "Release"

- Added new products

Version 3.0.0 "Release"

- Added driver for Wi-Fi module Calypso and proprietary high power radio module Thebe-I
- Replaced old module names by new module names

Version 3.1.0 "Release"

- Added driver for proprietary high power radio module Thebe-II and Themisto-I
- Bugfix in reset function of Proteus-* drivers
- Fixed typos in function names and resulting bug in Calypso driver

Version 3.2.0 "Release"

- Added driver for proprietary 2,4 GHz module Thyone-I and Thyone-I Plug
- Added driver for Bluetooth® LE module Proteus-III and Proteus-III Plug

Version 3.3.0 "Release"

- Added driver for Bluetooth® LE module Proteus-III-SPI
- Added implementation for SPI interfaces
- Added gpio implementation using libgpiod as wiringPi replacement

Version 3.4.0 "Release"

- Updated driver for proprietary 2,4 GHz module Thyone-I to add new features of latest module firmware
- Updated driver for Bluetooth® LE module Proteus-III add new features of latest module firmware
- This is the final version of the Wireless Connectivity SDK for Raspberry Pi. The SDK repository has been archived and the SDK will not be developed any further.

3.3 Host integration

As described in chapter 3.1, the Wireless Connectivity SDK provides an implementation for the Raspberry Pi platform. In order to use all the features of the radio module with a different type of host microcontroller, the module's drivers in the SDK have to be ported to the new custom platform.

In the following example, the steps involved in porting the drivers of the Bluetooth® LE 4.2 radio module Proteus-I to a custom platform are described.

- The directories **drivers/global** and **drivers/ProteusI** as well as the file **WE_common.h** have to be integrated into the custom project.

- In the function **InitDriver** of the file **Proteusl.c**, a thread is defined that listens to confirmation and indication messages that are transmitted from the attached radio module to the host processor.
  Case 1: Threads are supported, the **rxthread** has to be ported to the host's thread system.
  Case 2: Threads are not supported, a state machine has to be created that periodically checks for incoming UART bytes.

```
void *rx_thread()
{
    while(1)
    {
        /* wait for 1ms, then check if new RX data is available */
        delay (1);
        while (BytesAvailable())
        {
            /* interpret received byte */
            if (ReadByte(&readBuffer))
            {
                ...
```

Code 3: Code snippet of the rxthread

- The file **global.h** declares the shared functions that deal with the serial interface as well as the usage of GPIOs for pin related functions.

```
/* Switch pin to input/output high/low with/without pullup/pulldown
* input:
* - pin_number: number of pin
* - inout: input or output
* - pull: pullup, pulldown or no pull
* - out: output level high or low
*
* return: true, if success
*         false, otherwise
*/
extern bool SetPin( int pin_number, SetPin_InputOutput_t inout, SetPin_Pull_t pull,
    SetPin_Out_t out);

/*
* Open the serial interface
* input:
* - baudrate: baudrate of the interface
* return: true, if success
*         false, otherwise
*/
extern bool OpenSerial(int baudrate);
```

Code 4: Code snippet of the file global.h

  Here the definition of these functions, depending on the custom host peripherals, has to be created by the user. The existing files **global_serial.c**, **global_ftdi.c** and **global.c** can be removed from the project as it contains the corresponding implementation for the Raspberry Pi.

After dealing with the **rxthread** and the definition of the functions declared in **global.h**, the driver is functional. The corresponding demo project can be considered as a basis for application development on the custom platform.

3.4 Running sample applications on the Raspberry Pi

3.4.1 Hardware connections

For creating custom applications on the basis of the Raspberry Pi, connect the pins of the module to corresponding pins on the Raspberry Pi (power supply, ground, serial/SPI interface and other pins like reset). If the SPI version of a module is used, the driver assumes that *SPI0 CE0* is used as chip select pin. Please refer to figure 7 to get an overview of the pins of the Raspberry Pi used in the driver application examples.

