UM2041

User manual

How to use the STEVAL-WESU1

Introduction

The STEVAL-WESU1 system reference design represents a highly efficient and effective solution for precise motion sensing in wearable applications.

The system embeds a low power ARM Cortex-M3 microcontroller unit (STM32L151VE), an iNEMO inertial module (LSM6DS3), a high performance magnetometer (LIS3MDL), a barometric pressure sensor (LPS25HB), a Bluetooth® low energy wireless network processor (BLUENRG-MS) and power management circuitry that allows fast charging and precise energy estimation (STNS01 and STC3115).

The connectivity, granted by the best in class BLUENRG-MS and supported by the integrated balun (BALF-NRG-01D3), combines maximum RF performance with low area occupancy and design effort. The system has passed the RF Test for FCC certification (FCC ID: S9NWESU1) and IC certification (IC ID: 8976C-WESU1).

STEVAL-WESU1 FW provides a complete framework to build wearable applications, using inertial and environmental sensor drivers, battery profile measurements, and Bluetooth low energy for data communication. It is built on the STM32Cube™ framework, which facilitates customization and the integration of further middleware algorithms.

An application layer based on the BlueST protocol (see Section 6 BlueST Protocol SDK) streams data from different devices (inertial and environmental sensors plus battery devices and RSSI) and algorithms, while a serial console over BLE allows control over the configuration parameters of the connected boards.

An ST-WESU™ app based on the the BlueST protocol logs all sensor data and provides demos for algorithms, battery detection and RSSI levels. It includes a command line interface through a debug console, allowing further control via internal permanent and session setting registers.

Figure 1. **STEVAL-WESU1 package**

1 Getting started

The STEVAL-WESU1 is a system reference design for users wanting to develop wearable applications, with every element of the system designed to accelerate the development process: from embedded end-customer devices to mobile software development.

Note: A graphical Getting started guide is available on www.st.com.

Inside the STEVAL-WESU1 package, you will find all the main components that you need to experience the demo on our special platform firmware and dedicated mobile app.

Figure 2. Inside the STEVAL-WESU1 package

The package includes:

- One STEVAL-WESU1 reference design board (max. 35X30 mm size)

Figure 3. STEVAL-WESU1 board

- One LiPo 100mAh battery (UN38.3 certified)

Figure 4. **LiPO battery**

- STEVAL-WESU1 adapter board and flat cable

Figure 5. **STEVAL-WESU1 SWD adapter**

- Silicone wristband with fastening clip

Figure 6. **STEVAL-WESU1 silicone wristband**

- Moulded plastic board and battery case, plus two snap out buttons in the plastic frame.

Figure 7. **Plastic case housing the STEVAL-WESU1 board**

- Velcro adhesive strap.

1.1 System setup guide

Follow these steps to assemble the kit:

1. Take the board and carefully remove the disclaimer tab (save it, as it contains the FCC and IC certification numbers).
2. Plug the battery to the connector on the back of the board.
3. Extract the plastic case from the silicon watch strap, and open it with a plastic tool like a pen cap.
4. Carefully place the board with the battery inside it, taking care to align the user button with the external button hole. Put one of the two buttons on this hole and then snap the case shut.
5. Connect the USB cable to turn the board on for the first time, and verify that the red battery charging LED is lit.
6. Insert the plastic case inside the silicone wristband shroud and verify button operation by pressing it to shut the board down and once more to turn the board on again.
7. Fasten the wristband clip and enjoy the experience.

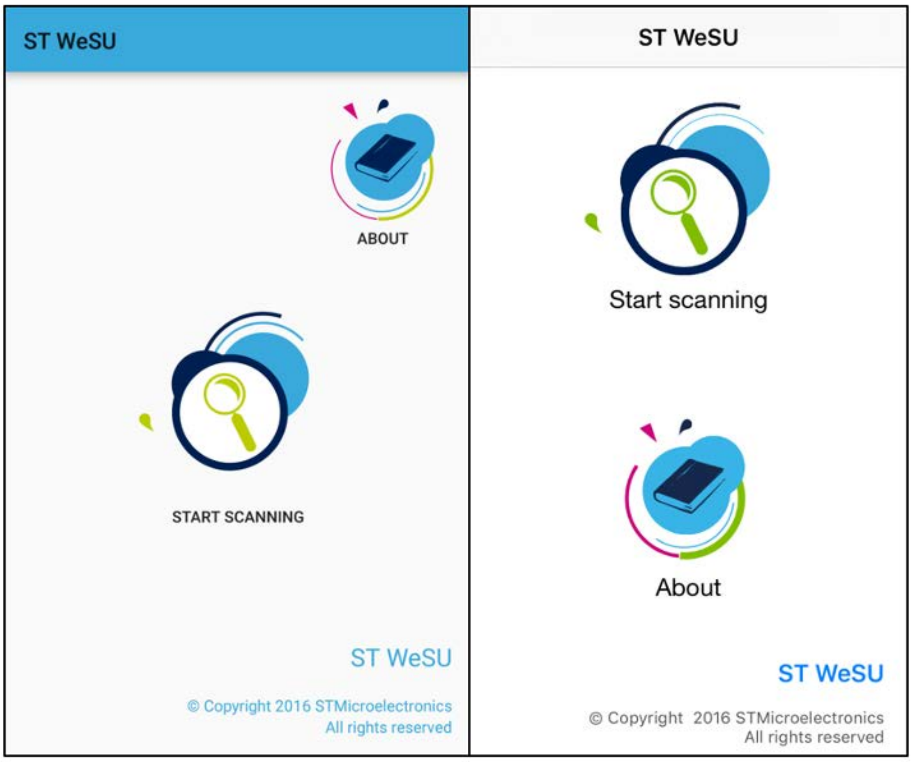
1.2 ST WeSU app setup

To visualize the information sent via Bluetooth low energy connectivity, install one of the following apps available for smartphones and tablets:

- ST WESU Android app available at Google Play™ store;
- ST WESU iOS app available at Apple Store™.

To install this app, you need a smartphone or tablet which supports BLE technology (4.0 or higher) (i.e., iPhone 4S/Android OS 4.3 and above).

Figure 8. **ST WeSU Android iOS start page**



Both versions of the app are based on a specific BlueST protocol SDK (see Section 6 BlueST Protocol SDK), which allows:

- Plotting and logging of the data from all of the sensors (supporting multiple connections)
- Sensor and algorithm demos (also with multiple connections)
- Battery/RSSI information
- Debug console for command line interface
- Configurable run time settings

1.3 System requirements

The STEVAL-WESU1 reference board requires:

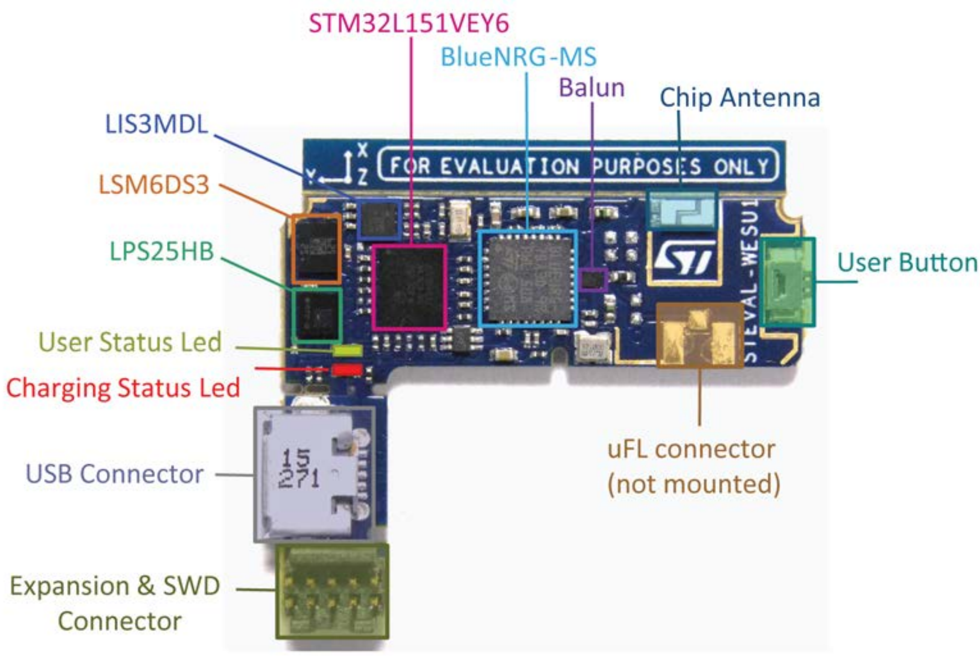
- A Windows™ (version 7, 8, 8.1 or 10) PC running an IAR, KEIL or System Workbench for STM32 firmware development environment
- One USB type A to micro B male cable to connect the STEVAL-WESU1 to the PC or wall adapter for power supply
- "ST-LINK/V2" (or equivalent) in-circuit debugger/programmer
- "ST-LINK utility" for binary firmware download (find the latest embedded software version on www.st.com).
- An Android (4.3 or higher) or iOS (8.0 or higher) smartphone or tablet with BLE technology (4.0 or higher).

