Document number:	*860245.010.7*	**Sheet:** 1		**Rev.:**	*V2.1*
Document title:	***Documentation, user manual, TKU***			**Date:**	*24.10.22*
Name:	*Bownce Ball C4, Anchor C5-Sample*			**modified:**	*16.01.25*
Product number:	*860245.014 (Ball C4)* *860245.015 (Anchor C5)*				
Company:	*Bownce*				
Customer product Number/Name:	*n.a.*				
Author/Editor:	*ThG*				
Verification:					

Bownce Ball/Anchor

Firmware Documentation

Documentation, user manual, TKU V2.1

Document number: *860245.011.7 V2.1 16.01.2025*

Version history:

Date	Version	Modifications
23.02.2022	1.0	new
24.10.2022	2.0	Chapter EOL Test added
16.01.2025	2.1	2.2 changed binary naming conventions

Relating Documents:

Document	Version	File
Bownce BLE Uart protocol specification	1.5 as of 07.10.22	505014_001_7_B3_V15_bownce_ble_protocol_221007pub.pdf

Table of Contents:

0 Preface

This document/documentation (TKU: short for „technische Kunden Unterlagen“ (German)) is a combination of manual, installation guide and technical specification. The application field, functions and the handling of the Bownce C1 Firmware (Ball and Anchor) are described.

This manual is only intended for trained personnel use. The execution of the documented installations, manuals etc. may cause harm to people and damage this control device and connected devices or facilities if used by untrained people.

1 Abbreviations

DFU	Device Firmware Update
BLE	Bluetooth Low Energy
NUS	Nordic UART Service
nvData	non volatile data area in Flash memory

2 Variants and Firmware Versions

2.1 Version numbering

Firmware Version Numbers consist of two parts:

- Firmware Revision String

The Firmware Revision String contains the Firmware Version Number in following format:

[major mumber].[minor number]

While the major number specifies the compatibility with the revision of the hardware

Major number	Compatibility with hardware
0	FW is compatible with functional sample
1	FW is compatible with A-Sample
2	FW is compatible with C1-Sample

The minor number is incremented with the “functionality” of the firmware. So, it is likely to have two firmware version with the same functionality but for different revisions of the hardware:

e.g:

1.16 and 2.16

Both versions has the same level of functionality (they are equal!) but 1.xx is for the A-Sample only and 2.xx is for the C1-Sample only!

Do not mix up Firmware and Hardware revisions!

- Build number

Build number is mainly used internally for communication between development and test. Build number and Firmware Revision string has 1:1 relationship but it is possible that build number has “gaps” between last and pre-last baseline version.

Firmware Revision String and build number can be requested via Bluetooth “Device Information Service”:

BLE service field		
Software Revision String	➔	Maps to “Build number”
Firmware Revision String		

2.2 Binary naming conventions:

The binary’s (Hex and Zip Files) follows the following naming conventions:

bownce_ball_1160_37_211213

The firmware revision string is coded into 4 digit number:

One digit for the major number and additional 2 digit for the minor number. The last digit is not used right now

Date of release

Build Number

There are two different files for programming the device:

- *.hex : only to be used for initial programming via the JTAG Interface. The hex file contains bootloader, softdevice, application and the master boot sector
- *.zip: This is the package for the DFU Service. It only contains the application. The bootloader can not be reprogrammed via DFU

2.3 Signatures

For secure DFU Firmware update the DFU package (*.zip file) needs to be signed. Unsigned packages or packages with wrong signs will be rejected by the bootloader.

The Keyring for the Anchor is

anchor.pem

with the public key

```
/** @brief Public key used to verify DFU images */
__ALIGN(4) const uint8_t pk[64] =
{
    0x17, 0xd5, 0xa9, 0xbe, 0x94, 0xe4, 0x36, 0xb3, 0x31, 0xde, 0x5d, 0x88, 0xab, 0x11, 0xbc, 0xad,
0xa6, 0x36, 0xa4, 0xf8, 0x61, 0xb5, 0x4e, 0x67, 0x63, 0x3e, 0xe9, 0x36, 0x28, 0x6c, 0x9a, 0xe7,
    0x03, 0xce, 0xcf, 0x4d, 0xbc, 0x14, 0x63, 0xe7, 0xaf, 0x7f, 0xe4, 0xaa, 0x8f, 0x72, 0x00, 0xd1,
0x15, 0xca, 0xfa, 0x5e, 0xc1, 0x69, 0x7e, 0x22, 0x86, 0x0d, 0x31, 0xca, 0x19, 0xc2, 0xf4, 0x6f
};
```

The keyring for the ball is

dfu_private_key.pem

with the public key:

