



ANR024 PROTEUS-E

ADVANCED DEVELOPER GUIDE

VERSION 1.0

MARCH 9, 2022

Revision history

Manual version	FW version	HW version	Notes	Date
1.0	1.0.0	1.0	• Initial version	February 2022

Abbreviations

Abbreviation	Name	Description
ADC	Analog to Digital Converter	
BTMAC		Bluetooth® conform MAC address of the module used on the RF-interface.
CPU	Central Processing Unit	
GAP	Generic Access Profile	The GAP provides a basic level of functionality that all Bluetooth® devices must implement.
GATT	Generic Attribut profile	
GPIO	General Purpose Input/Output	
I/O	Input/output	Pinout description.
I ² C	Inter-Integrated Circuit	
LDO	Low Dropout	Low dropout voltage regulator
Bluetooth LE	Bluetooth Low Energy	
LPM	Low power mode	Mode for efficient power consumption.
MAC		MAC address of the module.
MTU	Maximum transmission unit	Maximum packet size of the Bluetooth® connection.
Payload		The intended message in a frame / package.
PDU	Packet Data Unit	
RF	Radio frequency	Describes wireless transmission.
SDK	Software Development Kit	
Bluetooth SIG	Bluetooth Special Interest Group	
SoC	Sistem-on-Chip	
Soft device		Operating system used by the nRF52 chip.
SPI	Serial peripheral interface	Allows the serial communication with the module.
SPP	Serial Port Profile	
UART	Universal Asynchronous Receiver Transmitter	Allows the serial communication with the module.
UUID	Universally Unique Identifier	

Contents

1	Introduction	5
2	Bluetooth profiles	5
3	SPP-like profile	6
3.1	Generic Access Protocol (GAP)	6
3.2	Generic Attribute Profile (GATT)	7
3.2.1	Maximum transmission unit (MTU)	7
3.2.2	Company identifier	7
3.2.3	UUID	7
3.2.4	Primary Service	7
3.2.4.1	Characteristics	7
3.3	Bluetooth LE packet content	8
3.3.1	Payload packet format	8
3.3.2	Advertising packet content	9
3.3.3	Scan response packet content	9
3.4	Remote GPIO control	10
3.4.1	CMD_GPIO_REMOTE_WRITECONFIG_REQ	10
3.4.1.1	Example: Configure two GPIOs of the connected remote device to output high	11
3.4.2	CMD_GPIO_REMOTE_READCONFIG_REQ	13
3.4.2.1	Example: Read the current GPIO configuration of the connected remote device	13
3.4.3	CMD_GPIO_REMOTE_WRITE_REQ	15
3.4.3.1	Example: Set a remote output GPIO to low	16
3.4.4	CMD_GPIO_REMOTE_READ_REQ	17
3.4.4.1	Example: Read the values of remote GPIOs	18
3.4.5	CMD_GPIO_LOCAL_WRITE_IND	18
3.4.5.1	Example: GPIOs of the remote device have been written by its local host	18
4	App development	19
4.1	Connection setup message chart	19
4.2	Enable notifications	20
4.3	Bonding development hints	21
5	Custom firmware development	22
5.1	Important information for custom firmware development	22
5.1.1	How to adapt Nordic Semiconductor SDK examples to run on the Proteus-e hardware?	26
5.1.2	Firmware development hints	28
5.1.3	Qualifying the Proteus-e with respect to Bluetooth® 5.2	28
6	References	29
7	Important notes	30
7.1	General customer responsibility	30
7.2	Customer responsibility related to specific, in particular safety-relevant applications	30
7.3	Best care and attention	30

7.4	Customer support for product specifications	30
7.5	Product improvements	31
7.6	Product life cycle	31
7.7	Property rights	31
7.8	General terms and conditions	31
8	Legal notice	32
8.1	Exclusion of liability	32
8.2	Suitability in customer applications	32
8.3	Trademarks	32
8.4	Usage restriction	32
9	License terms	34
9.1	Limited license	34
9.2	Usage and obligations	34
9.3	Ownership	35
9.4	Firmware update(s)	35
9.5	Disclaimer of warranty	35
9.6	Limitation of liability	35
9.7	Applicable law and jurisdiction	36
9.8	Severability clause	36
9.9	Miscellaneous	36

1 Introduction

This document provides advanced information concerning the Proteus-e.

On the one hand the Bluetooth® LE interface of the Proteus-e is described, that must be implemented in a smart phone app to gain compatibility with the Proteus-e (chapter 2, 3 and 4).

On the other hand, valuable hints to start a custom firmware development on base of the Proteus-e hardware are given (chapter 5).

Please note that basic understanding of the Bluetooth® LE standard as well as application development background on the desired platform is necessary to fully understand this document. Please refer to the user manual of Proteus-e [4] to get basic information of the functions provided by the application firmware.

2 Bluetooth profiles

The Bluetooth® LE specification uses so called "profiles" to specify the general behaviour of a Bluetooth® LE enabled device on the radio to communicate with other Bluetooth® LE devices. Profiles are built on the Bluetooth® LE standard to clearly define what kind of data is transmitted. The device's application determines which profiles matches best, from hands-free capabilities to heart rate sensors to alerts and more.

A device may support more than one profile. For two devices to be compatible, they must support the same Bluetooth® LE profile.

The Proteus-e module ships with the so-called **SPP-like profile** (see section 3) created on the base of the Generic Attribute profile (GATT). This profile is a custom profile defined by Würth Elektronik eiSos and aims at providing a Bluetooth® LE based wireless replacement to a serial cable connection.

3 SPP-like profile

This section contains the key data of the SPP-like profile. Each device in the network must support this profile to communicate with a Proteus-e device. Customer applications may support and/or provide other profiles, services or interfaces in addition.