2 STEVAL-WESU1 hardware description

The STEVAL-WESU1 has the following main components mounted on the top side:

- STM32L151VEY6, ultra-low-power ARM Cortex-M3 MCU with 512 Kbytes FLASH, 48kBytes of RAM in WLCSP100 package
- BLUENRG-MS, Bluetooth low energy (BLE) single-mode network processor, compliant with Bluetooth specification core 4.1
- BALF-NRG-01D3, 50 Ω balun for BLUENRG-MS transceiver with integrated harmonic filter
- LSM6DS3, iNEMO inertial module 3D accelerometer (±2/4/8/16g) + 3D gyroscope (±125/245/500/1000/2000dps)
- LIS3MDL, MEMS 3D magnetometer (±4/8/12/16 gauss)
- LPS25HB, MEMS pressure sensor, 260-1260 mBar absolute digital output barometer
- USBULC6-2M6 ultra large bandwidth ESD protection
- External oscillator LSE (32kHz) and HSE (24MHz) for the STM32L151VEY6
- User button, white user LED and red charging LED
- USB, SWD and uFL connector (not mounted)
- chip antenna

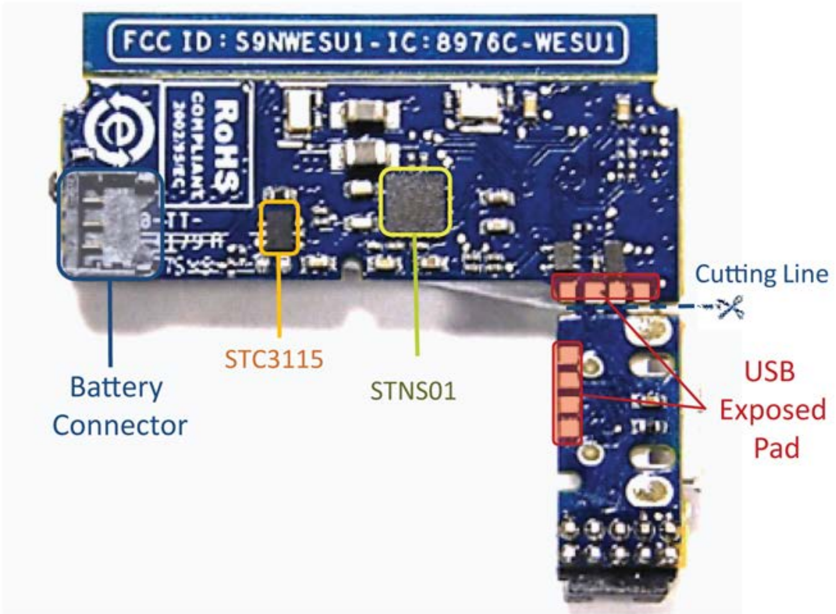
Figure 9. STEVAL-WESU1 top side components



On the bottom side, the following main components are mounted:

- STC3115, gas gauge IC with alarm output
- STNS01, Li-Ion linear battery charger
- Battery connector
- External oscillator low speed (32 kHz) and high speed (32 MHz) for BlueNRG-MS.

Figure 10. STEVAL-WESU1 bottom side components

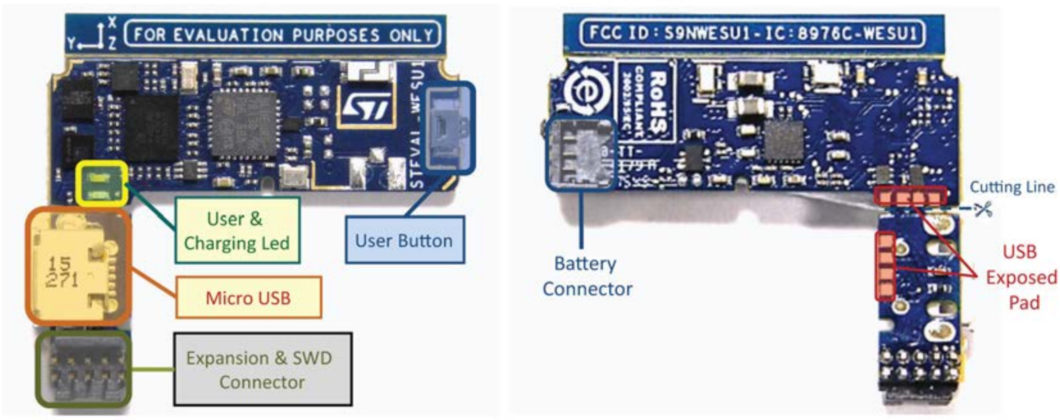


2.1 STEVAL-WESU1 board connections

The STEVAL-WESU1 includes several hardware connectors described below:

- micro-USB female plug, plus the same pins exposed in bottom pads.
- SWD connector (1.27 mm pitch)
- Battery connector

Figure 11. STEVAL-WESU1 top and bottom side board connections



2.2 ST-LINK connections

The ST-LINK V2 programmer is required to update the firmware. Plug the cable (with adapter, bundled with the package) to the board and then connect the laptop as shown below.

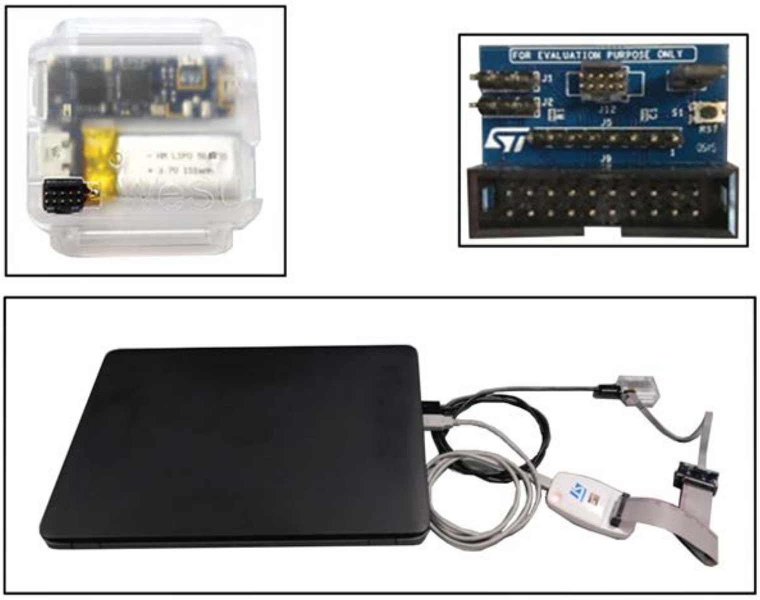
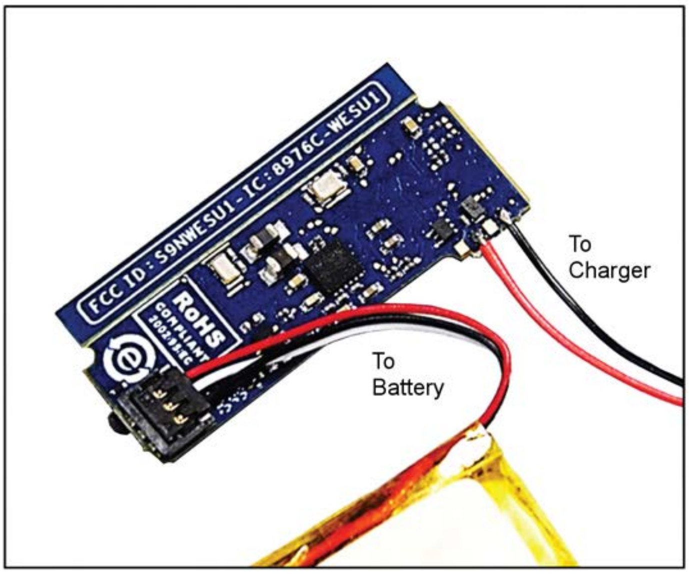


Figure 12. **ST-Link connection using the adapter**

2.3 Exposed pad connectors

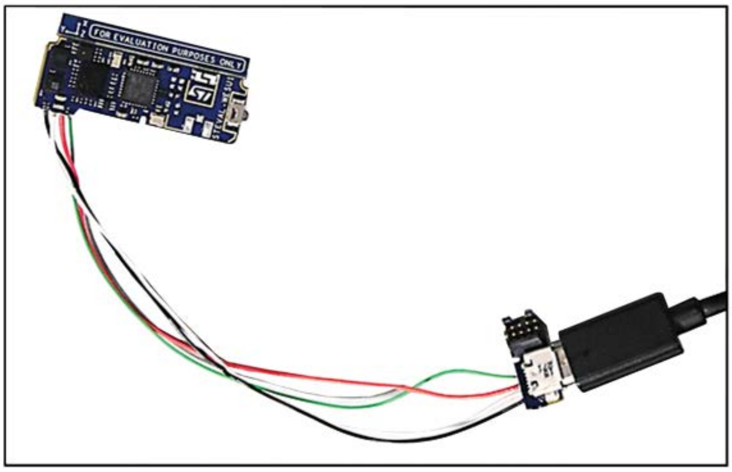
The STEVAL-WESU1 form factor can be reduced by cutting off the PCB tab hosting the USB and expansion connectors; the exposed pads on the bottom can be wired to the charger, as shown in the figure below:

Figure 13. **STEVAL-WESU1 exposed pad connections for battery charging**



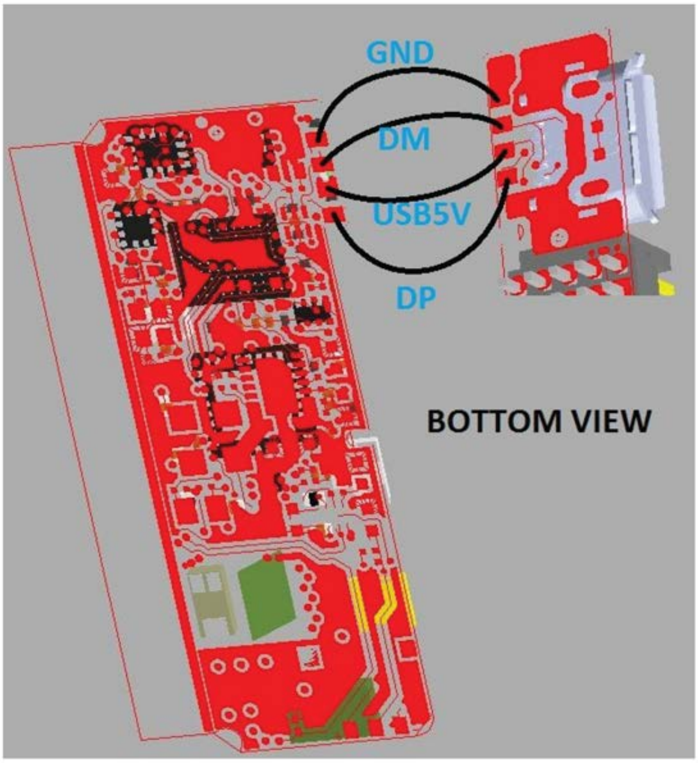
The exposed external pads can also be used to upgrade the firmware via USB using the DFU strategies with the simple four-wire connection shown below.