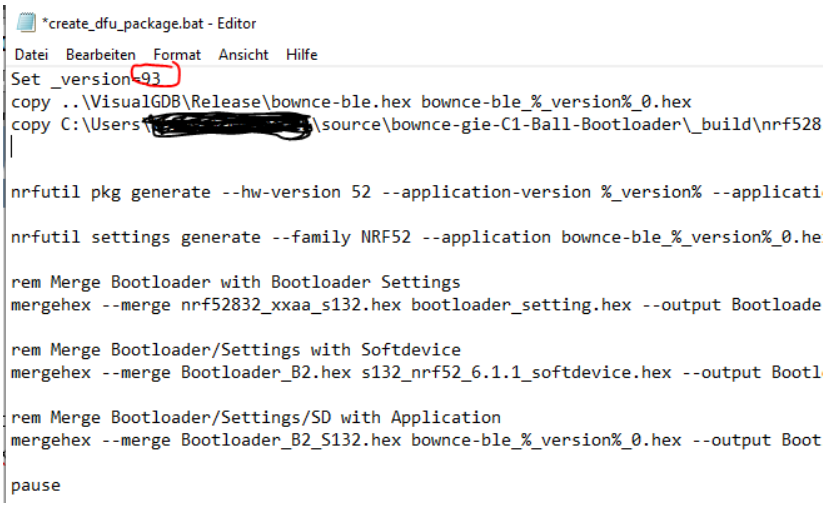
```
/** @brief Public key used to verify DFU images */
__ALIGN(4) const uint8_t pk[64] =
{
    0x90, 0x13, 0x3c, 0x7f, 0x0f, 0x29, 0xc2, 0xa2, 0xa9, 0x1f, 0x4c, 0xbf, 0x88, 0x09, 0x32, 0x64, 0x9b, 0xb8, 0xda, 0x6a, 0xfa, 0xbc, 0x78, 0x65, 0x71, 0x96, 0xdb, 0x59, 0xbb, 0x0f, 0x98, 0xdf,
    0x64, 0x83, 0x23, 0x88, 0x3e, 0xd4, 0x1a, 0xcd, 0xf5, 0xe1, 0xa5, 0x77, 0xfa, 0xd0, 0x42, 0x7f, 0x4c, 0xb8, 0x4e, 0xd0, 0xdf, 0xec, 0x45, 0x87, 0xf4, 0x74, 0xb7, 0xa3, 0x53, 0x04, 0xd7, 0x1d
};
```

The public keys are linked into the bootloader, while the private keys are used to sign the DFU package during the build process.

2.4 Building the DFU package

Building the hex files and dfu packages are done by automated script (batch file):

create_dfu_package.bat



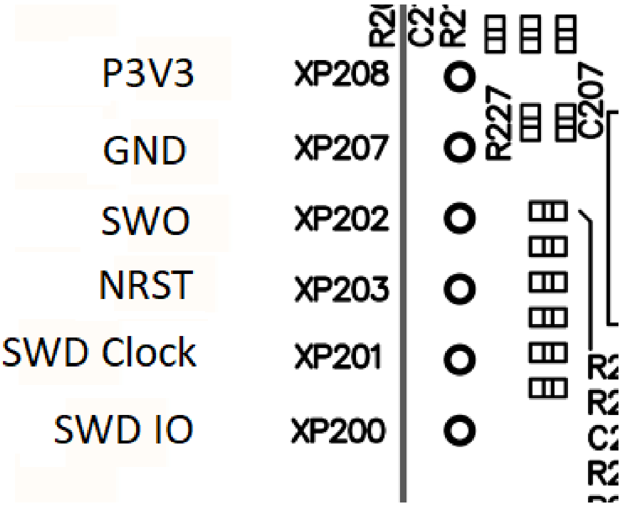
```
Set _version=93
copy ..\VisualGDB\Release\bownce-ble.hex bownce-ble_%_version%_0.hex
copy C:\Users\[redacted]\source\bownce-gie-C1-Ball-Bootloader\_build\nrf528


nrfutil pkg generate --hw-version 52 --application-version %_version% --applicati

nrfutil settings generate --family NRF52 --application bownce-ble_%_version%_0.he

rem Merge Bootloader with Bootloader Settings
mergehex --merge nrf52832_xxaa_s132.hex bootloader_setting.hex --output Bootloade

rem Merge Bootloader/Settings with Softdevice
mergehex --merge Bootloader_B2.hex s132_nrf52_6.1.1_softdevice.hex --output Bootl

rem Merge Bootloader/Settings/SD with Application
mergehex --merge Bootloader_B2_S132.hex bownce-ble_%_version%_0.hex --output Boot

pause
```

Only the _*version* constant has to be set manually before running the batch script. The _*version* conmstant muss correspond with the build number in the source code (See checklist in main.c)