3.1 Generic Access Protocol (GAP)

The main purpose of this protocol is to describe the parameters of lower layers of the Bluetooth® LE stack including discovery, scanning and security capabilities. The Proteus-e GAP specifications are listed below:

- See user setting `RF_Appearance` for the appearance.
- See user setting `RF_DeviceName` for the device name.
- See user setting `FS_BTMAC` (0x0018DAxxxxxx) for the device address (6 Byte MAC) of type "public".
- Timings:
 - See user settings `RF_AdvertisingInterval` for advertising interval configuration.
 - See user setting `RF_ConnectionInterval` for minimum and maximum connection interval configuration.
 - See user setting `RF_TXPower` for TX power value.
 - See user setting `RF_SecFlags` for security settings.
 - Slave latency: 0
 - Peripheral requests for connection parameters update if central has differing connection parameters
 - * Connection parameters update (initial): 5s
 - * Connection parameters update (periodic): 10s
 - * Connection parameters update counter before connection shut down: 3

3.2 Generic Attribute Profile (GATT)

3.2.1 Maximum transmission unit (MTU)

The Proteus-e supports up to 243 Byte of payload data. To use this feature a large MTU has to be requested by the central device. In this case, the GATT MTU size must be 243 Byte payload + 1 Byte header + 3 Byte NUS header, which is 247 Byte in total.

The PDU size should be 243 Byte payload + 1 Byte header + 3 Byte NUS header + 4 Byte Bluetooth® LE header, which is 251 Byte in total.



Check also the message charts in chapter 4 to see the MTU request in the connection setup process.

3.2.2 Company identifier

The Bluetooth® listed company identifier of Würth Elektronik eiSos (formerly "Amber wireless GmbH") is 0x031A (794_{dec}).

3.2.3 UUID

The Proteus-e uses a 128 Bit UUID of type "Vendor specific". The base UUID is adapted by the 16 Bit UUIDs of the primary service and the corresponding characteristics.

The use of these UUID is restricted to Proteus-e modules that come with a pre-installed firmware, or devices that communicate with them.

Service	16 Bit UUID	Full UUID
Proteus-e base	-	6E400000-C352-11E5-953D-0002A5D5C51B
Proteus-e primary service	0x0001	6E400001-C352-11E5-953D-0002A5D5C51B
RX_CHARACTERISTIC	0x0002	6E400002-C352-11E5-953D-0002A5D5C51B
TX_CHARACTERISTIC	0x0003	6E400003-C352-11E5-953D-0002A5D5C51B



The UUID can be adapted by means of the user settings RF_SPPBaseUUID, RF_SPPServiceUUID, RF_SPPTXUUID and RF_SPPRXUUID to generate a custom profile.

3.2.4 Primary Service

3.2.4.1 Characteristics

- The first characteristic of the Proteus-e primary service is RX_CHARACTERISTIC:
 - The data is sent from central/client to peripheral/server using a write command.
 - Server:
 - * Has to allow a write command as well as a write without response command.

- Client:
 - * Use write command to send data to the server.
- The second characteristic of the Proteus-e primary service is TX_CHARACTERISTIC:
 - The data is sent from peripheral/server to central/client using a notification.
 - Server:
 - * Has to allow/enable notifications. Notify client/central when sending data.
 - * When the notification enable bit is written in the CCCD (Client Characteristic Configuration Descriptor) by the central, the peripheral prints a CMD_CHANNELOPEN_RSP on the UART to signalize that the peripheral can send data to the central now. The central can only write this bit, when the configured security level of the peripheral has been met.
 - Client:
 - * Has to enable notifications.

The permissions to access the characteristics is determined by the security mode of the module.

Proteus-e security mode	CCCD read	CCCD write, RX attribute read/write, TX attribute read/write
No security	no protection, open link	no protection, open link
Just works	no protection, open link	requires encryption, but no MITM protection (Mode 1, Level 2)
Static pass key	no protection, open link	requires encryption and MITM protection (Mode 1, Level 3)

3.3 Bluetooth LE packet content

3.3.1 Payload packet format

To identify the type of data transmitted via Bluetooth® LE, the data protocol on the radio contains a 1 Byte packet header. Thus, the standard Bluetooth® LE payload has to match the following format to be accepted by the Proteus-e:

Bluetooth® LE Payload	
Header	Payload
1 Byte	Φ_{ST} Bytes

Table 1: Packet format

The maximum payload size Φ_{ST} is 243 Bytes. The Header has to be one of the following types:

0x01: RF_HEADER_TYPE_DATA: The following bytes contain the user payload data.

0x02: RF_HEADER_TYPE_CMD: The following bytes contain command data, like commands to switch the remote GPIOs (see chapter 3.4).

Other: Other headers are reserved for future use and packets with this header are currently discarded.

3.3.2 Advertising packet content

The standard Proteus-e advertising packet contains the following data:

- Advertising data flags
- Proteus-e device name as Full or Shortened Local Name (up to 26 Bytes)

3.3.3 Scan response packet content

The scan response packet is requested during scan if active scanning is enabled on the central device (i.e. smart phone). The standard Proteus-e scan response packet contains the following data:

- The UUID (128 Bit Proteus-e primary service UUID) of the SPP-like profile
- TX power level (1 Byte in two's complement notation)

3.4 Remote GPIO control

The Proteus-e contains a feature to control its free GPIOs via the Bluetooth® LE interface. To do so a connected remote device must send remote commands via the Bluetooth® LE interface to the Proteus-e.

In case the GPIOs of interest have not been configured by the locally connected host via UART commands, it must be configured first by the remote device via Bluetooth® LE using the CMD_GPIO_REMOTE_WRITECONFIG_REQ and CMD_GPIO_REMOTE_READCONFIG_REQ commands.