Figure 14. **USB plug with external USB connector for DFU update**



The USB feature is accessible via the exposed pads on the bottom of the PCB using the cable connection shown below:

Figure 15. **USB connection with external plug after cutting**



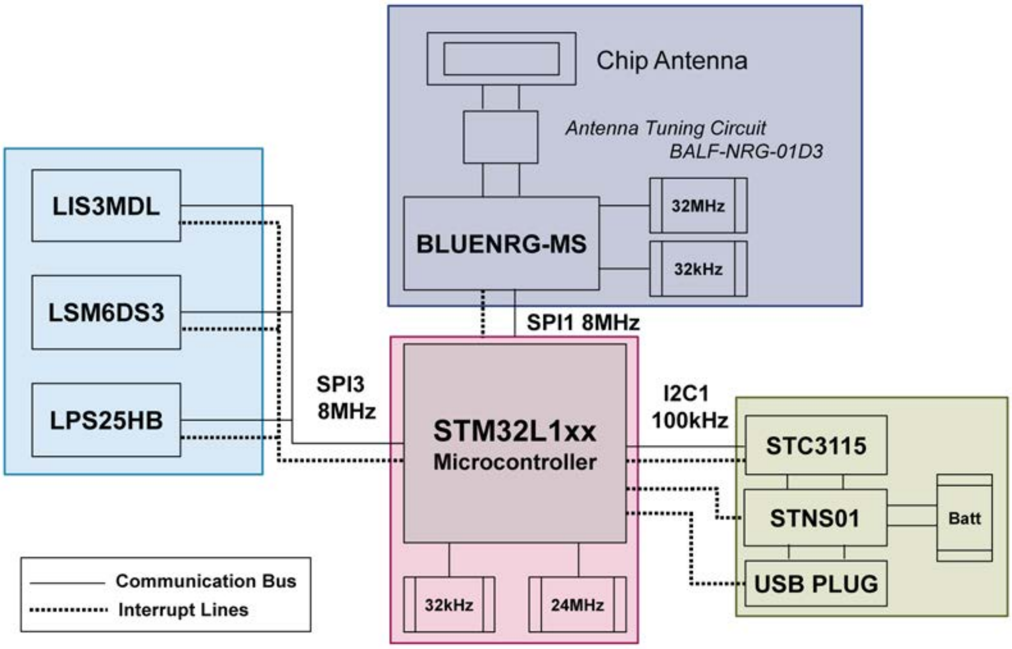
2.4 Hardware architecture

The whole system can be described in four separate functional subsystems:

- Microcontroller
- Sensors
- Connectivity
- Battery management

The sensors and the BLUENRG-MS devices are connected to the microcontroller through two separate SPI peripherals, while the power management is driven via an I²C peripheral and GPIOs.

Figure 16. STEVAL-WESU1 functional block diagram



2.4.1 Microcontroller

STM32L151VEY6 is an ultra-low-power microcontroller unit based on the ARM® Cortex®-M3, an industry standard 32-bit core designed, among other things, for low power applications. Low current consumption is crucial for wearable applications that need to wake up periodically to execute software tasks.

Figure 17. Microcontroller subsystem

The device therefore features a wide range of low power modes and voltage scaling for excellent power saving capabilities, thus minimizing consumption in different configurations.

The power consumption of the STM32L1 can be modulated by entering one of its seven low power modes; power consumption can be reduced by progressively disabling the frequency-independent current sources (clock sources, non-volatile memory and regulator) to the point where most of the chip is powered down.

In the STEVAL-WESU1 fw library, several power modes are implemented to achieve the best trade-off between power consumption and performance:

- **Full run:** microcontroller running @32MHz, all sensors configured and their output processed @100Hz to execute all the algorithms: MotionFX, Activity Recognition, Carry Position and Accelerometer Events (see note)
- **Full run WFI:** same as full run, but the microcontroller is put in wfi (wait-for-interrupt) state to save power during the explicit call to “Delay” function
- **Low power run:** microcontroller running @24MHz, sensors output processed @50Hz; all algorithms running except MotionFX (for which 100Hz is mandatory)
- **Low power:** microcontroller running @4MHz, sensors output processed @25Hz; only Accelerometer Events algorithm is running
- **Low power run with algorithms:** microcontroller running @4MHz, sensors output processed @50Hz, accelerometer, pressure sensor and gas gauge output processed; all algorithms running except MotionFX (for which 100Hz is mandatory)
- **Low power run accelerometer only:** microcontroller running @4MHz, sensors output processed @25Hz, accelerometer and gas gauge output processed; only Accelerometer Events algorithm is running
- **Low power run (MCU sleep):** microcontroller running @4MHz, sensors output processed @1Hz, accelerometer and gas gauge output processed; only Accelerometer Events algorithm is running
- **Low power run (MCU low power sleep):** microcontroller running @4MHz, sensors output processed @1Hz, accelerometer and gas gauge output processed; only Accelerometer Events algorithm is running
- **Standby with BLE PM:** microcontroller in stop mode, pedometer (see note) active, the system can be woken up by ble connection
- **Standby with BLE:** microcontroller in stop mode, can be woken up by BLE connection no sensor active
- **Standby:** microcontroller in stop mode, ble non discoverable, can be woken up by pressing the button
- **Shutdown:** the battery power path is disconnected from the system (except for the STC3115 gas gauge that is shut down); the microcontroller, the sensors and the BLE are not powered. The system can be powered by connecting the USB cable to a valid power source

Note: accelerometer events are algorithms embedded in the LSM6DS3 iNemo chip

Note: the pedometer algorithm is embedded in the LSM6DS3 iNemo chip

The mean power consumption for each powered mode is listed in table below; for more info on how to enter one of these power modes, please refer toSection 4.2.5.1 4.2.6.1 section.

Table 1. System current consumption values

Power modes	I_{DD} max (mA)	LED	BLE	Sensor	Battery GG	FX	AR	CP	AE
Full run	15.2	OFF	ON	ON	ON	ON	ON	ON	ON
Full run WFi	11.6	OFF	ON	ON	ON	ON	ON	ON	ON
Low power run	10.2	OFF	ON	ON	ON	ON	ON	ON	ON
Low power	4.0	OFF	ON	ON	ON	OFF	OFF	OFF	ON
Low power run with Algo	3.6	OFF	ON	ACC PRESS/ TEMP	ON	OFF	ON	ON	ON
Low power run with just Acc	3.0	OFF	ON	ACC	ON	OFF	OFF	OFF	ON

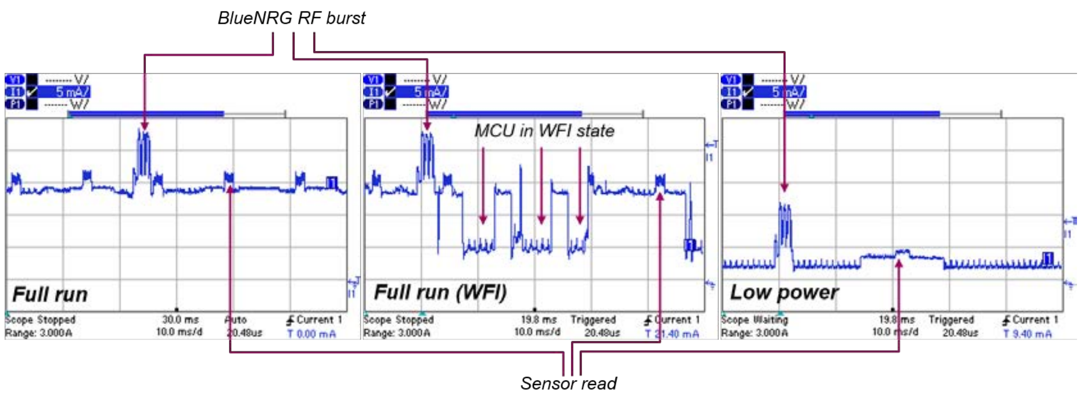
Power modes	I_{DD} max (mA)	LED	BLE	Sensor	Battery GG	FX	AR	CP	AE
Low power run (MCU sleep)	2.7	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF
Low power run (mCU low power sleep)	2.4	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF
Stand-by - Ble with PM	0.055	OFF	ON	ACC	OFF	OFF	OFF	OFF	OFF
Stand-by - Ble	0.04	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
Stand-by	0.026	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
Shutdown	0.005	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

A typical wearable application would probably use modes:

- Full Run
- Full Run with WFI
- Low Power

The current consumption measured directly on the battery path for these cases is shown in the following figure; the current waveform shows the peaks due to BlueNRG RF burst and sensor reads and the current consumption decrease in the microcontroller WFI state is also highlighted.

Figure 18. Current consumption in three typical running modes



It is possible to select one of the power modes (see table Table 1. System current consumption values) directly by APP, please refer to Section 4.2.5.1 Low power mode selection and Figure 59. Set running low power modes.

2.4.1.1 *SWD Connector and external peripheral connections*

The STEVAL-WESU1 is equipped with a custom 10 pin connector (1.27 mm pitch), which can be used:

- To program the microcontroller via a dedicated adapter connected to the programming tool (like ST-Link)
- As an expansion connector to allow user access to other board features, as described in the following table