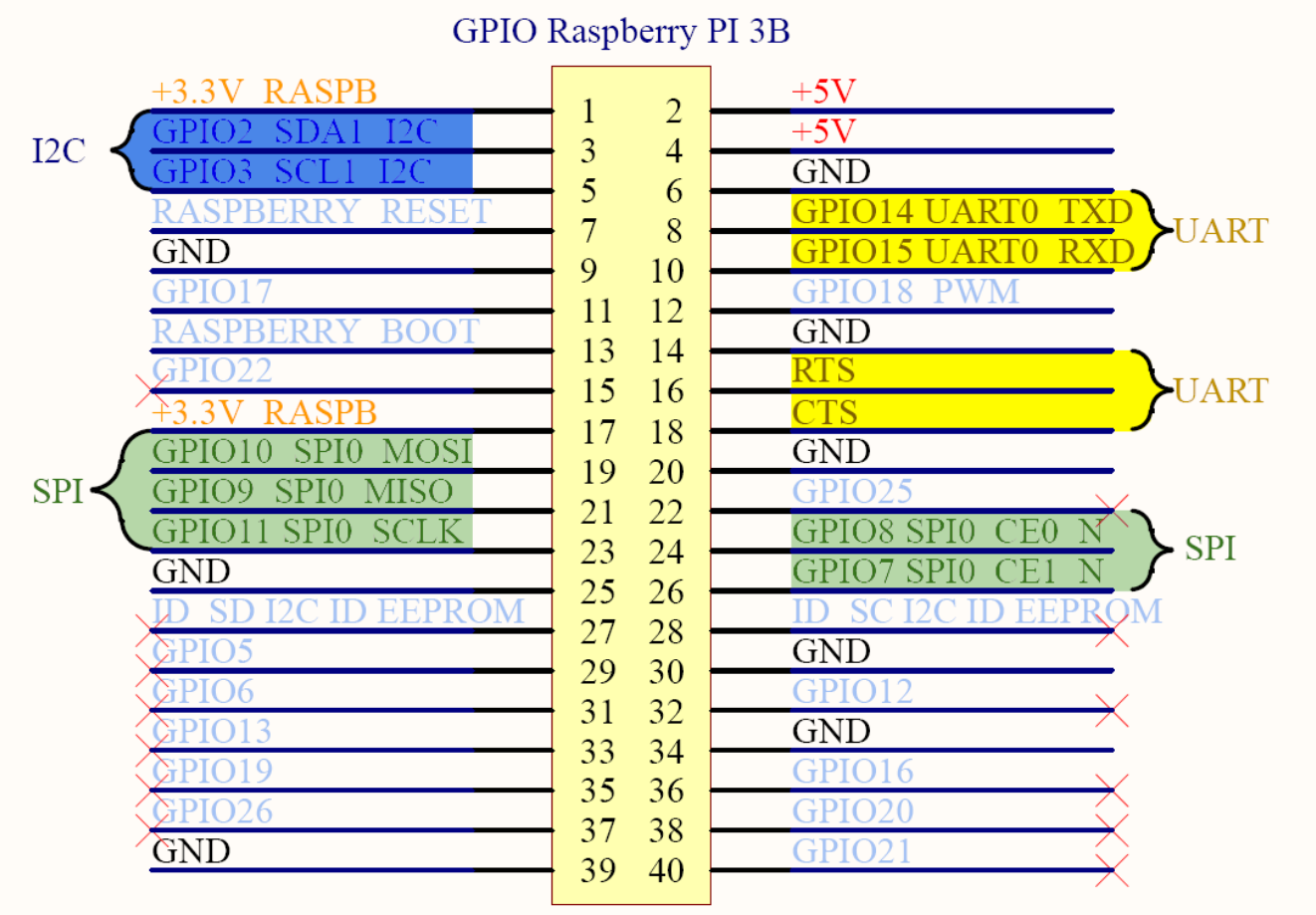

Figure 7: Extended connector from Raspberry Pi

Alternatively, Würth Elektronik eiSos USB dongles such as the Tarvos-II USB radio stick & Tarvos-III USB radio stick and evaluation boards can be directly connected to the USB-interface of the Raspberry Pi. If a USB port is to be used, the D2xx driver from FTDI has to be installed first (see section 3.4.4.2).

Some products, such as the Tarvos-II USB radio stick, operate in transparent mode by default. In that mode, the device does not interpret the commands sent by the Wireless Connectivity SDK and thus communication fails.
In order to use the module with the SDK, first set the default operation mode to command mode, for example using the Würth Elektronik eiSos software tool ACC.

3.4.2 Install the Raspberry Pi OS on the Raspberry Pi

1. Install the Raspberry Pi OS to a SD card according to official documentation [2].
   It is recommended to use the Raspberry Pi Imager.
2. After installing the image on the SD card, insert it into the Raspberry Pi's SD card slot, connect your monitor, mouse and keyboard. Now the Raspberry Pi is ready to boot up. Please start it by powering it up.
3. After booting the Raspberry Pi, switch off the Bluetooth® interface by clicking on the Bluetooth® button on the right upper corner of the screen (see figure 8). Otherwise it is not possible to use the UART interface of the Raspberry Pi.
4. Then turn on the WiFi for connecting to the internet by clicking on the WiFi button on the right upper corner of the screen and selecting the WiFi of your choice.