If this has been done, the GPIOs can be used as input and/or output pins (CMD_GPIO_REMOTE_WRITE_REQ / CMD_GPIO_REMOTE_READ_REQ).

The remote commands to be send have the following structure (see the following chapters):

Bluetooth® LE Payload	
Header	Command Payload
0x02	x Bytes

Table 2: Packet command format

3.4.1 CMD_GPIO_REMOTE_WRITECONFIG_REQ

This command can be used to configure the free GPIOs of the remote device.



Remote configuration can be blocked by writing the corresponding bit of the user setting CFG_Flags via UART commands.

Format:

Header	Command	Block ₁	...	Block _n
0x02	0x28	x Bytes		x Bytes

Response (CMD_GPIO_REMOTE_WRITECONFIG_CNF):

Header	Command 0x40	Block ₁	...	Block _n
0x02	0x68	x Bytes		x Bytes

CMD_GPIO_REMOTE_WRITECONFIG_REQ block structure

Each **Block** has the following format:

Length	GPIO_ID	Function	Value
0x03	1 Byte	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see Proteus-e manual

Function:

0x00: GPIO disconnected

0x01: GPIO works as input

0x02: GPIO works as output

Value:

- if **Function** is disconnected:
Values field is not used in this Block
- if **Function** is input:
0x00: GPIO has no pull resistor
0x01: GPIO has pull down resistor
0x02: GPIO has pull up resistor
- if **Function** is output:
0x00: GPIO is output low
0x01: GPIO is output high

CMD_GPIO_REMOTE_WRITECONFIG_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Status
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see Proteus-e manual

Status:

0x00: Success

0x01: Failed

0xFF: Remote configuration not allowed (blocked by the user setting CFG_Flags of the remote device)

3.4.1.1 Example: Configure two GPIOs of the connected remote device to output high

Configure the GPIOs with ID **0x01** and **0x02** to output high:

Header	Command	Block ₁	Block ₂
0x02	0x28	0x03 0x01 0x02 0x01	0x03 0x02 0x02 0x01

Response:

Header	Command 0x40	Block ₁	Block ₂
0x02	0x68	0x02 0x01 0x00	0x02 0x02 0x00

Configured both GPIOs with success.

3.4.2 CMD_GPIO_REMOTE_READCONFIG_REQ

This command can be used to read the configuration of the free GPIOs of the remote device.
Format:

Header	Command
0x02	0x2C

Response (CMD_GPIO_REMOTE_READCONFIG_CNF):

Header	Command 0x40	Block ₁	...	Block _n
0x02	0x6C	x Bytes		x Bytes

CMD_GPIO_REMOTE_READCONFIG_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Function	Value
0x03	1 Byte	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see Proteus-e manual

Function:

0x00: GPIO is not configured yet (Length is 0x02 and **Value** is empty)

0x01: GPIO works as input

0x02: GPIO works as output

Value:

- if **Function** is input:

0x00: GPIO has no pull resistor

0x01: GPIO has pull down resistor

0x02: GPIO has pull up resistor

- if **Function** is output:

0x00: GPIO is output low

0x01: GPIO is output high

3.4.2.1 Example: Read the current GPIO configuration of the connected remote device

Read the current GPIO configuration of the connected remote device:

Header	Command
0x02	0x2C

Response:

Header	Command 0x40	Blocks
0x02	0x6C	0x03 0x01 0x02 0x01 0x03 0x02 0x02 0x01 0x02 0x03 0x00 0x02 0x04 0x00 0x02 0x05 0x00 0x02 0x06 0x00

The GPIOs with GPIO_ID **0x01** and **0x02** are output high. The remaining GPIOs with GPIO_ID **0x03**, **0x04**, **0x05** and **0x06** are not configured.

3.4.3 CMD_GPIO_REMOTE_WRITE_REQ

This command can be used to write the free GPIOs of the remote device. This command can be only run successfully if the respective pins of the remote device are configured as output pins.



Perform a CMD_GPIO_REMOTE_READCONFIG_REQ before using the CMD_GPIO_REMOTE_WRITE_REQ command to ensure the pins are setup correctly.

Format:

Header	Command	Block ₁	...	Block _n
0x02	0x29	x Bytes		x Bytes

Response (CMD_GPIO_REMOTE_WRITE_CNF):

Header	Command 0x40	Block ₁	...	Block _n
0x02	0x69	x Bytes		x Bytes

CMD_GPIO_REMOTE_WRITE_REQ block structure

Each **Block** has the following format:

Length	GPIO_ID	Value
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see Proteus-e manual

Value:

0x00: Set GPIO to low

0x01: Set GPIO to high

CMD_GPIO_REMOTE_WRITE_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Status
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see Proteus-e manual

Status:**0x00:** Success**0x01:** Failed**3.4.3.1 Example: Set a remote output GPIO to low**Set the output GPIO (GPIO_ID **0x01**) of the connected remote device to low:

Header	Command	Block ₁
0x02	0x29	0x02 0x01 0x00

Response:

Header	Command 0x40	Block ₁
0x02	0x69	0x02 0x01 0x00

Successfully set GPIO with GPIO_ID **0x01** to low.

3.4.4 CMD_GPIO_REMOTE_READ_REQ

This command can be used to read the free GPIOs of the remote device. This command can be only run successfully if the respective pins of the remote device are configured.



Perform a CMD_GPIO_REMOTE_READCONFIG_REQ before using the CMD_GPIO_REMOTE_READ_REQ command to ensure the pins are setup correctly.