Figure 19. SWD connector and external peripheral connections

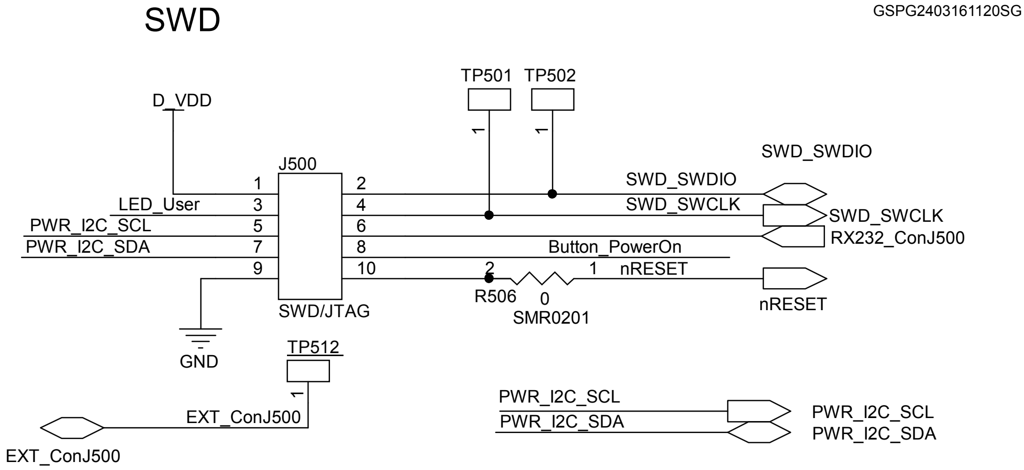
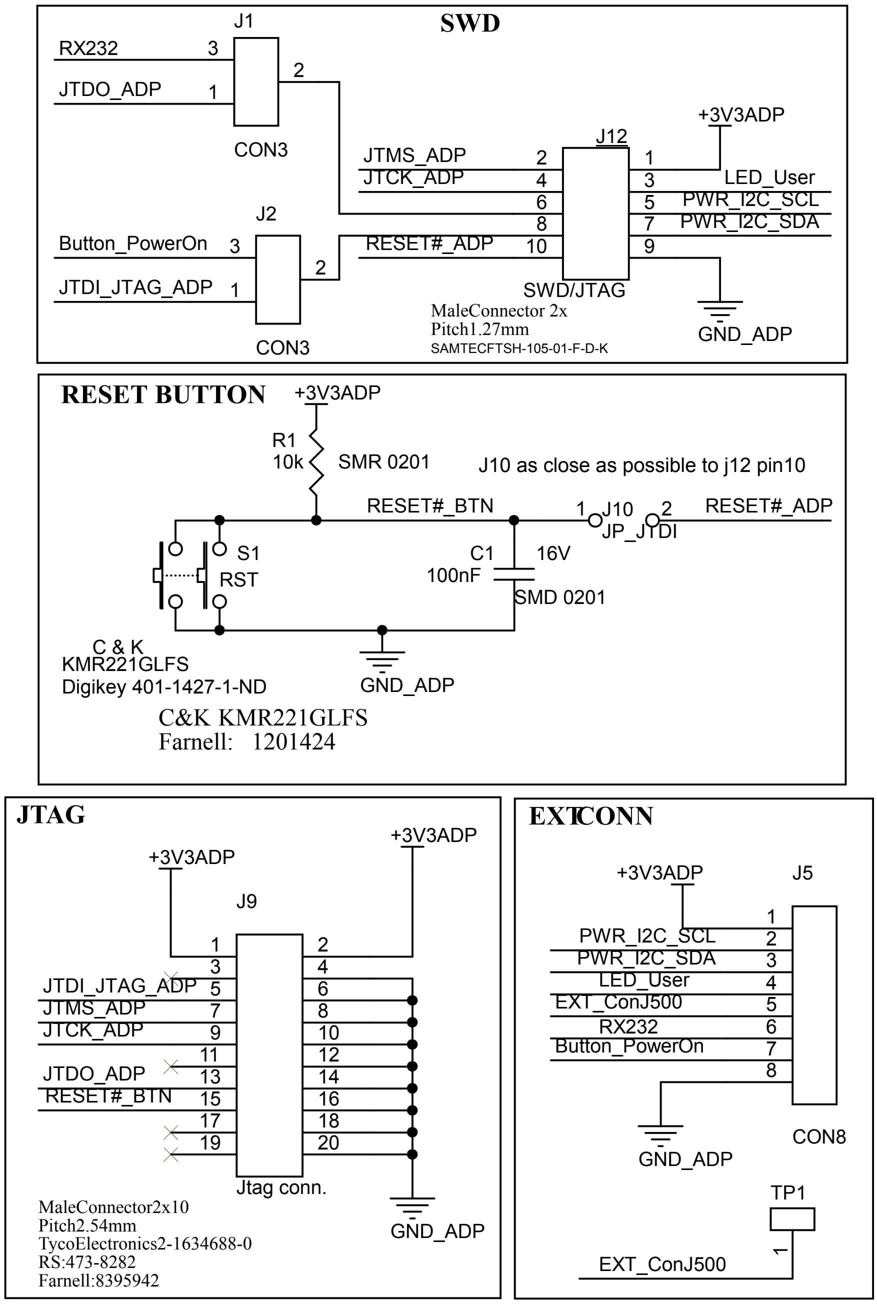


Table 2. Expansion connector GPIO description

Exp. Pin J500	Port/Pin	Default Func.	I²C	USART	PWM	ADC
3	B0	User LED			TIM3CH3	ADC_CH8
5	B8	PWR_I2C_SCL	I2C1_SCL			
6	A3	-		USART2_RX	TIM2CH4	ADC_CH3
7	B9	PWR_I2C_SDA	I2C1_SDA			
8	A2	Push Button		USART2_TX	TIM2CH3	ADC_CH2

2.4.1.2 Programming adapter

Figure 20. Programming adapter description



GSPG2403161145SG

The programming adapter can be used as an in-circuit debugger/programmer connection through J9 or as an expansion connector through J5.

It is equipped with the following jumpers:

- **J1 and J2 for the UART**: J1 and J2 must be set to the 2-3 position in order to use the J5 connector for signal monitoring on an oscilloscope or external connections through J5. Position 1-2 is reserved.
- **J10 (single position) for reset pin connection**: J10 must be fitted to enable the programmer tool to reset the microcontroller; this also enables the S1 reset button.

Note: if you connect the adapter to WeSU board with J10 fitted, a reset command is issued.

2.4.2 Sensors

The integrated sensors are perfect for motion algorithms in wearable motion tracking, featuring extremely low power capabilities and advanced performance in terms of accuracy and embedded digital features.

Figure 21. Sensor array

2.4.2.1 *LSM6DS3*

The LSM6DS3 is a system-in-package featuring a 3D digital accelerometer and a 3D digital gyroscope performing at 1.25 mA (up to 1.6 kHz ODR) in high performance mode and enabling always-on low-power features for an optimal motion experience for the consumer. Up to 8 Kbyte of FIFO with dynamic allocation of significant data (i.e., external sensors, timestamp, etc.) allows overall system power saving.

ST’s family of MEMS sensor modules leverages the robust and mature manufacturing processes already used for the production of micromachined accelerometers and gyroscopes.

The various sensing elements are manufactured using specialized micromachining processes, while the IC interfaces are developed using CMOS technology that allows the design of a dedicated circuit which is trimmed to better match the characteristics of the sensing element. The LSM6DS3 has a full-scale acceleration range of ±2/±4/±8/±16 g and an angular rate range of ±125/±245/±500/±1000/±2000 dps.

High robustness to mechanical shock makes the LSM6DS3 the preferred choice of system designers for the creation and manufacturing of reliable products. The LSM6DS3 is available in a plastic land grid array (LGA) package.

2.4.2.2 *LIS3MDL*

The LIS3MDL is an ultra-low-power high performance three-axis magnetic sensor. The LIS3MDL has user-selectable full scales of ±4/ ±8/ ±12/±16 gauss.

The device may be configured to generate interrupt signals for magnetic field detection. The LIS3MDL includes an I²C serial bus interface that supports standard and fast mode (100 kHz and 400 kHz) and SPI serial standard interface.

The LIS3MDL is available in a small thin plastic land grid array package (LGA) and is guaranteed to operate over an extended temperature range of -40 °C to +85 °C.

2.4.2.3 *LPS25HB*

The LPS25HB is an ultra-compact absolute piezo-resistive pressure sensor with an absolute range from 260 to 1260 hPa. It includes a monolithic sensing element and an IC interface able to take the information from the sensing element and to provide a digital signal to the external world.

Thanks to its high accuracy (1 Pa RMS, 24-bit ADC resolution), its bandwidth (1 – 25 Hz) and its very low power consumption (4 µA low power mode, 25 µA high performance mode), the integration of this sensor is suitable for height estimation (e.g., VRU vertical reference unit) and to enhance standard IMU performance with a high frequency altitude reference.

2.4.3 Bluetooth low energy connectivity

Figure 22. BLE connectivity subsystem

2.4.3.1 *BLUENRG-MS and BALF-NRG-01D3*

The BLUENRG-MS is a very low power Bluetooth low energy (BLE) single-mode network processor, compliant with Bluetooth specification v4.1. The BLUENRG-MS can act as master or slave. The entire Bluetooth low energy stack runs on the embedded Cortex M0 core. The non-volatile Flash memory allows on-field stack upgrading.

The BLUENRG-MS allows applications to meet the tight advisable peak current requirements imposed with the use of standard coin cell batteries. The maximum peak current is only 8.2 mA at 0 dBm of output power. Ultra low-power sleep modes and very short transition times between operating modes allow very low average current consumption, resulting in longer battery life. The BLUENRG-MS offers the option of interfacing with external microcontrollers using the SPI transport layer.

BALF-NRG-01D3 is a 50Ω conjugate match to BLUENRG-MS (QFN32 package) that integrates balun transformer and harmonics filtering. It features high RF performance with a very small footprint and RF BOM reduction. It has been chosen as the best trade-off between cost, size and high radio performance. The layout is optimized to suit a 4-layer design and a chip antenna.

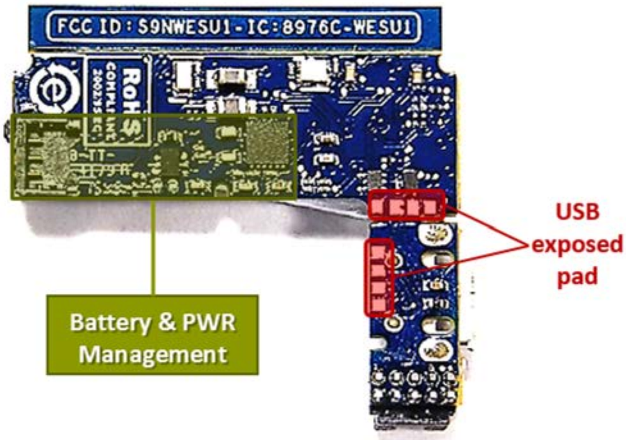
2.4.3.2 *uFL connector*

The uFL connector U201 (not mounted) is connected to the BLUENRG-MS RF path through C206 (not mounted). By soldering a 51 pF capacitor and desoldering L204, the uFL connector RF path can be activated; this is useful for debugging.

2.4.4 Battery and power management

The power management system allows USB battery charging with the STNS01 USB battery charger; battery level information is provided by the STC3115 device.

Figure 23. **Battery and power management subsystem**



2.4.4.1 *STNS01*

The STNS01 is a linear charger for single-cell Li-Ion batteries.

In the STEVAL-WESU1 system, it is configured as a battery charger and also functions as a power path switch between the USB power source and the battery power source.

The STNS01 battery charger is designed to charge single cell Li-Ion batteries up to 4.2 V using a CC-CV charging algorithm (see the STNS01 datasheet for more details). When a valid input voltage is detected, the STNS01 starts the charge cycle and the CHG pin switches from high impedance to low level. The CHG pin is connected to LED2 to monitor the charger.

The charging status LED (LED 2) can be:

- steady ON: the USB plug is correctly connected and the board is charging
- steady OFF: the board is not charging; reconnect the USB cable to force a re-start)
- flashing: charging failure (e.g., overtemperature, three-wire battery not connected), or battery not present.

The SYS pin is the voltage output of the STNS01 selected power path. This pin is connected to the linear voltage regulator providing VDD to all other devices.

The STM32L151VE is also connected to its SHDN pin to disconnect the power delivery to most of the devices and enable the shipment mode.

2.4.4.2 *STC3115*

The STC3115 includes the hardware functions required to implement a low-cost gas gauge for battery monitoring. The STC3115 uses current sensing, Coulomb counting and accurate measurements of the battery voltage to estimate the state-of-charge (SOC) of the battery. An internal temperature sensor simplifies implementation of temperature compensation.

An alarm output signals a low SOC condition and can also indicate low battery voltage. The alarm threshold levels are programmable.