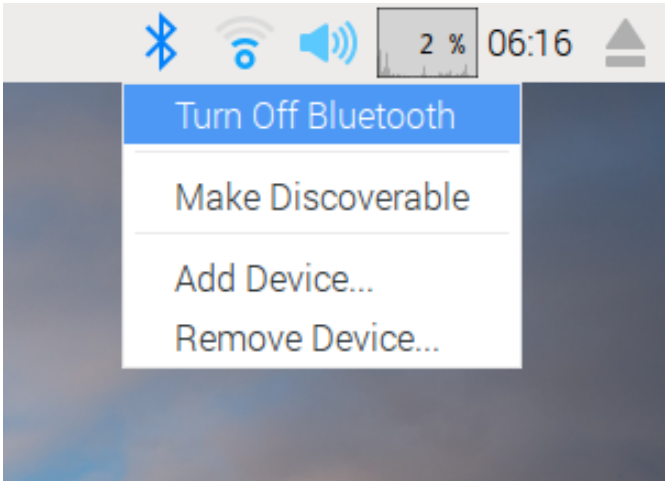

Figure 8: Switch off the Bluetooth® and connect to internet via WiFi

5. After connecting to the internet make sure your Raspberry Pi is up to date with the latest versions of Raspberry Pi OS. To update the system open a terminal by clicking on the terminal symbol in the left upper corner (see figure 9).
6. Then upgrade the Raspberry Pi OS by typing in terminal:

```
sudo apt-get update
sudo apt-get upgrade
```

Figure 9: Terminal button

3.4.3 Configuring the peripherals

1. Next, the peripherals have to be enabled. To do so open the menu by clicking on the Raspberry Pi symbol on the left upper corner of the screen and open the **Preferences** → **Raspberry Pi Configuration** window (see figure 10). Enable the SPI or serial depending on the required interface.

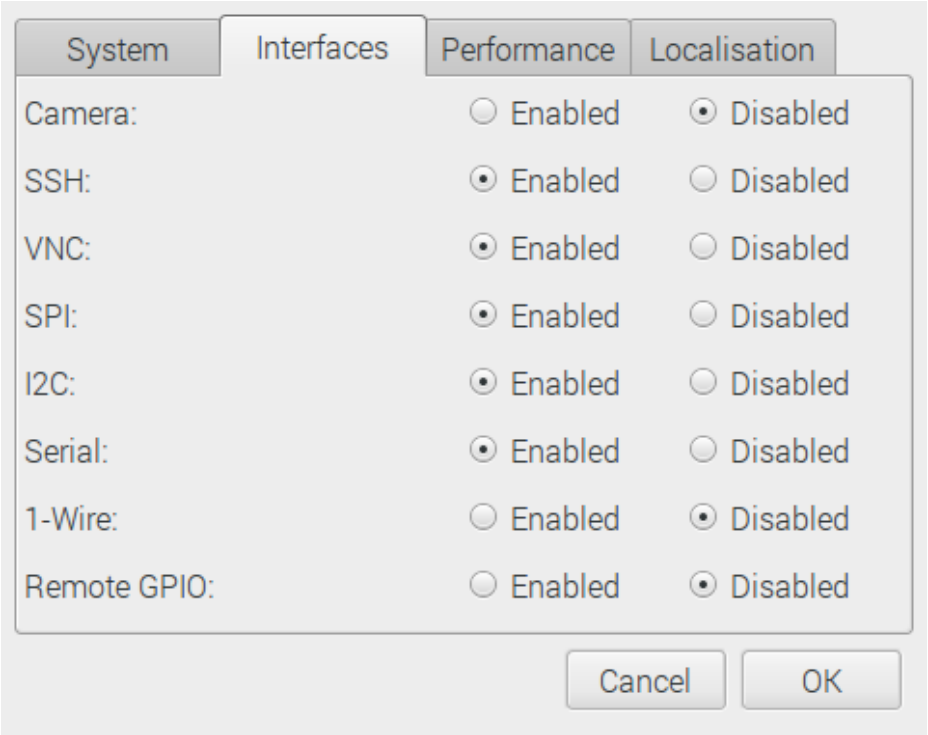

Figure 10: Raspberry Pi interface configuration

2. After enabling the interfaces a dialog should appear asking for a reboot to apply the changes. If no dialog appears reboot by clicking on the Raspberry symbol on the left upper corner of the screen and select **Shutdown**.