Format:

Header	Command	Block ₁	...	Block _n
0x02	0x2A	x Bytes		x Bytes

Response (CMD_GPIO_REMOTE_READ_CNF)

Header	Command 0x40	Block ₁	...	Block _n
0x02	0x6A	x Bytes		x Bytes

CMD_GPIO_REMOTE_READ_REQ block structure

Each **Block** has the following format:

Length	GPIO_ID ₁	...	GPIO_ID _n
1 Bytes	1 Byte		1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see Proteus-e manual

CMD_GPIO_REMOTE_READ_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Value
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see Proteus-e manual

Value:

0x00: The remote GPIO is low.

0x01: The remote GPIO is high.

0xFF: Failed reading remote GPIO value.

3.4.4.1 Example: Read the values of remote GPIOs

Read the value of the GPIOs with GPIO_ID **0x01** and **0x02** of the connected remote device:

Header	Command	Block ₁
0x02	0x2A	0x02 0x01 0x02

Response:

Header	Command 0x40	Block ₁	Block ₂
0x02	0x6A	0x02 0x01 0x00	0x02 0x02 0x01

Successfully read the values of the remote GPIOs with GPIO_ID **0x01** (GPIO is low) and **0x02** (GPIO is high).

3.4.5 CMD_GPIO_LOCAL_WRITE_IND

This message informs the connected remote device, that the radio module's local host has written the GPIOs.



Please note that only the GPIOs that have been successfully updated are part of this message. Failed attempts of GPIO updates will not be indicated by this message.

Format:

Header	Command	Block ₁	...	Block _n
0x02	0xA8	x Bytes		x Bytes

Each **Block** has the format of CMD_GPIO_REMOTE_READ_CNF block structure.

3.4.5.1 Example: GPIOs of the remote device have been written by its local host

Header	Command	Block ₁	Block ₂
0x02	0xA6	0x02 0x01 0x00	0x02 0x02 0x01

The GPIOs with GPIO_ID **0x01** (GPIO is low) and **0x02** (GPIO is high) of the remote device have been written by its local host.

4 App development

The definition of the SPP-like profile (see section 3) in combination with the message chart of chapter 4.1 is sufficient to develop custom apps for mobile devices. To implement this profile from scratch fundamental knowledge of app development as well as of the Bluetooth® LE standard is required.

To make life easy Würth Elektronik eiSos published the source code of the compatible "Proteus Connect App" for iOS [6] and Android [5] on GitHub. This app can be used as starting point for own app developments.

4.1 Connection setup message chart

The following message chart shows which steps are run during the connection setup process between a mobile phone and a Proteus-e radio module. To implement the central role in a mobile device app to connect to the Proteus-e peripheral the steps of the central device shown below have to be reproduced.

1. First of all, the central device must place a connection request to setup the physical connection. Here the timing parameters like the connection interval are negotiated.
2. In case the Proteus-e has its security enabled (see user setting `RF_SecFlags`), a pairing request must be placed to get the permission to access the peripheral's characteristics.
3. Afterwards a MTU request is necessary to allow a larger payload.
4. Next the discovery of the characteristics must be done and the notification of the TX characteristic has to be enabled.

After all these steps have been done, data transmission in both directions is possible.