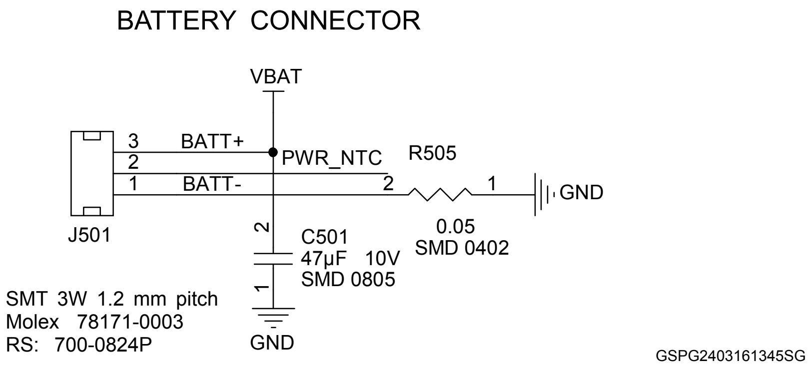
The STC3115 offers advanced features to ensure high performance gas gauging in all application conditions.

2.4.4.3 *Battery connector*

Battery connector is placed as shown in Figure 11. STEVAL-WESU1 top and bottom side board connections, and the hardware connection is as shown in the figure below.

Resistor R505 is placed as close as possible to this connector in order to sense the battery current more accurately, the value of which is then monitored by the STC3115 device.

Figure 24. Battery connector

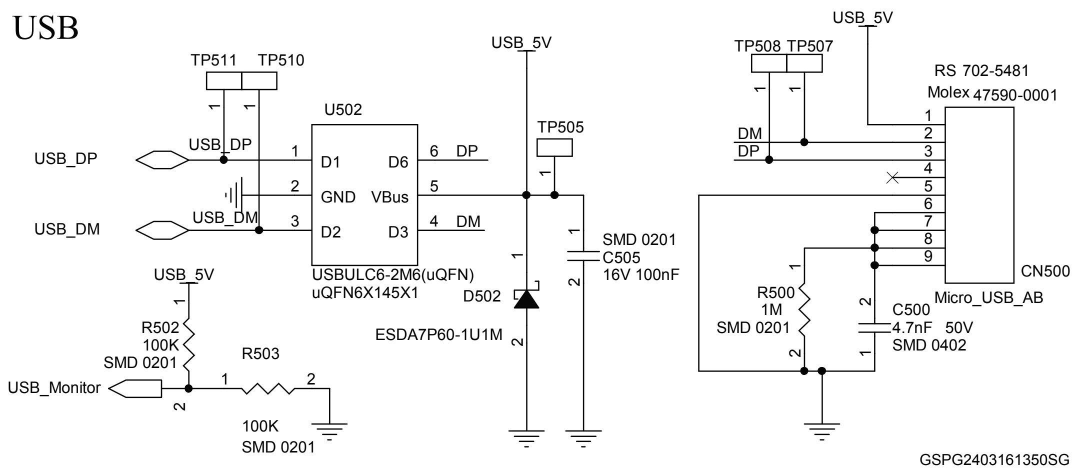


2.4.4.4 *USB connector*

The USB connector accepts a micro USB type B and it is used to charge the battery and to power the board even if battery is not present.

The USBULC6-2M6 ESD protection just after the USB connector, avoids any voltage spike damages, due to the plug operations, towards all the devices powered.

Figure 25. USB connector with ESD protection



3 STEVAL-WESU1 Firmware

3.1 Overview

This firmware package expands the functionality of the STM32Cube platform adding the following features to build a wearable application:

- Complete mid-level driver set necessary to build applications using pressure and temperature sensor (LPS25HB) and motion sensors (LIS3MDL and LSM6DS3)
- Data logger sample application about inertial and environmental sensors measurements, battery profile measurements, and algorithm demos. The data acquisition from different sensors is provided via SPI, and battery charging profile via I2C.
- Complete middleware to easily communicate with a BLE client application using a proprietary protocol (BlueST Protocol SDK).
- Low power setting configurable by app.
- Command line interface (CLI) using a debug console by app.
- Configuration interface: using EEPROM (permanent) and RAM (session) settings.
- OTA/USB-DFU: Firmware upgrade over the air through BLE connectivity (using BlueST APP) or USB IF.

3.2 Architecture

The firmware is based on the STM32Cube™ framework technology developed to build applications with the STM32 microcontroller.

The package provides a board support package (BSP) for the sensors and the middleware components for Bluetooth low energy communication with any external mobile device.

The firmware driver layers to access and use the hardware components are:

- **STM32Cube HAL layer**: simple, generic, multi-instance APIs (Application Programming Interfaces) which interact with the upper layer applications, libraries and stacks. These APIs are based on the common STM32Cube framework so other layers like the middleware layer can function without requiring specific hardware information for a given microcontroller unit (MCU). This structure improves library code reusability and guarantees easy portability across other devices.
- **Board support package (BSP) layer**: provides firmware support for the STM32 Nucleo board (excluding MCU) peripherals. These specific APIs provide a programming interface for certain board specific components like LEDs, user buttons, etc., but can also be used to fetch board serial and version information, and support initializing, configuring and reading data from sensors. The **BSP** provides the drivers for the STEVAL-WESU1 board peripherals, adding the connections to the microcontroller peripherals.

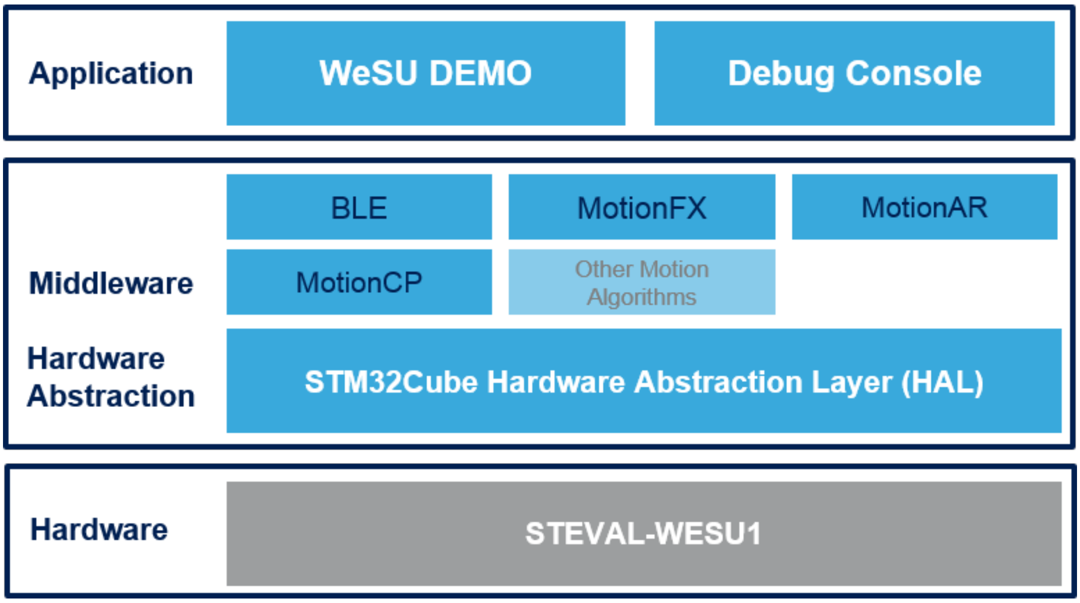


Figure 26. **STEVAL-WESU1 firmware architecture**

These specific APIs provide a programming interface for the on-board peripherals, but they can also be used to provide user support for initializing, configuring and reading data from communication buses, specific to STEVAL-WESU1.

This helps the user build the firmware using specific APIs for the following hardware subsystems:

- **BNRG**: to control connectivity
- **Platform**: to control and configure all the devices in the battery and power subsystem plus button, LEDs and GPIOs
- **Sensors**: to link, configure and control all the sensors

3.3 Folder structure

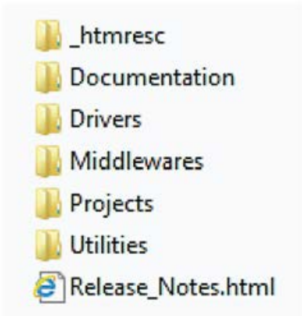


Figure 27. **STEVAL-WESU1 firmware package folder structure**

Embedded firmware library version information is included in “Release_Notes.html”.

3.3.1 Documentation

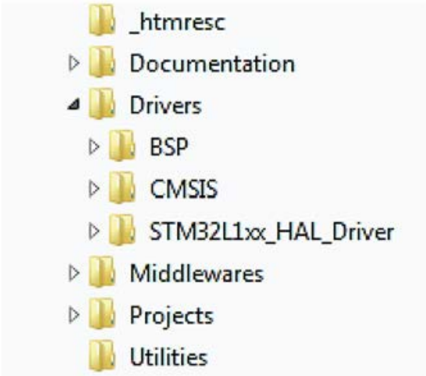
Contains a compiled HTML file generated from the source code and documentation that describes the firmware framework, drivers for the on-board components and APIs to manage all the different functions. See the “STEVAL-WESU1_FW.chm” manual for further details.

3.3.2 Drivers

All firmware packages adhering to the STM32Cube framework contain the following main groups:

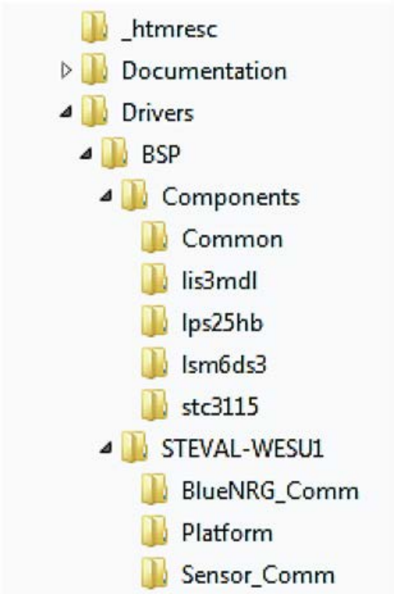
- **BSP**: the board specific drivers for whole the HW components
- **CMSIS**: vendor-independent hardware abstraction layer for ARM Cortex-M series.
- **STM32L1xx_HAL_Drivers**: microcontroller HAL libraries

Figure 28. **firmware drivers folder**



- **Components**: platform independent device drivers for LPS25HB, LIS3MDL, LSM6DS3, and STC3115.
- **STEVAL-WESU1**: mid-level drivers for each hardware subsystem, giving the developer application-level control of the BlueNRG communication, platform, and sensors subsystems.