3. Now, after enabling the serial interface, the Raspbian OS claims it for console output. To disable this feature, please remove the string "console = serial0,115200" from the file **/boot/cmdline.txt** and save it. Root privilige is needed to change the file. To open the file accordingly type in terminal:

```
sudo leafpad /boot/cmdline.txt
```

4. Please check whether the serial interface is still enabled by opening the file **/boot/config.txt** and check whether the string "enable_uart=1" is still included. If not, please add it and save the file. Root privilige is needed to change the file. To open the file accordingly type in terminal:

```
sudo leafpad /boot/config.txt
```

5. Please reboot as before and check whether the files **/boot/cmdline.txt** and **/boot/config.txt** are still as described in the previous two points. If not, adapt the two files again as described before and reboot. Otherwise the UART-interface to the module is not active and thus the module communication fails.

6. In order to use the peripherals as a non-root user, the local user has to be a member of the peripheral group. In order to check this, type in the following in the terminal,

```
groups
```

   If the output contains GPIO and ,if required, SPI, then skip the next step.

7. Add the current user to the groups by typing in the following commands in the terminal,

```
sudo adduser pi gpio
sudo adduser pi spi
```

   Logout and login to update the user group settings.

3.4.4 Install the required libraries

By default termios is used to communicate with the UART interface of the Raspberry Pi. This is installed by default and no further action is required. The same is true for the SPI interface and the usage of the spidev API.
The library to control the GPIO pins has to be installed however.

3.4.4.1 Install the libgpiod library

To access the GPIO pins *libpgiod* is used. It can be installed by typing the following in the terminal:

```
sudo apt-get update
sudo apt-get install libgpiod2 libgpiod-dev libgpiod-doc
```

Note that as of now Raspberry Pi OS is based on Debian Buster and installs libpgiod version 1.2. This version of libgpiod does not support Pull-up and Pull-Down for GPIO pins yet. This feature was introduced with version 1.5. For the sample applications using modules with UART interface, libgpiod version 1.2 is sufficient.
For modules using the SPI interface however, Pull-up and Pull-Down is required and libgpiod has to be build and installed manually. See libgpiod documentation:
https://git.kernel.org/pub/scm/libs/libgpiod/libgpiod.git/about/

3.4.4.2 Install the FTDI driver for USB (optional)

To run a Würth Elektronik eiSos USB dongle such as the Tarvos-II USB radio stick on the Raspberry Pi, the FTDI library is used and has to be installed.

To do so, first download the latest D2xx Linux driver [1] for the ARM hard-float architecture. There are three versions available of the the FTDI D2xx driver which support the Raspberry Pi. To find out which architecture is used by your Raspberry Pi and Raspberry Pi OS, type the following in the terminal:

```
lscpu
```

The command will print information about the CPU. For output "Architecture: armv71" use the v7 version of the D2xx driver.

To install the FTDI driver the following commands have to be run in the terminal. It is assumed that the driver is version 1.4.22 for armv7 and saved to the ~/**Downloads directory**.

1. Go to the directory where the driver is saved

   ```
   cd ~/Downloads
   ```

2. Unpack the gzip file

   ```
   tar -xvf libftd2xx-arm-v7-hf-1.4.22.tgz
   ```

3. Copy the needed header files to the system folder /usr/local/include

   ```
   sudo cp release/ftd2xx.h /usr/local/include
   sudo cp release/WinTypes.h /usr/local/include
   ```

4. Copy the libraries to the system folder /usr/local/lib

   ```
   sudo cp release/build/lib* /usr/local/lib
   ```

5. Create a symbolic link to the latest version

   ```
   sudo ln -sf /usr/local/lib/libftd2xx.so.1.4.22 /usr/local/lib/libftd2xx.so
   ```

6. Update the shared libraries

   ```
   sudo ldconfig
   ```

7. As the D2xx driver is incompatible with the FTDI VCP driver in the Linux kernel, the kernel modules "ftdi_sio" and "usbserial" have to be unloaded. To do so, please run:

   ```
   sudo modprobe -r ftdi_sio
   sudo modprobe -r usbserial
   ```

8. Now all supported USB dongles can be used.

3.4.5 Install the Wireless Connectivity SDK

The Wireless Connectivity SDK was developed in the Code::Blocks development environment.