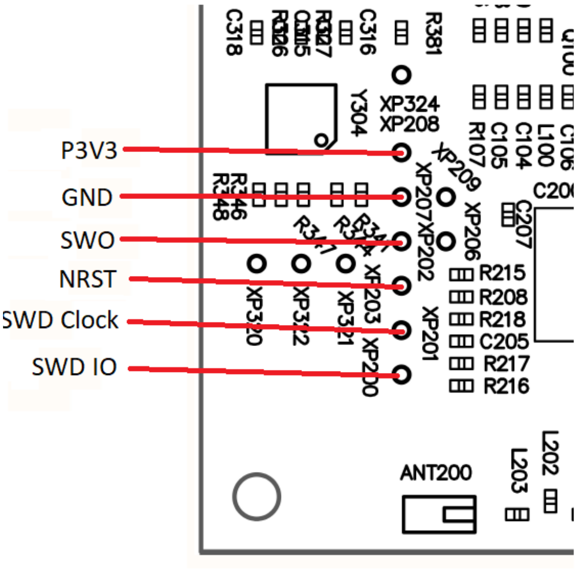
3 Flashing the decice via JTAG

3.1 Connecting the device

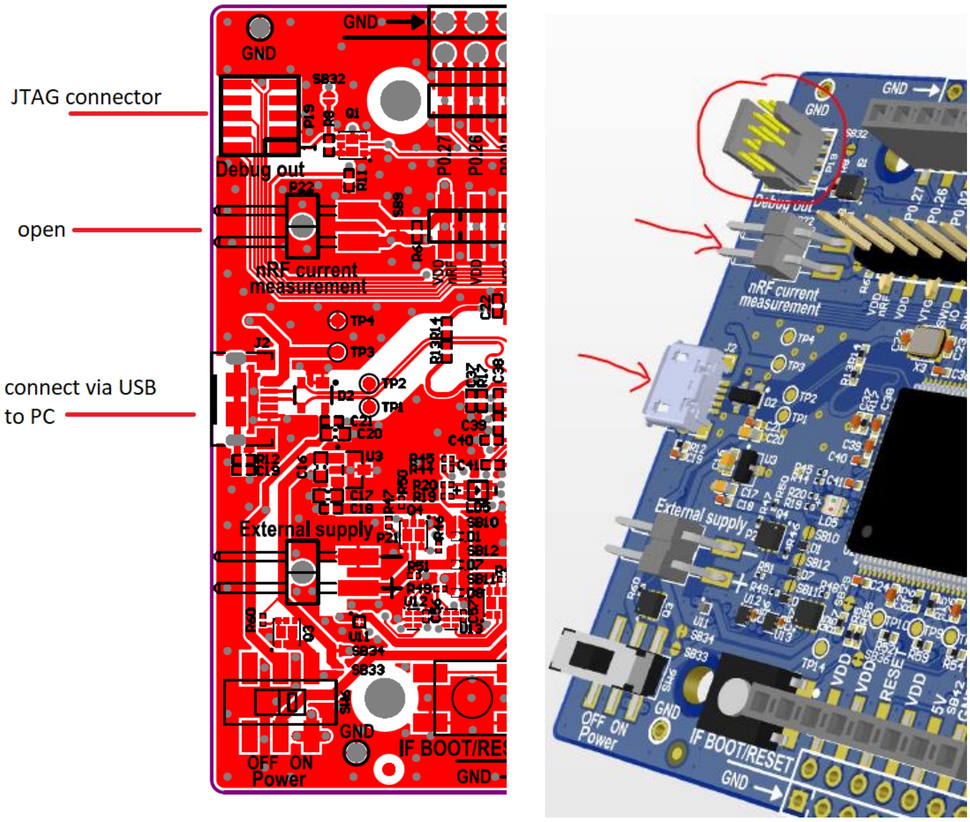
Ball: (A-Sample and C1-Sample equal)