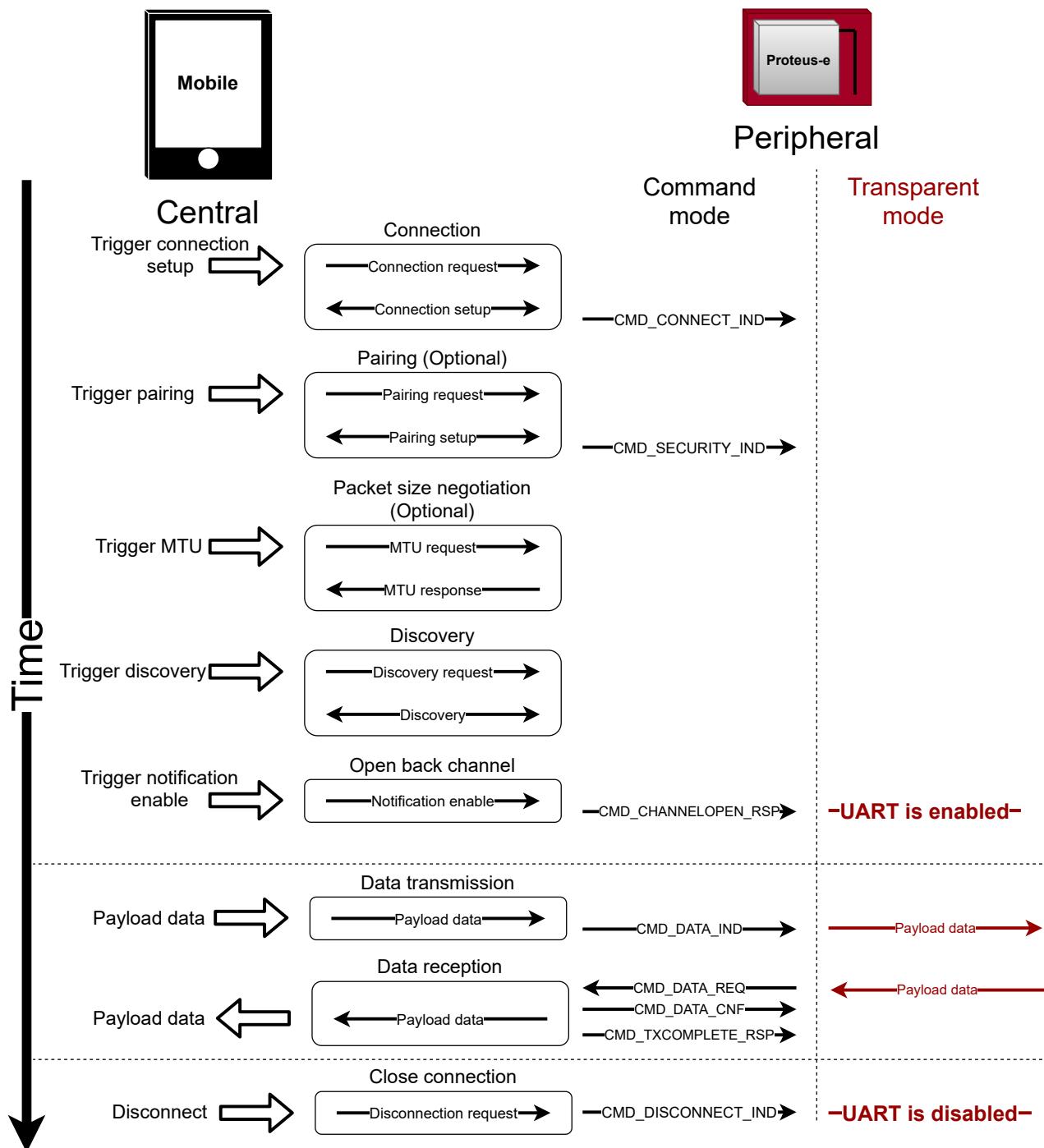


Figure 1: Connection setup chart

4.2 Enable notifications

As described in the previous chapter 4.1 the final step for a successful connection setup is the enabling of the notification of the TX_CHARACTERISTIC. To do so, the Android's Bluetooth® LE stack offers the following function, that has to be called with the TX_CHARACTERISTIC.

```
IBluetoothGatt mService;
/** TX NOTIFICATION
 * Enable or disable notifications / indications for a given characteristic .
*/
```

```

* <p>Once notifications are enabled for a characteristic , a
* {@link BluetoothGattCallback#onCharacteristicChanged} callback will be
* triggered if the remote device indicates that the given characteristic
* has changed.
*
* <p>Requires {@link android.Manifest.permission#BLUETOOTH} permission.
*
* @param characteristic The characteristic for which to enable notifications
* @param enable Set to true to enable notifications/indications
* @return true, if the requested notification status was set successfully
*/
public boolean setCharacteristicNotification(BluetoothGattCharacteristic characteristic ,
boolean enable) {
if (DBG) {
Log.d(TAG, "setCharacteristicNotification () - uuid:" + characteristic.getUuid()
+ " enable:" + enable);
}
if (mService == null || mClientIf == 0) return false;

BluetoothGattService service = characteristic.getService();
if (service == null) return false;

BluetoothDevice device = service.getDevice();
if (device == null) return false;

try {
mService.registerForNotification (mClientIf, device.getAddress(),
characteristic.getInstancId(), enable);
} catch (RemoteException e) {
Log.e(TAG, "", e);
return false;
}

return true;
}

```

Code 1: Example code to enable the TX characteristic notification

Please note that the iOS's Bluetooth® LE stack calls the corresponding function automatically. Thus calling a notification enable function from the app's application layer is not needed.

4.3 Bonding development hints

The firmware of the Proteus-e provides the bonding feature that allows to re-pair without repeating the authentication step (e.g. entering the static passkey). Thus, in the initial connection all bonding data is stored in the devices' flash to be used during the setup of subsequent connections.

The function `CMD_DELETEBONDS_REQ` of the Proteus-e allows to remove not needed bonding data from the module's flash. Thus in case of missing bonding data on one of the two connection partners, a re-bonding has to be initiated by the central device! Otherwise, the security level is not met to send the "notification enable" message and thus the channel for data transmission can not be opened.



Please note that iOS devices do not run the re-bonding step by default, if bonding data is missing on one of the two connection partners.

In certain cases, the bonding data on the iOS device has to be cleared first, such that iOS starts the re-bonding step.

5 Custom firmware development

Using the Proteus-e hardware a custom firmware can be developed to better fit the customer's needs. Based on the Nordic Semiconductor SDK (nRF5 SDK [2]) and demo examples various Bluetooth® LE profiles and custom applications can be realized and flashed on the Proteus-e module. The versatile and well documented Nordic stack ensures quick and easy realization of various standard Bluetooth® LE profiles. Chapter 5.1 contains the information needed to run Nordic standard examples on the Proteus-e hardware.