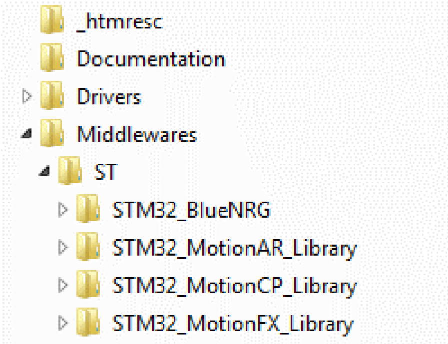
Figure 29. **BSP folders**



3.3.3 Middleware

Libraries and protocols for BlueNRG Bluetooth low energy, and other algorithm libraries, including sensor fusion, real time activity recognition and carry position recognition libraries. Further integration of ulterior algorithm is facilitated by the common STM32Cube framework.

Figure 30. Middleware folder



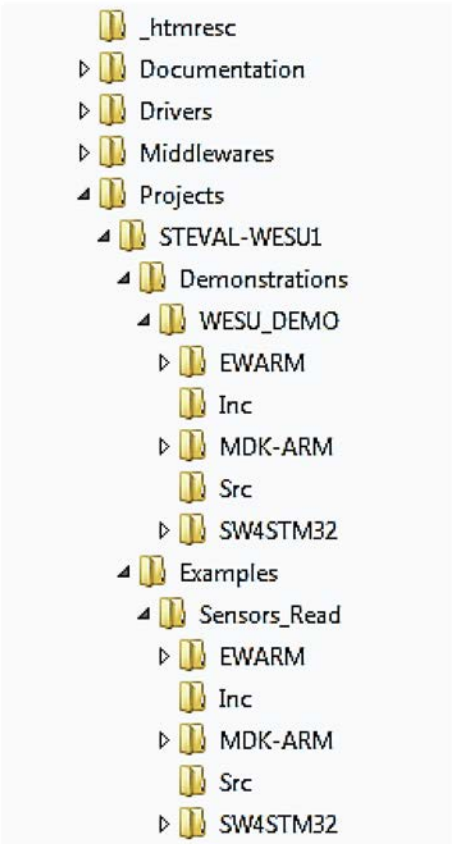
3.3.4 Projects folder

This directory contains a “Demonstrations” folder with a project that directly supports the ST WeSU app and an “Examples” folder with a project providing reference firmware.

With the ST WeSU app on an Android or iOS device, it is easy to display sensor and algorithm data in customized plots or demos.

The demonstration firmware aims to provide the functions available in the ST WeSU app, according to the BlueST protocol (see Section 6 BlueST Protocol SDK).

Figure 31. Demonstrations and Examples folders



To set up a suitable development environment for creating applications for the STEVAL-WESU1, the projects are available for the following environments:

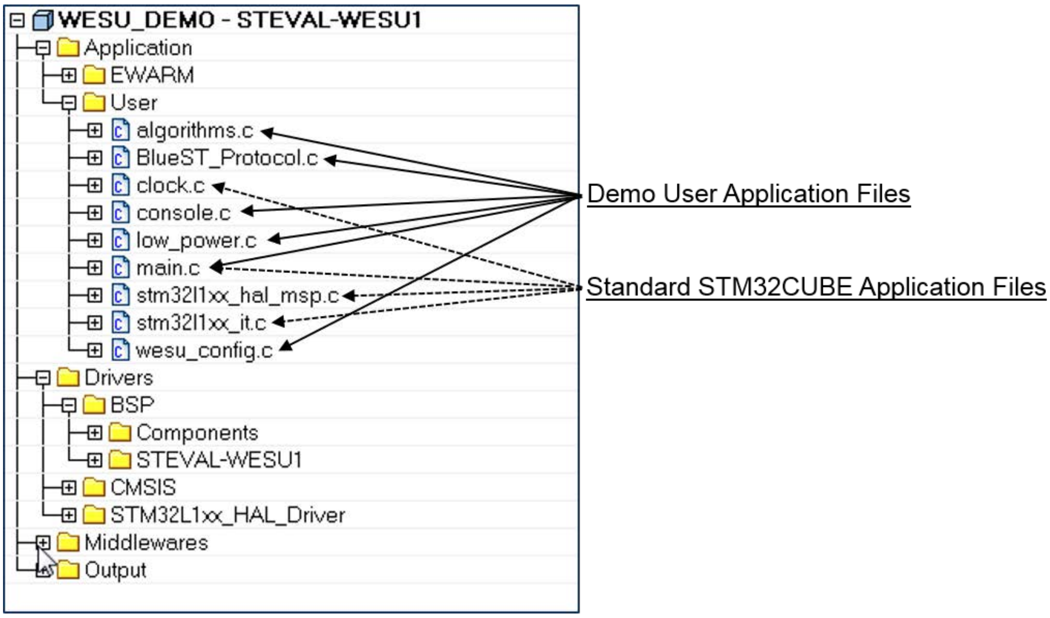
- **IAR** Embedded Workbench for ARM® (**EWARM**)
- **KEIL** RealView Microcontroller Development Kit (**MDK-ARM**)

- **System Workbench** for STM32 (**SW4STM32**)

3.3.5 Demonstration firmware overview

Typical demonstration firmware developed from the STM32Cube framework is found in an application folder containing the files groups described below:
- Demo user application files
- Standard STM32Cube application files

Figure 32. **Demo application folder**



The demo application files include all the functions to support the ST WeSU mobile app:
- **algorithms.c**: APIs to perform configuration, initialization, run, and processing for the algorithms included in the middleware.
- **BlueST_Protocol.c:** APIs to support the BlueST SDK protocol (see Section 6 BlueST Protocol SDK) in terms of service, characteristics, and data transmission/reception with BLE.
- **console.c**: APIs to provide the CLI commands for the debug console.
- **low_power.c:** APIs to configure the low power modes for the system
- **main.c**: The main demonstration routine including the top-level APIs
- **wesu_config.c:** APIs to configure, initialize, and check system-level functions and to control the permanent and session registers.

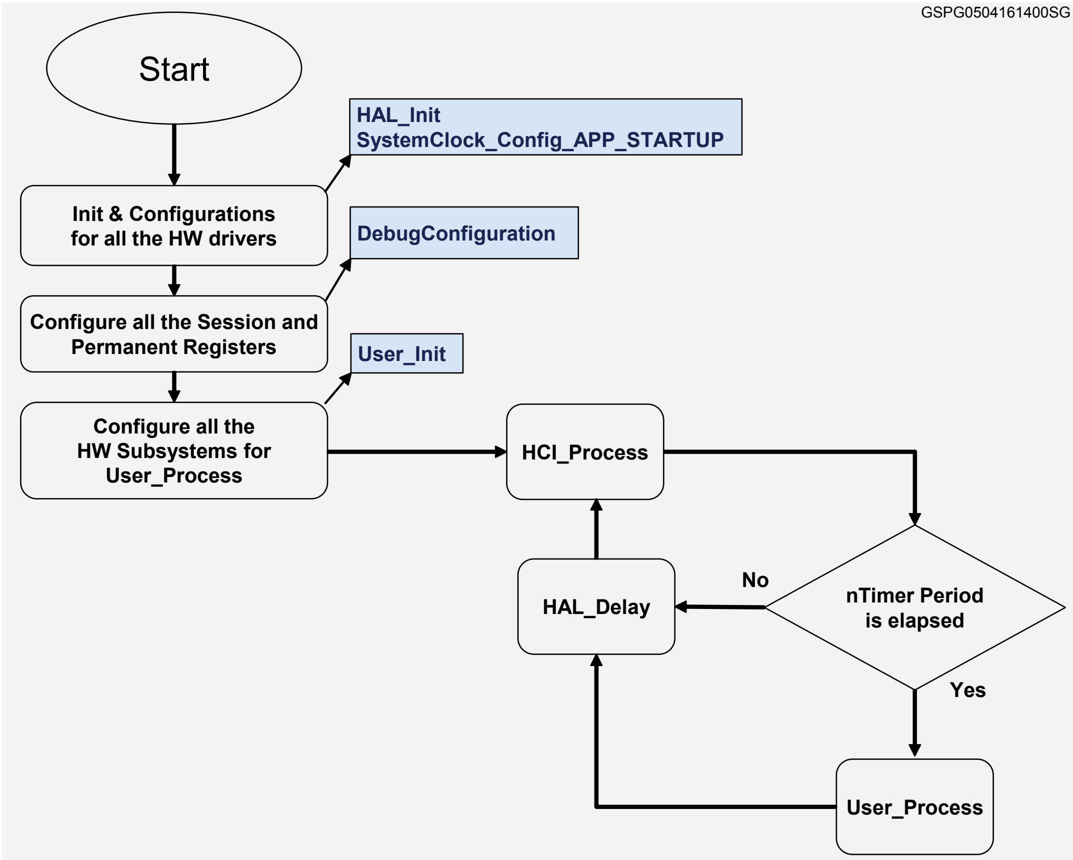
The standard STM32Cube application files have the same configuration as any standard example using the STM32 HAL libraries, with the simple addition of the peripherals used for the demo purposes in the following files:
- clock.c
- main.c
- stm32l1xx_hal_msp.c
- stm32l1xx_hal_it.c

3.3.5.1 *Demo main*

The main file includes the following functions:
- `HAL_Init`: configures the FLASH prefetch, time base source, NVIC and HAL low-level driver APIs
- `Error_Handler`: executed in case of error
- `HAL_Delay`: commonly used in any standard HAL example to provide an accurate delay (in milliseconds) based on the SysTick timer as time base source; in this case, it is a user file implementation using the _WFI instruction