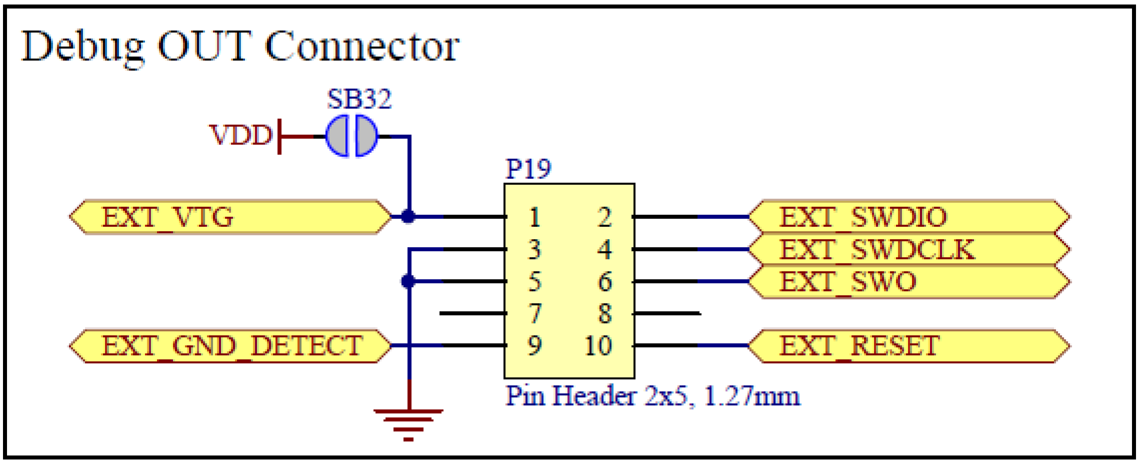


Anchor: (A-Sample and C1-Sample equal)



For programming the NORDIC PCA10040 DevBoard can be used.





3.2 Programming the device

The driver for the Segger JTAG-Interface needs to be installed on the PC:

[SEGGER - The Embedded Experts - Downloads - J-Link / J-Trace]

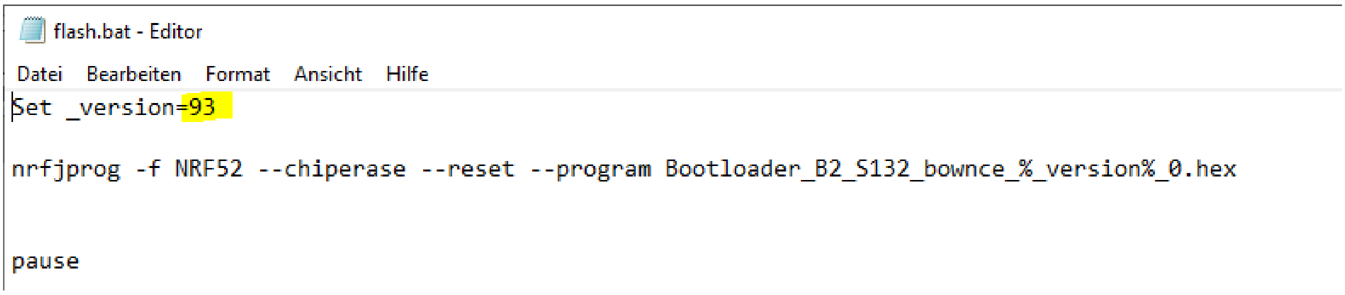
As well as the NORDIC command line tools:

nRF Command Line Tools - Downloads - nordicsemi.com

For programming the batch script

Flash.bat

Can be used:



```
Set _version=93

nrfjprog -f NRF52 --chiperase --reset --program Bootloader_B2_S132_bownce_%_version%_0.hex


pause
```

The proper _version (build number) hast to be set manually.

Notice: Offline sessions and configuration data (e.g. Device name) gets lost during this process. Device name will be reset to the default name!

When updating via DFU, offline data and configuration will remain!

4 Connecting the device via BLE

4.1 Connection parameter

Connection parameter are set according iOS Accessory-Design-Guidlines

https://developer.apple.com/accessories/Accessory-Design-Guidelines.pdf

```
#define MIN_CONN_INTERVAL               MSEC_TO_UNITS(15, UNIT_1_25_MS)        /**< Minimum
acceptable connection interval (0.15 seconds). */
#define MAX_CONN_INTERVAL               MSEC_TO_UNITS(15, UNIT_1_25_MS)        /**< Maximum
acceptable connection interval (0.15 second). */
#define SLAVE_LATENCY                   0                                       /**< Slave
latency. */
#define CONN_SUP_TIMEOUT                MSEC_TO_UNITS(4000, UNIT_10_MS)         /**< Connection
supervisory timeout (4 seconds). */

#define SEC_PARAM_BOND                  1                                       /**< Perform
bonding. */
#define SEC_PARAM_MITM                  0                                       /**< Man In The
Middle protection not required. */
#define SEC_PARAM_LESC                  0                                       /**< LE Secure
Connections not enabled. */
#define SEC_PARAM_KEYPRESS              0                                       /**< Keypress
notifications not enabled. */
#define SEC_PARAM_IO_CAPABILITIES       BLE_GAP_IO_CAPS_NONE                    /**< No I/O
capabilities. */
#define SEC_PARAM_OOB                   0                                       /**< Out Of Band
data not available. */
#define SEC_PARAM_MIN_KEY_SIZE          7                                       /**< Minimum
encryption key size. */
#define SEC_PARAM_MAX_KEY_SIZE          16                                      /**< Maximum
encryption key size. */
```

- Peripheral Latency of up to 30 connection intervals.
- Supervision Timeout from 2 seconds to 6 seconds.
- Interval Min of at least 15 ms.
- Interval Min is a multiple of 15 ms.
- One of the following:
- Interval Max at least 15 ms greater than Interval Min.
- Interval Max and Interval Min both set to 15 ms.
- Interval Max * (Peripheral Latency + 1) of 2 seconds or less.
- Supervision Timeout greater than Interval Max * (Peripheral Latency + 1) * 3.