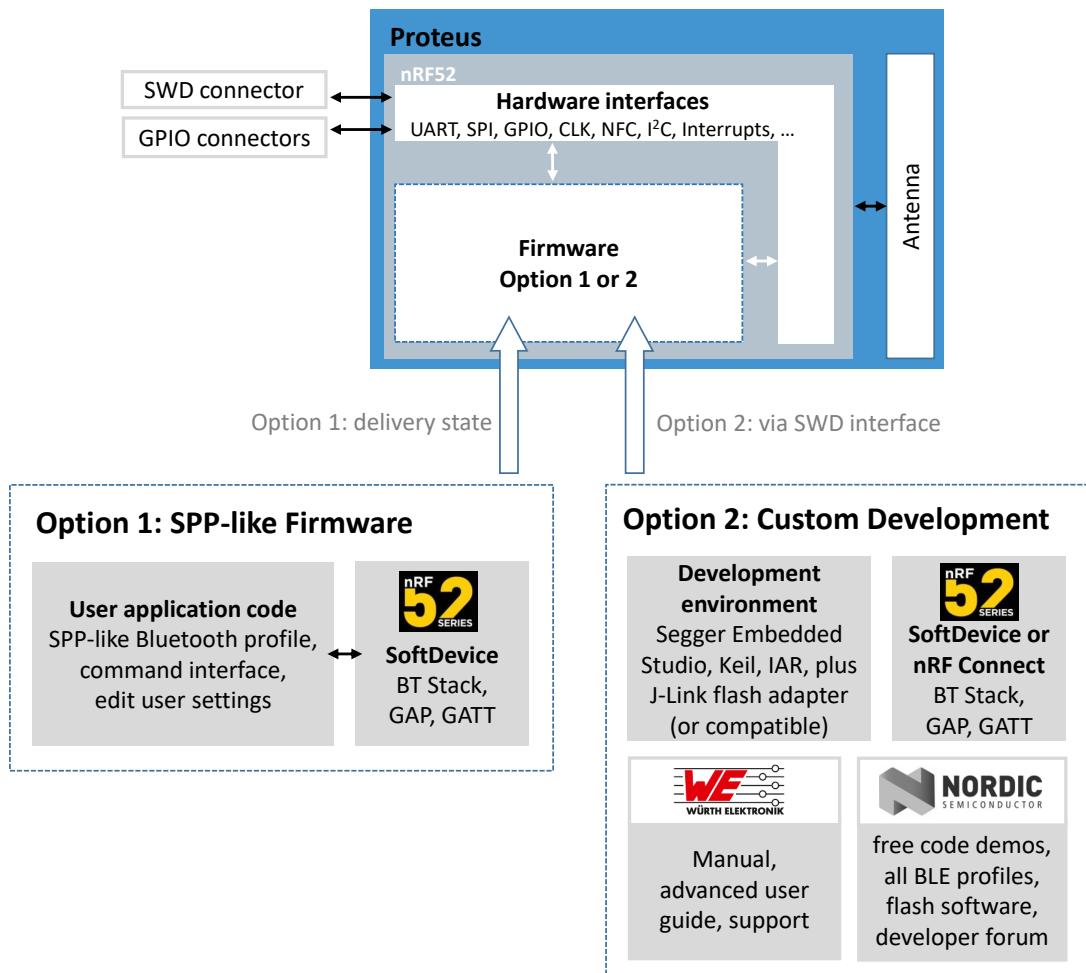


Figure 2: Options for running the Proteus-e with standard or custom firmware

5.1 Important information for custom firmware development

To start a custom firmware development on top of the Proteus-e hardware, the following information must be considered:

- **Chip**

The Proteus-e contains the Nordic Semiconductor nRF52805 SoC. The CPU is a 64MHz ARM Cortex-M4.

- **Pinout**

The Proteus-e provides the following pins of the Nordic SoC with its pads. Only the *ANT*, *RF*, *GND*, *VDD*, *Reset*, *SWDCLK* and *SWDIO* pins are fixed. All other pins

can be used for custom firmware development. For special functions like external low frequency crystal (XL) or analog input (AIN) the respective pins have to be used.

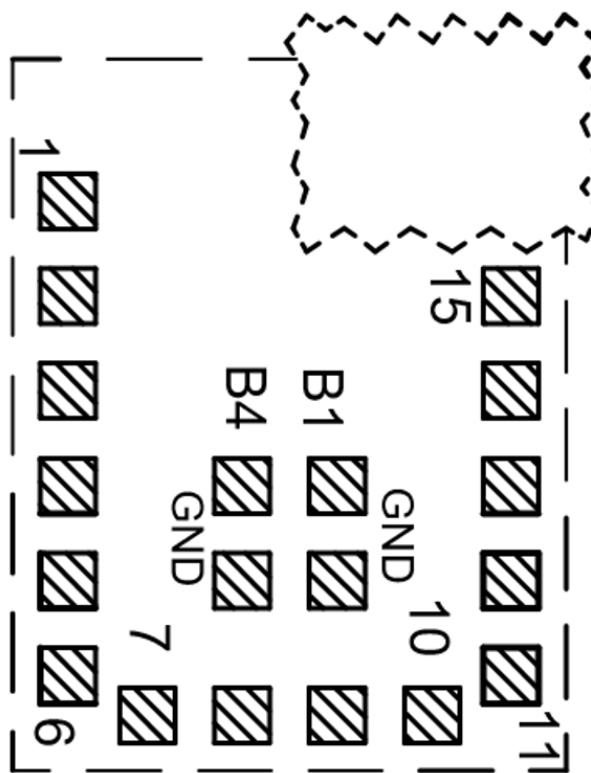


Figure 3: Pinout

No.	Pad Name	No.	Pad Name
1	ANT	11	P0.00/XL1
2	RF	12	P0.01/XL2
3	SWDIO	13	P0.16
4	SWDCLK	14	P0.18
5	P0.21/Reset	15	P0.20
6	P0.12	B1	GND
7	VDD	B2	GND
8	P0.05/AIN3	B3	GND
9	P0.04/AIN2	B4	GND
10	P0.14		

- Hardware for development & debugging
Using Segger J-Link flasher and the SWD interface is required for firmware development and debugging. Checkout the Proteus-e evaluation board. It provides the easiest way to develop firmware based on Proteus-e module or apps for the SPP-like profile.
- Software development environment
Nordic Semiconductor provides software packages for several compilers (KEIL, IAR, GCC, Segger Embedded).

It includes the required Bluetooth® LE stack ("Softdevice"), many demo examples for Bluetooth® LE profiles and services to conveniently develop a custom firmware on base of the Nordic SoC. Further library's for hardware peripheral (such as ADC, I2C, SPI, UART etc.) are also include in the SDK and examples. More information and details about the chip and the operating system is bundled in the Nordic Semiconductor Infocenter:

<http://infocenter.nordicsemi.com/>

Please check the tab "nRF52 Series" to access the newest information about the nRF52 radio chip and the software environment.