1. First download and install the software Code::Blocks. Open a terminal and type:

```
sudo apt-get install codeblocks
```

2. Now download the Wireless Connectivity SDK driver [8] as zip file to the location ~/**Downloads**

3. The file is going to be extracted to the folder ~/**Projects**. If the folder does not exist create it by typing in terminal:

```
mkdir ~/Projects
```

4. Now extract the Wireless Connectivity SDK to ~/**Projects** by typing in terminal:

```
unzip ~/Downloads/WirelessConnectivity-SDK.zip -d ~/Projects
```

5. Then start the desired project via Code::Blocks by typing in terminal. For example,

```
codeblocks ~/Projects/WirelessConnectivity-SDK/Example_ProteusIII/ProteusIII.
    cbp &
```

6. Select the build target using the drop-down menu, e.g. Debug.

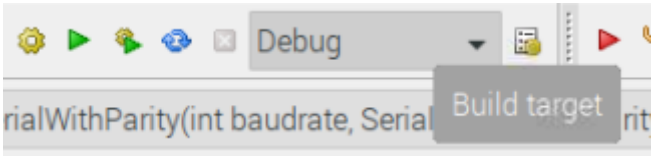

Figure 11: Select build target of the application

7. Then press **Build** → **Rebuild** to build the project (see figure 12).

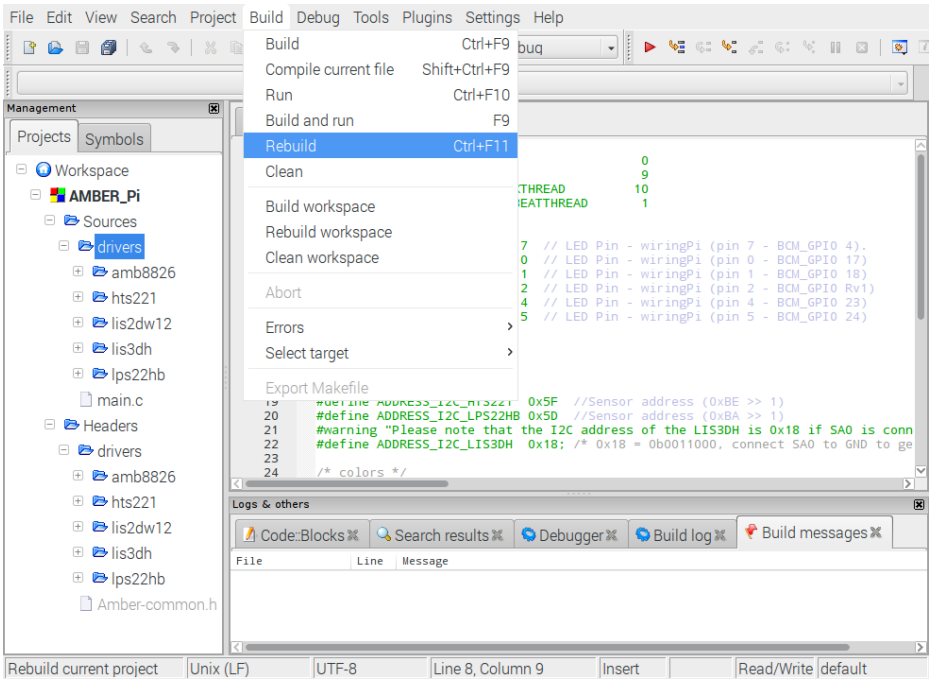

Figure 12: Rebuild the application

8. If it builds without errors the setup succeeded.
9. In case of further question, please contact our technical support [7].

3.4.6 FAQ - Frequently asked questions

3.4.6.1 The initialization function fails, what can I do?

The initialization function usually sets up the serial interface, performs a pin reset and waits for the module's response. In case this fails, there are several possibilities:

- The module is not powered up. Please check the *VCC* and *GND* connection.
- The *RESET* line is not connected, thus no pin reset was applied.
- The *UART RX* and *UART TX* lines are not connected, thus the module response was not transmitted.
- The UART interface does not run well. Please check the UART settings and initialization.
- The connected module or USB dongle does not run in command mode and thus does not respond to a pin reset and/or command request. In this case, set the device to command mode, for example by using the software tool ACC.
- The **rxthread** function, waiting for module response, does not work correctly.