4.2 Advertising

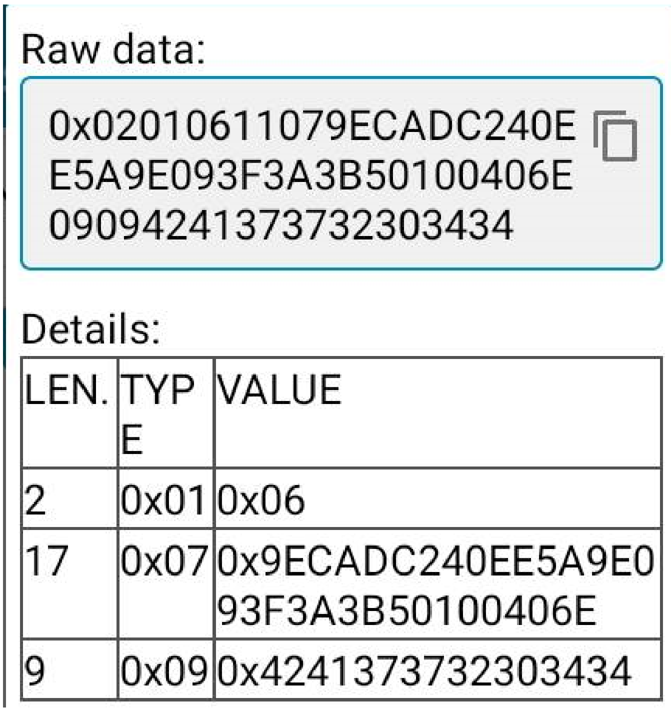
GAP is transmitting following informations during advertising (advertising package)

- **Flags** (0x01)
  General, Discoverable, Dual Mode disabled (classic Bluetooth not supported)
- **Complete List of 128Bit Service Class UUIDS** (0x07)
  Service ID oft he Nordic UART Service
  UUIDS of pre-assigned services are transmittet during device discovery
- **Complete Local Name** (0x09)
  Default Name is *BAxxxxxx* for Anchor
  and *BBxxxxxx* fort he Ball

where xxxxxx are the lowest 6 digits from the unique Microcontroller ID

The Default Name can be overwritten by the GAP Service

Example of Advertising Package:



Raw data:

```
0x02010611079ECADC240E
E5A9E093F3A3B50100406E
09094241373732303434
```

Details:

LEN.	TYPE	VALUE
2	0x01	0x06
17	0x07	0x9ECADC240EE5A9E093F3A3B50100406E
9	0x09	0x4241373732303434

4.3 Available Services

4.3.1 Generic Access (0x1800)

- Device Name (read/write) 0x2A00
- Appearence (read) 0x2A01
- Peripheral Preffered Connection Parameters 0x2A04

  (Connection Intervall: 15ms – 15ms; Slave Latency: 0; Supervision Timeout Multiplier: 400)
- Central Adress resolution (0x2AA6)

  (Address resolution supported)

4.3.2 Generic Attribute (0x1801)

- Service Change (Used for Bootloader) (0x2A05)

4.3.3 Device Information (0x180A)

- Manufacturer Name String (0x2A29)

  (Read) → ***GIE***
- Model Number String (0x2A24)

  (Read) → ***BownceBall*** / ***BownceAnchor***
- Serial Number String (0x2A25)

  (Read) → Microcontroller Device ID
- Hardware Revision String (0x2A27)

  (Read) → ***C1-Sample***
- Firmware Revision String (0x2A26)

  (Read) → ***2.17***
- Software Revision String (0x2A28)

  (Read) → ***94***[1]
- System ID (0x2A23)

  (Read) → TBD by Bluetooth SIG[2]
- IEEE 11073-20601 Regulations (0x2A2A)

  (Read) → TBD by Bluetooth SIG[3]
- PnP ID (0x2A50)

  (Read) → TBD by Bluetooth SIG[4]

[1] Software Revision String is mapped tot he build number

[2]

[3]

4.3.4 Nordic UART Service (6E400001-B5A3-F393-E0A9-E50E24DCCA9E)

- RX Characteristic (6E400003-B5A3-F393-E0A9-E50E24DCCA9E) (Write)
- TX Characteristic (6E400002-B5A3-F393-E0A9-E50E24DCCA9E) (Notify)
- Client Characteristic Configuration (0x2902)