- `RTC_Config`: configures RTC prescaler and data registers
- `RTC_TimeStampConfigDefault`: configures the default time and date
- `SystemClock_Config_APP_STARTUP`: configures the main system clock; suitable for application using one of the following clock configuration functions:
  - `SystemClock_Config_HSI_32MHz`
  - `SystemClock_Config_RTC_HSE32MHz`
  - `SystemClock_Config_HSI_12MHz`
  - `SystemClock_Config_HSE_18MHz`
  - `SystemClock_Config_MSI_2MHz`
- `DebugConfiguration`: configures debug facilities
- `User_Init`: initializes all the firmware sub-blocks used by the `User_Process` function; specifically, it:
  - Initializes register management
  - configures LED GPIO, button GPIO and EXTI Line
  - configures the RTC
  - initializes all the devices in the PWR Subsystem
  - initializes the BlueNRG subsystem management
  - initializes the sensor subsystem management
  - initializes all the algorithms
- `User_Process`: controls the process which is continuously executed in the main loop; specifically, it executes the following actions:
  - get data time to use RTC as calendar
  - backup and save the system parameters
  - switch the system in run mode
  - manage the LED in order to give different feedback for each step executed
  - get all the sensors data
  - execute the algorithms
  - control the BLE connection
  - manage the debug console
  - control the low power transitions using the user button or the app
- `Read_Sensors`: checks whether the sensor is enabled in the configuration registers. If enabled, get the data and save the value directly in the corresponding session registers
- `ManageBleConnection`
  - check BLE connection
  - update the connection parameters
  - update the sensors and other data to be sent through BLE, according to the corresponding configuration
- `HCI_Process`: BlueNRG HCI provides a standard interface for accessing the capabilities of the Bluetooth controller and BlueNRG LE stack.
- `Main`: uses the predefined APIs to execute the principal process and perform the demo application. This procedure allows the user to execute the most important `User_Process` function, the interval in ms is adjusted using the “nTimerFrequency” value. The functions executed include:
  - `HAL_Init();`
  - `SystemClock_Config_APP_STARTUP();`
  - `DebugConfiguration();`
  - `User_Init();`
  - `DBG_PRINTF(Firmware version);`
  - `APP_BSP_LED_Off(LED);`
  - Infinite loop: `HCI_Process()`; check if the timestamp has increased by the value “nTimerPeriod”; `User_Process()`; `HAL_Delay` (1 ms)

Figure 33. Main function with internal infinite loop



3.3.5.2 *BLE services*

This firmware uses three Bluetooth services:

- `HW_Features_Service`: transmits hardware characteristics, including:
  - Temperature
  - Pressure
  - Battery voltage, battery current, battery SOC, battery status
  - 3D gyroscope, 3D magnetometer, 3D accelerometer
  - Sensor fusion data (AHRS values)
  - Algorithm1 data (activity recognition)
  - Algorithm2 data (carry position)
  - Algorithm3 data (Pedometer provided by HW using LSM6DS3)
  - Algorithm4 data (FreeFall provided by HW using LSM6DS3)
- `Configuration_Service`: for configuration management according to the BlueST protocol:
  - Register management characteristics
- `Console_Service`: to transmit two main characteristics:
  - Terminal
  - Stderr

This package is compatible with the ST WeSU Android/iOS application (Version 1.0 and above) available at Google Play/Apple Store, respectively.

This application can be used to display information sent with the BlueST SDK Protocol (Section 6 BlueST Protocol SDK).

3.3.5.3 *Memory mapping*

The demo application firmware is developed in order to provide all the functions necessary to support mobile app data streaming, remote configuration settings, and firmware upgrade using DFU or OTA.

To provide these features, specific firmware memory mapping has been implemented as in the following tables.

Table 3. Memory mapping

Memory type	Section	Start address	End address	Size (bytes)	Definition
INTERNAL FLASH (512K)	USB DFU[1]	0x08000000	0x08002FFF	12 K	DFU using USB connection
	Reset Manager	0x08003000	0x080037FF	2 K	Application OTA management
	Service manager[2]	0x08003800	0x08007FFF	18 K	OTA (over the air) firmware upgrade
	BlueNRG GUI	0x08008000	0x0801FFFF	96 K	BlueNRG GUI USB-BlueNRG bridge
	Application	0x08020000	0x0807FFFF	384K	Application Used for application firmware code: Demo application or Sensor example

1. *Activation through the APP, using settings page or at startup/reboot, pressing the button for at least 3 seconds*
2. *Activation through the APP, using settings page or at startup/reboot, without USB cable connection, pressing the button during the first led blink*

Internal EEPROM and RAM are used for the application; specific areas are dedicated for register implementation. The following features are directly adjustable with the mobile app (marked with a double asterix (**) in the following tables):

- configure and control the on-board functions related to the microcontroller, sensors, connectivity and power management..
- store the data from the sensors, battery charging devices and other on-board devices, including system status information.
- allow remote control via the mobile app using settings menu or debug console in the ST WeSU app.

Some information (e. g., the accelerometer full scale or output data rate settings) is stored in the permanent registers only, while other data (e. g., sensor output data) is stored in the session registers; some information (e. g., the timer setting) is stored in both locations for temporary and permanent modifications.

A detailed description of permanent and session registers is available in the STEVAL-WESU1.chm.

Table 4. Permanent register location

Memory type	Section	Start address	End address	Size (bytes)	Definition
INTERNAL EEPROM (16K)	USER EEPROM AREA	0x08080000	0x08080FEF	4080	
	NEW_APP_MEM_INFO	0x08080FF0	0x08080FF7	8	
	USB_DFU_MEM_INFO	0x08080FF8	0x08080FFF	8	
	**PERMREG_STRUCT START_ADDRESS	0x08081000	0x080813FF	1K	PERSISTENT REGS
	**PERMREG_STRUCT_BCK START_ADDRESS	0x08081400	0x080817FF	1K	PERSISTENT BCK REGS
	NOT USED	0x08081800	0x08082DFF	5632	reserved
	TEST_ID_EEPROM ADDRESS	0x08082E00	0x08082EDF	224	
	TEST_DATETIME_EEPROM ADDRESS	0x08082EE0	0x08082EFF	32	
	PRODUCTION_DATA START_ADDRESS	0x08082F00	0x08082FFF	256	
	NOT USED	0x08083000	0x08083FFF	4K	reserved

Table 5. Session register location

Memory type	Section	Start address	End address	Size (bytes)	Definition
INTERNAL RAM (80K)	USER RAM	0x20000000	0x20012FFF	76K	
	**SESSION REGISTERS	0x20013000	0x200133FF	1K	SESSION REGS
	RFU (Reserved Future Use)	0x20013400	0x20013FFF	3K	

3.3.6 Sample firmware overview

The sample sensor firmware gives insight into the development of simple applications involving data from a single sensor, use the BLE service and features, and other functions called by the ST WeSU app, in accordance with the BlueST protocol (Section 6 BlueST Protocol SDK).

Firmware examples developed using the STM32Cube framework require an “Application” folder with the files being either:

- Example application files
- Standard STM32Cube application files

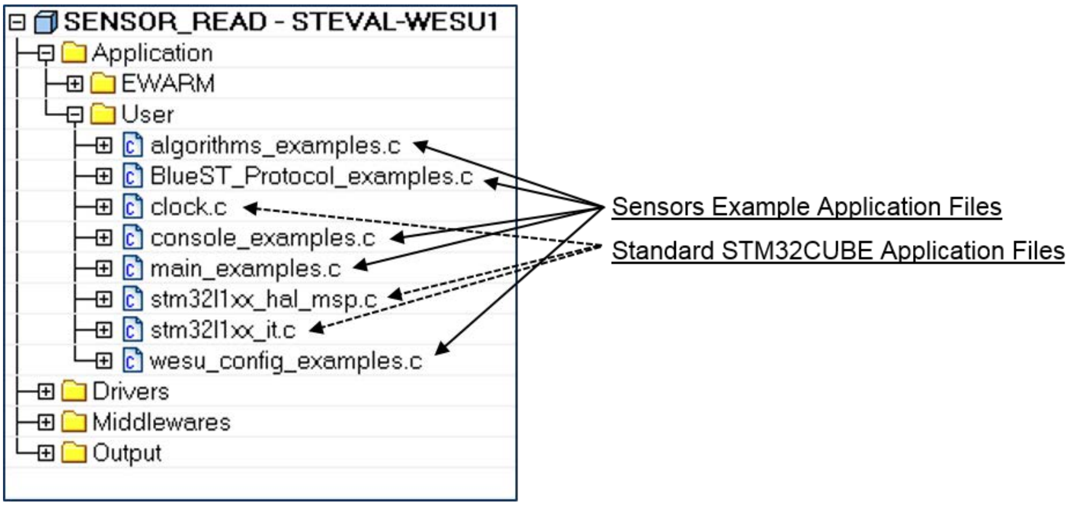


Figure 34. Example application files package

The “Example” application files are the same as those for the “Demonstration” application, with some differences for certain files (identified with “filename_example.c”). You can use the ST WeSU mobile app to evaluate data streaming as shown in the demo.

3.4 Toolchains

The STM32Cube expansion framework supports the following development tool-chain and compiler environments so you can build applications with the STEVAL-WESU1:

- IAR Embedded Workbench for ARM® (EWARM) toolchain + ST-LINK:
  - open IAR Embedded Workbench (v8.20 and above)
  - open the IAR project file EWARM\Project.eww
  - rebuild all files and load your image into target memory
  - run the application
- KEIL RealView Microcontroller Development Kit (MDK-ARM) toolchain + ST-LINK:
  - open µVision (V26.2 and above) toolchain
  - open the µVision project file MDK-ARM\Project.uvprojx
  - rebuild all files and load your image into target memory
  - run the application
- System Workbench for STM32 + ST-LINK:
  - open System Workbench for STM32 (2.6.0.20180906 and above)

Note: AC6 C/C++ Embedded Development Tools for MCU version 1.12.0 (and above) is required for proper flashing and debugging of the STEVAL-WESU1 board. V1.11 is recommended for “Debugging tools for MCU” and “AC6 Linker Script Editor”.

  - set the default workspace proposed by the IDE (ensure that they are not placed in the workspace path)
  - select "File" -> "Import" -> "Existing Projects into Workspace"; press "Browse" in "Select root directory" and choose the path where the System Workbench project is located
  - rebuild all files and load your image into target memory
  - run the application

When establishing your IDE workspace, ensure that the folder installation path is not too deep to avoid any eventual toolchain errors.

Any further firmware development can start using the reference projects included, and the SWD as the mandatory debug interface.