If available, use the examples for the Nordic evaluation platform (like PCA10040) as a starting point. See also chapter 5.1.1 for more information how to run Nordic standard examples on top of the Proteus-e.

- Clock sources

The Proteus-e module contains a dedicated RF clock (HFCLK). The Proteus-e does not contain a dedicated low frequency clock (LFCLK). Thus custom firmware must use the internal RC-oscillator as long as no external clock crystal is connected to the respective pins (*XL1*, *XL2*) on the customer PCB.

Example for enabling the internal RC oscillator for nRF Connect SDK 17.0.2:

```
// <o> NRF_SDH_CLOCK_LF_SRC – SoftDevice clock source.
// <0=> NRF_CLOCK_LF_SRC_RC
// <1=> NRF_CLOCK_LF_SRC_XTAL
// <2=> NRF_CLOCK_LF_SRC_SYNTH
#ifndef NRF_SDH_CLOCK_LF_SRC
#define NRF_SDH_CLOCK_LF_SRC 0
#endif

// <o> NRF_SDH_CLOCK_LF_RC_CTIV – SoftDevice calibration timer interval.
#ifndef NRF_SDH_CLOCK_LF_RC_CTIV
#define NRF_SDH_CLOCK_LF_RC_CTIV 16
#endif

// <o> NRF_SDH_CLOCK_LF_RC_TEMP_CTIV – SoftDevice calibration timer interval under
// constant temperature.
// <i> How often (in number of calibration intervals) the RC oscillator shall be calibrated
// <i> if the temperature has not changed.
#ifndef NRF_SDH_CLOCK_LF_RC_TEMP_CTIV
#define NRF_SDH_CLOCK_LF_RC_TEMP_CTIV 2
#endif

// <o> NRF_SDH_CLOCK_LF_ACCURACY – External clock accuracy used in the LL to compute
// timing.
// <0=> NRF_CLOCK_LF_ACCURACY_250_PPM
// <1=> NRF_CLOCK_LF_ACCURACY_500_PPM
// <2=> NRF_CLOCK_LF_ACCURACY_150_PPM
// <3=> NRF_CLOCK_LF_ACCURACY_100_PPM
// <4=> NRF_CLOCK_LF_ACCURACY_75_PPM
// <5=> NRF_CLOCK_LF_ACCURACY_50_PPM
// <6=> NRF_CLOCK_LF_ACCURACY_30_PPM
// <7=> NRF_CLOCK_LF_ACCURACY_20_PPM
// <8=> NRF_CLOCK_LF_ACCURACY_10_PPM
// <9=> NRF_CLOCK_LF_ACCURACY_5_PPM
// <10=> NRF_CLOCK_LF_ACCURACY_2_PPM
// <11=> NRF_CLOCK_LF_ACCURACY_1_PPM
#ifndef NRF_SDH_CLOCK_LF_ACCURACY
#define NRF_SDH_CLOCK_LF_ACCURACY 1
```

#endif

Code 2: sdk_config.h

- Voltage regulator

As internal voltage regulator, we recommend to use the DCDC instead of the LDO. The DCDC has to be switched on explicitly in application code. Example for SDK 17.0.2:

```
sd_power_dc_dc_mode_set(NRF_POWER_DCDC_ENABLE);
```

Changing from LDO to DCDC reduces the current consumption of the module to meet lowest power specifications.

- Certification and Bluetooth®-Listing

Custom firmware may require additional certification. Any (end-)device containing Bluetooth® IP must be listed by the Bluetooth® SIG which requires membership and qualification. Please contact the Bluetooth® SIG or your preferred Bluetooth® certification laboratory to obtain the Bluetooth®-listing



To make use of the existing certification and listing of the Proteus-e, it is mandatory to use the Bluetooth® stack Nordic Semiconductor S112 in version 7.3.0.

- Serial number

The unique serial number (used for tracing and the generation of the Proteus-e BT-MAC) is placed in the user information configuration register (UICR->Customer[0]) and will be removed by flashing a customer firmware onto the SoC.

5.1.1 How to adapt Nordic Semiconductor SDK examples to run on the Proteus-e hardware?



The following description is based on the SDK 17.0.2. Code may differ when using a different Softdevice and/or SDK version.

Please perform the following steps to run a Nordic standard example on the Proteus-e:

1. Open the desired example project for the nRF52805 radio chip and compile.
2. In case of success¹, enable the DCDC by adding the following line at the end of the stack init function.

```
static void ble_stack_init(void){
    .
    .
    .
    // Enable DCDC
    err_code = sd_power_dc_dc_mode_set(NRF_POWER_DCDC_ENABLE);
    APP_ERROR_CHECK(err_code);
}
```

3. If no external crystal has been connected to the radio module, enable the internal RC-oscillator as shown in code example 2.
4. Go to the file board.h and add the include for the Proteus-e.h board file.

```
#if defined(BOARD_PCA10040)
#include "pca10040.h"
#elif defined(BOARD_PROTEUSE)
#include "Proteuse.h"
#else
#error "Board_is_not_defined"
#endif
```

¹If you have a Nordic evaluation board available, please check that the original example without modifications runs successfully on the Nordic evaluation board.