4.3.5 Secure DFU Service (0xFE59)

- Buttonless DFU without bonds (0x8EC90003-F315-4F60-9FB8-838830DAEA50) (INDICATE, WRITE)
- Client Characteristics Configuration (0x2902)

5 Flash memory layout

Page No.	Start	End	Content
0 – 37	0	0x25FFF	Softdevice
38 – 89	0x26000	0x59FFF	Application including spare
119	0x77000	0x77FFF	nvData
120 – 125	0x78000	0x7DFFF	Bootloader
126	0x7E000	0x7EFFF	Master Boot Code
127	0x7F000	0x7FFFF	Bootloader Settings[5]

During Device Firmware Update only the pages 38 – 89 are erased and re-programmed by the bootloader. All other pages are write protected during the update process

[4]

[5] The Bootloader uses this page to store informations about the installed application firmware. E.g Version informations / build number

6 Device status indication via LED

6.1 Bownce Ball

Device status	LED indication
Application	
Device is initializing application firmware	Two green LED flashs within one second
Device is in normal operation without BLE connection	green FLASH once a second
Device is in normal operation with BLE connection established	blue FLASH once a second
Device is in abnormal state because of internal error	LED is blinking red once a second
Device is in Power-Off Mode	No indication
Bootloader	
Bootloader is starting	LED blue – red – off

6.2 Bownce Anchor

Device status	LED indication
Application	
Device is initializing application firmware	Two green LED flashs within one second
Device is calibrationg load cell	LED is permanently red
Device is in normal operation without BLE connection	green FLASH once a second
Device is in normal operation with BLE connection established	blue FLASH once a second
Device is in abnormal state because of internal error	LED is blinking red once a second
Device is in Power-Off Mode	No indication
Bootloader	
Bootloader is starting	LED blue – red – off

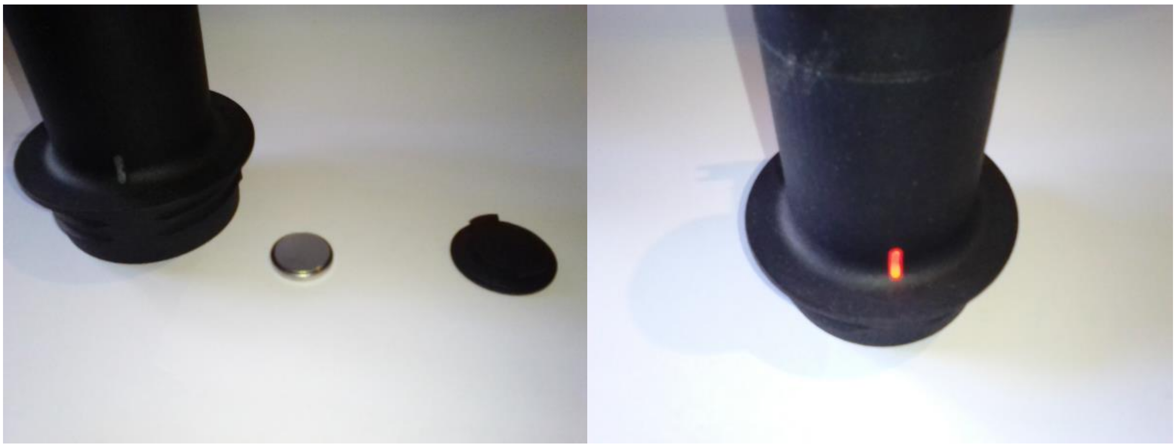
6.2.1 Anchor load cell calibration

After removing and re-installing the battery the anchor application firmware will perform a load cell offset calibration. The calibration takes about 4 seconds. The calibration values will be stored in nvData.