3.4.6.2 ProteusIII-SPI: Pin Wake-up leads to "GetPin: Could not read pin level"

When using the ProteusIII-SPI, the same pin is used to wake-up the module and to check for new messages. So there is a chance that the pin is accessed at the same time by different functions within the application. Mostly this will have no effect and the wake-up will succeed. If it leads to errors, the problem can be fixed by using a mutex.

4 References

[1] FTDI D2XX driver. https://ftdichip.com/drivers/d2xx-drivers/.

[2] Raspbian download page. https://www.raspberrypi.org/.

[3] STMicroelectronics . UM1724 - User manual - STM32 Nucleo-64 boards (MB1136). https://www.st.com/.

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

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

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

[7] Würth Elektronik Support. https://we-online.com/wireless-connectivity/support.

[8] Würth Elektronik. Wireless Connectivity SDK for Raspberry Pi - Radio module drivers in C-code. https://github.com/WurthElektronik/WirelessConnectivity-SDK.

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

5 Important notes

The following conditions apply to all goods within the wireless connectivity product range of Würth Elektronik eiSos GmbH & Co. KG:

5.1 General customer responsibility

Some goods within the product range of Würth Elektronik eiSos GmbH & Co. KG contain statements regarding general suitability for certain application areas. These statements about suitability are based on our knowledge and experience of typical requirements concerning the areas, serve as general guidance and cannot be estimated as binding statements about the suitability for a customer application. The responsibility for the applicability and use 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, where appropriate to investigate and to decide whether the device with the specific product characteristics described in the product specification is valid and suitable for the respective customer application or not. Accordingly, the customer is cautioned to verify that the documentation is current before placing orders.

5.2 Customer responsibility related to specific, in particular safety-relevant applications

It has to be clearly pointed out that the possibility of a malfunction of electronic components or failure before the end of the usual lifetime cannot be completely eliminated in the current state of the art, even if the products are operated within the range of the specifications. The same statement is valid for all software sourcecode and firmware parts contained in or used with or for products in the wireless connectivity and sensor product range of Würth Elektronik eiSos GmbH & Co. KG. In certain customer applications requiring a high level of safety and especially in customer applications in which the malfunction or failure of an electronic component could endanger human life or health, it must be ensured by most advanced technological aid of suitable design of the customer application that no injury or damage is caused to third parties in the event of malfunction or failure of an electronic component.

5.3 Best care and attention

Any product-specific data sheets, manuals, application notes, PCN's, warnings and cautions must be strictly observed in the most recent versions and matching to the products firmware revisions. This documents can be downloaded from the product specific sections on the wireless connectivity homepage.

5.4 Customer support for product specifications

Some products within the product range may contain substances, which are subject to restrictions in certain jurisdictions in order to serve specific technical requirements. Necessary information is available on request. In this case, the field sales engineer or the internal sales person in charge should be contacted who will be happy to support in this matter.

5.5 Product improvements

Due to constant product improvement, product specifications may change from time to time. As a standard reporting procedure of the Product Change Notification (PCN) according to the JEDEC-Standard, we inform about major changes. In case of further queries regarding the PCN, the field sales engineer, the internal sales person or the technical support team in charge should be contacted. The basic responsibility of the customer as per section 5.1 and 5.2 remains unaffected. All wireless connectivity module driver software ¨wireless connectivity SDK¨ and it's source codes as well as all PC software tools are not subject to the Product Change Notification information process.

5.6 Product life cycle

Due to technical progress and economical evaluation we also reserve the right to discontinue production and delivery of products. As a standard reporting procedure of the Product Termination Notification (PTN) according to the JEDEC-Standard we will inform at an early stage about inevitable product discontinuance. According to this, we cannot ensure that all products within our product range will always be available. Therefore, it needs to be verified with the field sales engineer or the internal sales person in charge about the current product availability expectancy before or when the product for application design-in disposal is considered. The approach named above does not apply in the case of individual agreements deviating from the foregoing for customer-specific products.

5.7 Property rights

All the rights for contractual products produced by Würth Elektronik eiSos GmbH & Co. KG on the basis of ideas, development contracts as well as models or templates that are subject to copyright, patent or commercial protection supplied to the customer will remain with Würth Elektronik eiSos GmbH & Co. KG. Würth Elektronik eiSos GmbH & Co. KG does not warrant or represent that any license, either expressed or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, application, or process in which Würth Elektronik eiSos GmbH & Co. KG components or services are used.

5.8 General terms and conditions

Unless otherwise agreed in individual contracts, all orders are subject to the current version of the "General Terms and Conditions of Würth Elektronik eiSos Group", last version available at *www.we-online.com*.