5. Then create the Proteus-e board file. To do so, please copy the board file of the Nordic evaluation board (like PCA10040) and add the pinout, led button numbering, button numbering and clock definition of the Proteus-e:

```
#ifndef PROTEUS_E_H
#define PROTEUS_E_H

#define NRF_PIN_LED_1      0
#define NRF_PIN_BUSY_UARTENABLE 1
#define NRF_PIN_UARTRTS 4
#define NRF_PIN_GPIO2 5
#define NRF_PIN_OPERATIONMODE 12
#define NRF_PIN_UARTCTS 14
#define NRF_PIN_UARTTX 16
#define NRF_PIN_UARTRX 18
#define NRF_PIN_GPIO1 20
#define NRF_RESET 21

#define LEDS_ACTIVE_STATE 1
#define LEDS_NUMBER 1
#define LEDS_LIST { NRF_PIN_LED_1 }
#define BSP_LED_0 NRF_PIN_LED_1

#define BUTTONS_ACTIVE_STATE 0
#define BUTTON_PULL NRF_GPIO_PIN_PULLUP
#define BUTTONS_NUMBER 1
#define BUTTONS_LIST { NRF_PIN_BUSY_UARTENABLE }
#define BSP_BUTTON_0 NRF_PIN_BUSY_UARTENABLE

#define RX_PIN_NUMBER NRF_PIN_UARTRX
#define TX_PIN_NUMBER NRF_PIN_UARTTX
#define CTS_PIN_NUMBER NRF_PIN_UARTCTS
#define RTS_PIN_NUMBER NRF_PIN_UARTRTS

#endif // PROTEUS_E_H
```

Code 3: Content of the Proteuse.h

6. In the project options, we need to link to the Proteus-e hardware instead to the Nordic evaluation board hardware. This can be done by adding "BOARD_PROTEUSE" macro and by removing the respective macro of the Nordic platform in the precompiler options of the project.
7. Then check that the application code uses the pins names defined in the Proteus-e.h . Probably peripheral pins (UART, SPI,...), LED pins and/or button pins have to be adapted to fit the pin definition of the Proteus-e.h .



Please make sure that the selected pin number and its function matches the underlying hardware (e.g. evaluation board).

8. Now all necessary changes have been done. Thus, recompile the whole project and check for warnings and errors.
9. In case of success, erase the whole chip and flash ONLY the Softdevice onto the chip. The J-Flash tool can be used to do so.

10. After this, flash the compiled project code onto the chip using Segger Embedded (or the IDE of your choice) without erasing the flash area of the Softdevice.
11. Now, the whole code has been flashed and testing can start.

5.1.2 Firmware development hints

When creating a custom firmware the following hints may be useful during development:

- In standard Nordic examples, the *Reset* pin is hard coded. We recommend using the pin definition of the board-file to guarantee that changes in the layout take effect.
- Reviewing the pin settings (direction, pull-up/-down resistors) of the firmware is the first option when experiencing leakage current.
- The *UART RX* pin is quite sensitive towards wrong levels during UART start-up. A floating *UART RX* pin of the SoC may result in unwanted behaviour. In this case, an internal or external pull-up resistor can be installed to prevent floating. Be aware that this resistor will lead to leakage current.
- Checkout the errata sheet of the nRF52 SoC to have an overview of known issues with the nRF52 SoC and possible software workarounds.
- Checkout the sections "Known issues" of the used SDK and soft device versions to be aware of potential issues.

5.1.3 Qualifying the Proteus-e with respect to Bluetooth® 5.2

The Proteus-e has been listed as end product with respect to Bluetooth® 5.1 specification. In case a listing with respect to a newer version of the Bluetooth® standard is desired another Bluetooth® LE stack must be used.

The end product listing (EPL) mainly has to be built up by adding two parts:

1. The listing of the controller subsystem, that contains the test of the module and/or radio chip hardware to be Bluetooth® compliant. The hardware of the Proteus-e fulfills the specifications to be listed as a Bluetooth® 5.2 controller subsystem.
2. The listing of the host subsystem, that contains the test of the Bluetooth® stack used within the product. The Bluetooth® stack used in the Proteus-e is the S112 V7.3.0, that does not fulfill the specifications to be listed as a Bluetooth® 5.2 host subsystem. However, another Bluetooth® 5.2 listed stack, like "Zephyr BLE Host" (nRF Connect SDK [1]) developed by Nordic Semiconductor, can be used to create a new Bluetooth® 5.2 listed end product listing.

Please refer to Application Note ANR27 [3] for more information on the Bluetooth® SIG listing process and options.

6 References

- [1] Nordic Semiconductor. nRF Connect SDK. <https://www.nordicsemi.com/Products/Development-software/nRF-Connect-SDK>.
- [2] Nordic Semiconductor. nRF5 SDK. <https://www.nordicsemi.com/Products/Development-software/nrf5-sdk>.
- [3] Würth Elektronik. Application note 27 - Bluetooth listing guide. <http://www.we-online.com/ANR027>.
- [4] Würth Elektronik. Proteus-e user manual. <https://www.we-online.de/katalog/de/manual/2612011024000>.
- [5] Würth Elektronik. Source code of Proteus Connect app for Android. <https://github.com/WurthElektronik/Proteus-Connect-Android>.
- [6] Würth Elektronik. Source code of Proteus Connect app for iOS. <https://github.com/WurthElektronik/Proteus-Connect-iOS>.

7 Important notes

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

7.1 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.

7.2 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.

7.3 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.

7.4 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.

7.5 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 7.1 and 7.2 remains unaffected. All wireless connectivity module driver software "wireless connectivity SDK" and its source codes as well as all PC software tools are not subject to the Product Change Notification information process.

7.6 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.

7.7 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.

7.8 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.

8 Legal notice

8.1 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.

8.2 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 information 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.

8.3 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.

8.4 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.

9 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.

9.1 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.

9.2 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.**

9.3 Ownership

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

9.4 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.

9.5 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.

9.6 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.

9.7 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.

9.8 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.

9.9 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

1	Connection setup chart	20
2	Options for running the Proteus-e with standard or custom firmware	22
3	Pinout	23

List of Tables

1	Packet format	8
2	Packet command format	10



more than you expect



**Internet
of Things**



**Monitoring
& Control**



**Automated Meter
Reading**

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