__Procedure for user handling:__

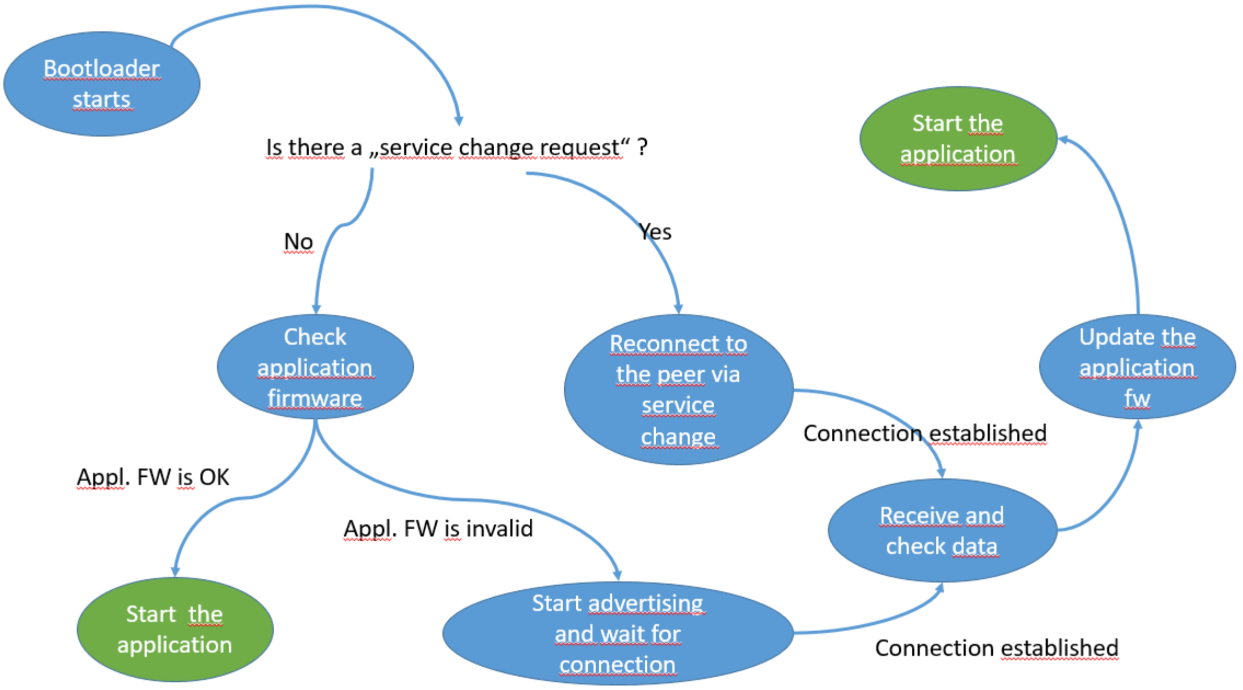
When installing the battery for the first time or replacing a discharged battery just insert the battery into the compartment and close the cover. Then leave the anchor in upright position until the red LED goes off. The anchor device automatically switches into normal operation mode.

***Important**: There **musst be no load** on the load cell during calibration. Usually that means the rope musst be deinstalled before calibration starts.*



7 Start-up procedure and Bootloader

Each time the device starts after battery re-connect or after wakeup from Power-Off mode, the bootloader starts first. The following chart gives a simplified overview about the startup process:


Bootloader starts
Is there a „service change request“ ?
No
Yes
Check application firmware
Reconnect to the peer via service change
Start the application
Update the application fw
Connection established
Appl. FW is OK
Appl. FW is invalid
Receive and check data
Start the application
Start advertising and wait for connection
Connection established

In the unlikely case that the bootloader will not find a valid application in the Flash memory[6], the bootloader itself will initialize the BLE Stack, start advertising and wait for BLE connection.

The bootloader on the Anchor device will advertise with ist default name: **BABoot**

(***B****ownce****A****nchorBoot*)

The bootloader on the Ball device will advertise with ist default name: **BBBoot**

(***B****ownce****B****allBoot*)

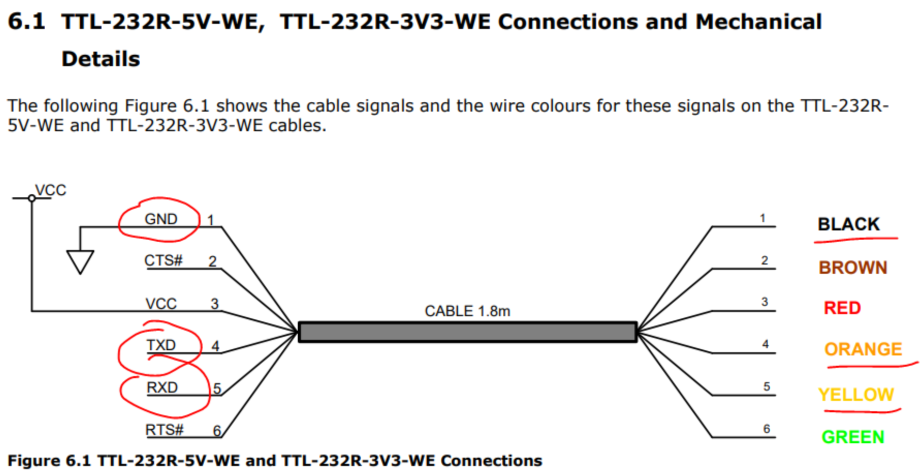
The peer on a mobile device can then connect to the bootloader as it would do when connecting the DFU Service thru the application firmware.

[6] This can happen when a running DFU process gets interrupted e.g. by removing the battery at a certain state of the process. This is very unlikely but not absolutly impossible. In any case the bootloader itself stays intact!