6 Legal notice

6.1 Exclusion of liability

Würth Elektronik eiSos GmbH & Co. KG considers the information in this document to be correct at the time of publication. However, Würth Elektronik eiSos GmbH & Co. KG reserves the right to modify the information such as technical specifications or functions of its products or discontinue the production of these products or the support of one of these products without any written announcement or notification to customers. The customer must make sure that the information used corresponds to the latest published information. Würth Elektronik eiSos GmbH & Co. KG does not assume any liability for the use of its products. Würth Elektronik eiSos GmbH & Co. KG does not grant licenses for its patent rights or for any other of its intellectual property rights or third-party rights.

Notwithstanding anything above, Würth Elektronik eiSos GmbH & Co. KG makes no representations and/or warranties of any kind for the provided information related to their accuracy, correctness, completeness, usage of the products and/or usability for customer applications. Information published by Würth Elektronik eiSos GmbH & Co. KG regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof.

6.2 Suitability in customer applications

The customer bears the responsibility for compliance of systems or units, in which Würth Elektronik eiSos GmbH & Co. KG products are integrated, with applicable legal regulations. Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of Würth Elektronik eiSos GmbH & Co. KG components in its applications, notwithstanding any applications-related in-formation or support that may be provided by Würth Elektronik eiSos GmbH & Co. KG. Customer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences lessen the likelihood of failures that might cause harm and take appropriate remedial actions. The customer will fully indemnify Würth Elektronik eiSos GmbH & Co. KG and its representatives against any damages arising out of the use of any Würth Elektronik eiSos GmbH & Co. KG components in safety-critical applications.

6.3 Trademarks

AMBER wireless is a registered trademark of Würth Elektronik eiSos GmbH & Co. KG. All other trademarks, registered trademarks, and product names are the exclusive property of the respective owners.

6.4 Usage restriction

Würth Elektronik eiSos GmbH & Co. KG products have been designed and developed for usage in general electronic equipment only. This product is not authorized for use in equipment

where a higher safety standard and reliability standard is especially required or where a failure of the product is reasonably expected to cause severe personal injury or death, unless the parties have executed an agreement specifically governing such use. Moreover, Würth Elektronik eiSos GmbH & Co. KG products are neither designed nor intended for use in areas such as military, aerospace, aviation, nuclear control, submarine, transportation (automotive control, train control, ship control), transportation signal, disaster prevention, medical, public information network etc. Würth Elektronik eiSos GmbH & Co. KG 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 electronic component, which is used in electrical circuits that require high safety and reliability function or performance. By using Würth Elektronik eiSos GmbH & Co. KG products, the customer agrees to these terms and conditions.

7 License terms

This License Terms will take effect upon the purchase and usage of the Würth Elektronik eiSos GmbH & Co. KG wireless connectivity products. You hereby agree that this license terms is applicable to the product and the incorporated software, firmware and source codes (collectively, "Software") made available by Würth Elektronik eiSos in any form, including but not limited to binary, executable or source code form.
The software included in any Würth Elektronik eiSos wireless connectivity product is purchased to you on the condition that you accept the terms and conditions of this license terms. You agree to comply with all provisions under this license terms.

7.1 Limited license

Würth Elektronik eiSos hereby grants you a limited, non-exclusive, non-transferable and royalty-free license to use the software and under the conditions that will be set forth in this license terms. You are free to use the provided Software only in connection with one of the products from Würth Elektronik eiSos to the extent described in this license terms. You are entitled to change or alter the source code for the sole purpose of creating an application embedding the Würth Elektronik eiSos wireless connectivity product. The transfer of the source code to third parties is allowed to the sole extent that the source code is used by such third parties in connection with our product or another hardware provided by Würth Elektronik eiSos under strict adherence of this license terms. Würth Elektronik eiSos will not assume any liability for the usage of the incorporated software and the source code. You are not entitled to transfer the source code in any form to third parties without prior written consent of Würth Elektronik eiSos.
You are not allowed to reproduce, translate, reverse engineer, decompile, disassemble or create derivative works of the incorporated Software and the source code in whole or in part. No more extensive rights to use and exploit the products are granted to you.