8 EOL UART Protocol

8.1 Ball

Connect the device via FTDI cable.



6.1 TTL-232R-5V-WE, TTL-232R-3V3-WE Connections and Mechanical Details

The following Figure 6.1 shows the cable signals and the wire colours for these signals on the TTL-232R-5V-WE and TTL-232R-3V3-WE cables.

Figure 6.1 TTL-232R-5V-WE and TTL-232R-3V3-WE Connections

Wiring:

Wiring, Testpins C4 Sample		
Signalname	Testpad on PCB	Function
BSP_EOL_UART_TX	XP209	Controller TxD
BSP_EOL_UART_RX	XP206	Controller RxD

Communication settings:

19200/8/1/none

Commands:

Note: After switching device into Off-Mode, current consumption is not reliable becouse of pull up/down resistors in FTDI Interface. For relaiable current consumption meassurement wiring hast to be disconnected!

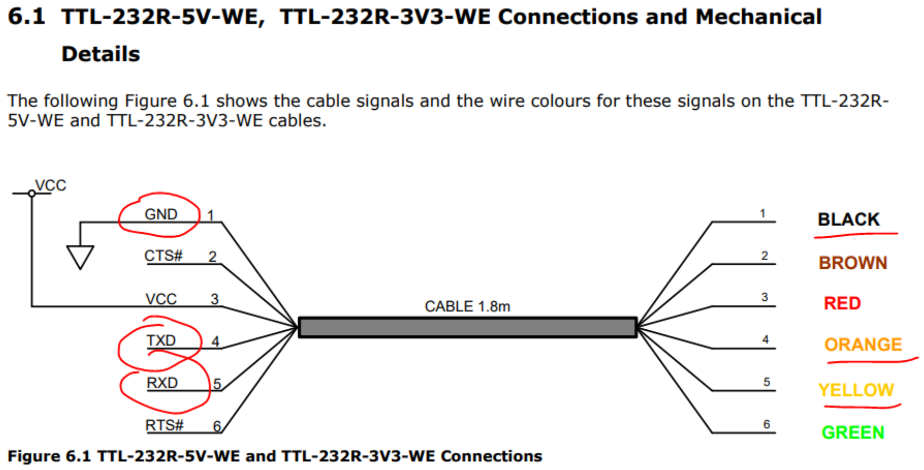
Command	Description	Response
0x01	UART_CMD_GOTO_SLEEP	No Response. Device goes into Off-Mode immediately
0x02	UART_CMD_BMI270_SELFTEST	Device performs BMI270 Selftest for acceleration and gyro. Response 0x00 → Test passed Response 0x01 → Test failed
0x03	UART_CMD_GET_FW_VERSION	Device responds with FW Version (104)
0x04	UART_CMD_GET_UID	Device responds with 8 Digit

		Controller UID
0x05	UART_CMD_ADXL_SELFTEST	Device performs ADXL Selftest. Response 0x00 → Test passed Response 0x01 → Test failed
0x09	UARTS_CMD_FINAL_DISABLE	Finally disable the EOL Test. No response

After finally disabling the EOL Uart Service, the UART can only be reactivated by flashing the device via JTAG/SWD interface.

8.2 Anchor

Connect the device via FTDI cable.



6.1 TTL-232R-5V-WE, TTL-232R-3V3-WE Connections and Mechanical Details

The following Figure 6.1 shows the cable signals and the wire colours for these signals on the TTL-232R-5V-WE and TTL-232R-3V3-WE cables.

Figure 6.1 TTL-232R-5V-WE and TTL-232R-3V3-WE Connections

Wiring:

Wiring, Testpins C5 Sample		
Signalname	Testpad on PCB	Function
BSP_EOL_UART_TX	XP205	Controller TxD
BSP_EOL_UART_RX	XP206	Controller RxD

Communication settings:

19200/8/1/none

Commands:

Note: After switching device into Off-Mode, current consumption is not reliable becouse of pull up/down resistors in FTDI Interface. For relaiable current consumption meassurement wiring hast to be disconnected!

Command	Description	Response
0x01	UART_CMD_GOTO_SLEEP	No Response. Device goes into Off-Mode immediately
0x02	UART_CMD_BMI270_SELFTEST	Device performs BMI270 Selftest for acceleration and gyro. Response 0x00 → Test passed Response 0x01 → Test failed
0x03	UART_CMD_GET_FW_VERSION	Device responds with FW Version (54)
0x04	UART_CMD_GET_UID	Device responds with 8 Digit Controller UID
0x09	UARTS_CMD_FINAL_DISABLE	Finally disable the EOL Test. No response

After finally disabling the EOL Uart Service, the UART can only be reactivated by flashing the device via JTAG/SWD interface.