7.2 Usage and obligations

The responsibility for the applicability and use of the Würth Elektronik eiSos wireless connectivity product with the incorporated Firmware in a particular customer design is always solely within the authority of the customer. Due to this fact, it is up to you to evaluate and investigate, where appropriate, and to decide whether the device with the specific product characteristics described in the product specification is valid and suitable for your respective application or not.
You are responsible for using the Würth Elektronik eiSos wireless connectivity product with the incorporated Firmware in compliance with all applicable product liability and product safety laws. You acknowledge to minimize the risk of loss and harm to individuals and bear the risk for failure leading to personal injury or death due to your usage of the product.
Würth Elektronik eiSos' products with the incorporated Firmware are not authorized for use in safety-critical applications, or where a failure of the product is reasonably expected to cause severe personal injury or death. Moreover, Würth Elektronik eiSos' products with the incorporated Firmware are neither designed nor intended for use in areas such as military, aerospace, aviation, nuclear control, submarine, transportation (automotive control, train control, ship control), transportation signal, disaster prevention, medical, public information network etc. You

shall inform Würth Elektronik eiSos about the intent of such usage before design-in stage. In certain customer applications requiring a very high level of safety and in which the malfunction or failure of an electronic component could endanger human life or health, you must ensure to have all necessary expertise in the safety and regulatory ramifications of your applications. You acknowledge and agree that you are solely responsible for all legal, regulatory and safety-related requirements concerning your products and any use of Würth Elektronik eiSos' products with the incorporated Firmware in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by Würth Elektronik eiSos. YOU SHALL INDEMNIFY WÜRTH ELEKTRONIK EISOS AGAINST ANY DAMAGES ARISING OUT OF THE USE OF WÜRTH ELEKTRONIK EISOS' PRODUCTS WITH THE INCORPORATED FIRMWARE IN SUCH SAFETY-CRITICAL APPLICATIONS.

7.3 Ownership

The incorporated Firmware created by Würth Elektronik eiSos is and will remain the exclusive property of Würth Elektronik eiSos.

7.4 Firmware update(s)

You have the opportunity to request the current and actual Firmware for a bought wireless connectivity Product within the time of warranty. However, Würth Elektronik eiSos has no obligation to update a modules firmware in their production facilities, but can offer this as a service on request. The upload of firmware updates falls within your responsibility, e.g. via ACC or another software for firmware updates. Firmware updates will not be communicated automatically. It is within your responsibility to check the current version of a firmware in the latest version of the product manual on our website. The revision table in the product manual provides all necessary information about firmware updates. There is no right to be provided with binary files, so called "Firmware images", those could be flashed through JTAG, SWD, Spi-Bi-Wire, SPI or similar interfaces.

7.5 Disclaimer of warranty

THE FIRMWARE IS PROVIDED "AS IS". YOU ACKNOWLEDGE THAT WÜRTH ELEKTRONIK 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 YOUR INTENDED PURPOSE OR USAGE. WÜRTH ELEKTRONIK 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 THE WÜRTH ELEKTRONIK EISOS' PRODUCT WITH THE INCORPORATED FIRMWARE IS USED. INFORMATION PUBLISHED BY WÜRTH ELEKTRONIK EISOS REGARDING THIRD-PARTY PRODUCTS OR SERVICES DOES NOT CONSTITUTE A LICENSE FROM WÜRTH ELEKTRONIK EISOS TO USE SUCH PRODUCTS OR SERVICES OR A WARRANTY OR ENDORSEMENT THEREOF.

7.6 Limitation of liability

Any liability not expressly provided by Würth Elektronik eiSos shall be disclaimed.
You agree to hold us harmless from any third-party claims related to your usage of the Würth Elektronik eiSos' products with the incorporated Firmware, software and source code. Würth Elektronik eiSos disclaims any liability for any alteration, development created by you or your customers as well as for any combination with other products.

7.7 Applicable law and jurisdiction

Applicable law to this license terms shall be the laws of the Federal Republic of Germany. Any dispute, claim or controversy arising out of or relating to this license terms shall be resolved and finally settled by the court competent for the location of Würth Elektronik eiSos' registered office.

7.8 Severability clause

If a provision of this license terms is or becomes invalid, unenforceable or null and void, this shall not affect the remaining provisions of the terms. The parties shall replace any such provisions with new valid provisions that most closely approximate the purpose of the terms.

7.9 Miscellaneous

Würth Elektronik eiSos reserves the right at any time to change this terms at its own discretion. It is your responsibility to check at Würth Elektronik eiSos homepage for any updates. Your continued usage of the products will be deemed as the acceptance of the change.
We recommend you to be updated about the status of new firmware and software, which is available on our website or in our data sheet and manual, and to implement new software in your device where appropriate.
By ordering a wireless connectivity product, you accept this license terms in all terms.

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