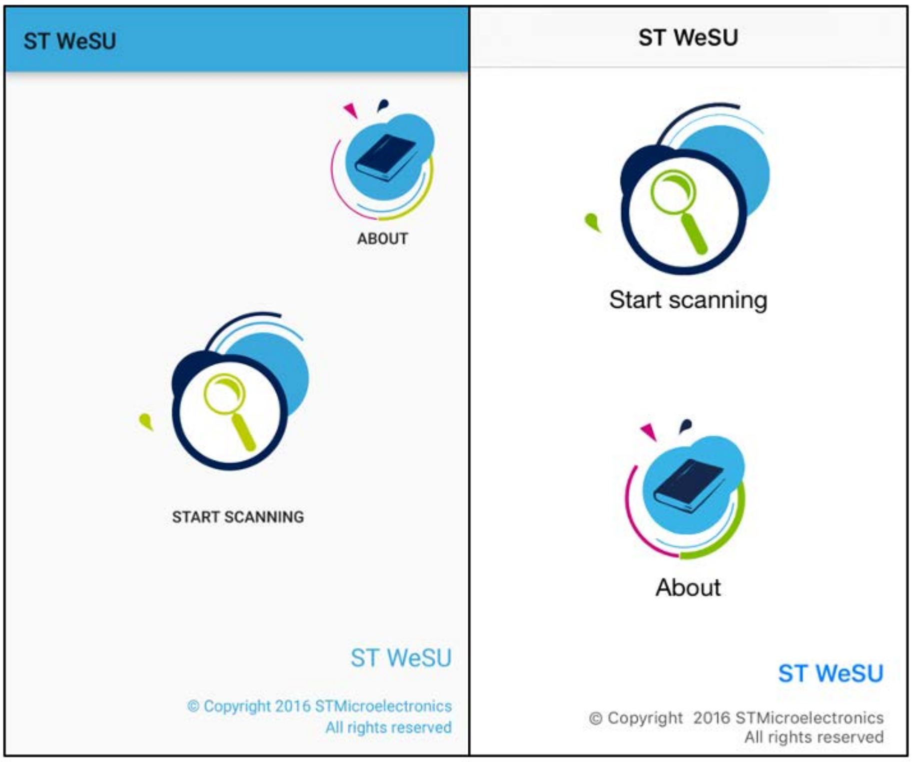
4 ST WeSU app

The STEVAL-WESU1 demo application firmware is designed to work with the ST WeSU App (Android ver. 4.4.0 / iOS Ver. 8.0, or above), available free of charge at Google Play and Apple Store.

STBLESensor app is also available (www.st.com/en/product/stblesensor).

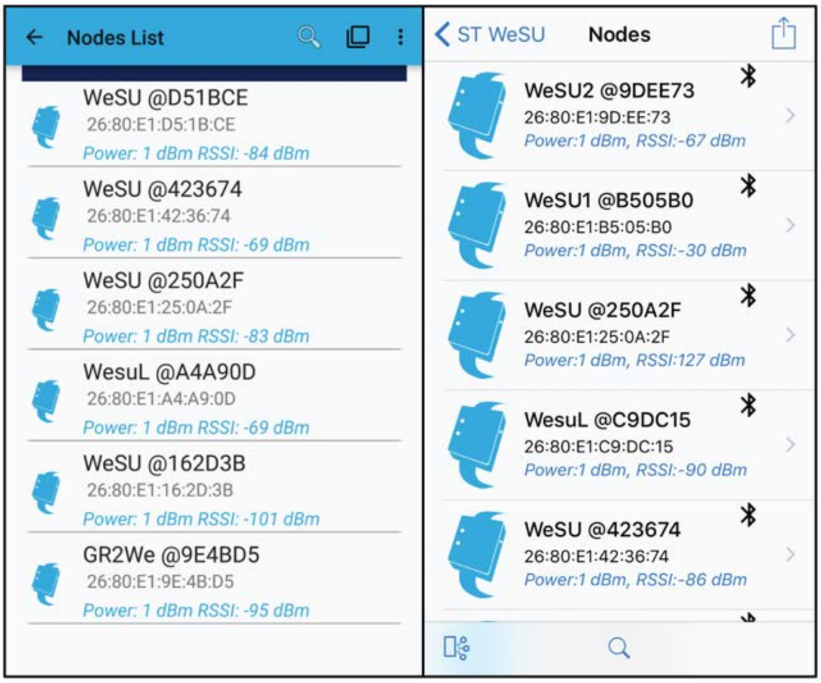
The ST WeSU app opens with the start page below, from which you can command your mobile device to scan for nearby nodes.

Figure 35. **ST WeSU app start page (Android and iOS)**



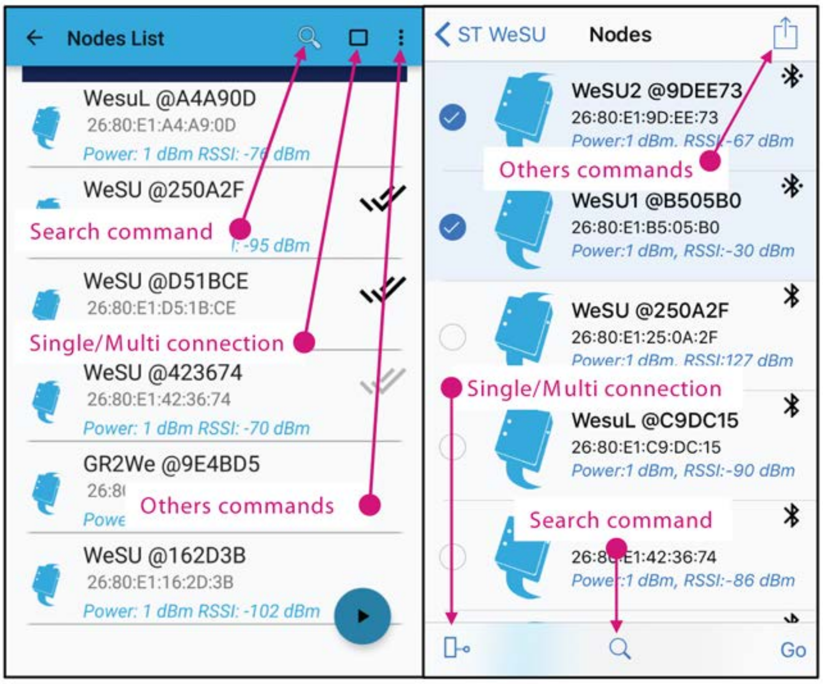
You can then choose from one of the available boards to start all the demos and other supported functions.

Figure 36. **ST WeSU app (Android and iOS) device list**



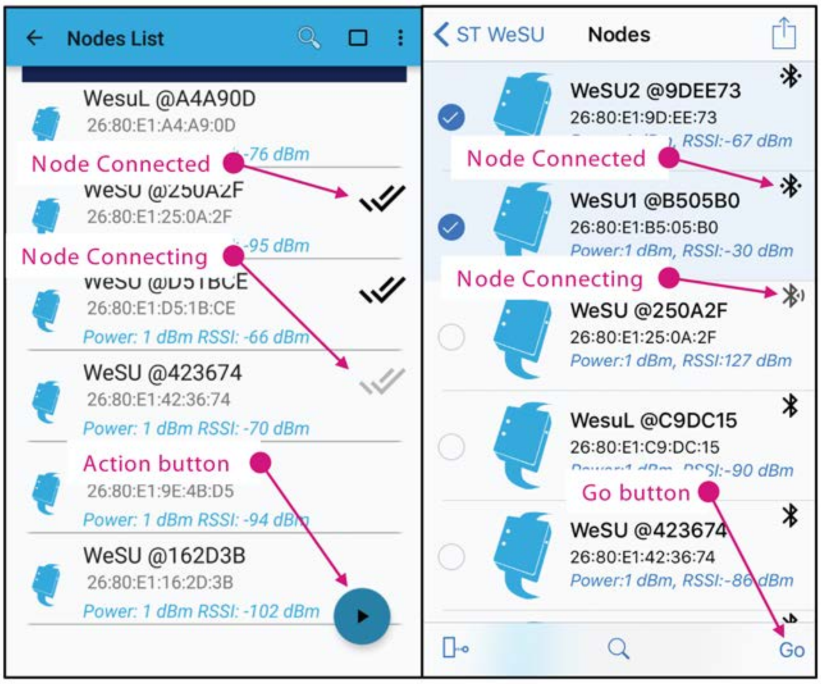
The app is able to support multiple node connections and switch from single to multiple connections from the menu options at the top of the page; alternatively, you can simply activate multiple connections by pressing and holding down an available node item.

Figure 37. **Node list (Android and iOS) commands**



The double tick (✔) mark for multiple nodes on the Android page is black when a particular node is connected, and gray when the node is connecting; in iOS, the tick represents a connected node. To start the demos you need to press the Action/Go button shown below.

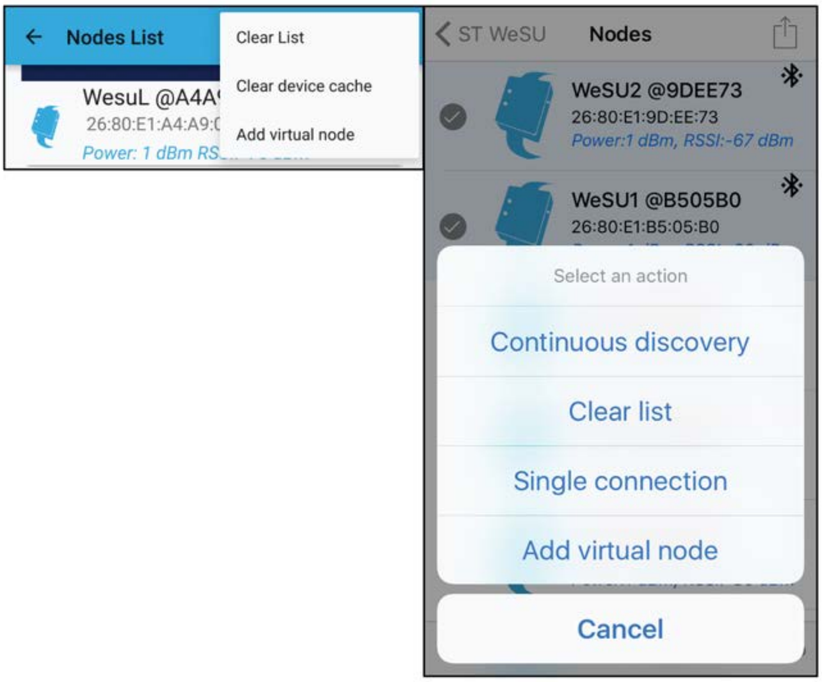
Figure 38. **Node status (Android and iOS) indication**



From the node list view, you can:

- restart scanning for nearby nodes from the search button or by swiping down the node list;
- switch from single multiple connection
- clear list to remove unconnected devices
- clear device cache (only Android) to clear Bluetooth device memory on specific node data saved at first-time node connection; this command forces the closure of all existing connections
- add virtual node is primarily for debugging purposes; it adds a virtual node and generates random data

Figure 39. **Nodes list (Android and iOS) menu**



In single node connection mode, the AHRS motion demo showing a simple spinning cubes starts as soon upon board selection, providing immediate STEVAL-WESU1 motion information.

Other demos are accessible via the demo selector button indicated below, or by scrolling the active page from right to left. The “Action” selector gives access to all available board actions and application settings.

Figure 40. **Demo index and Action menu (Android)**

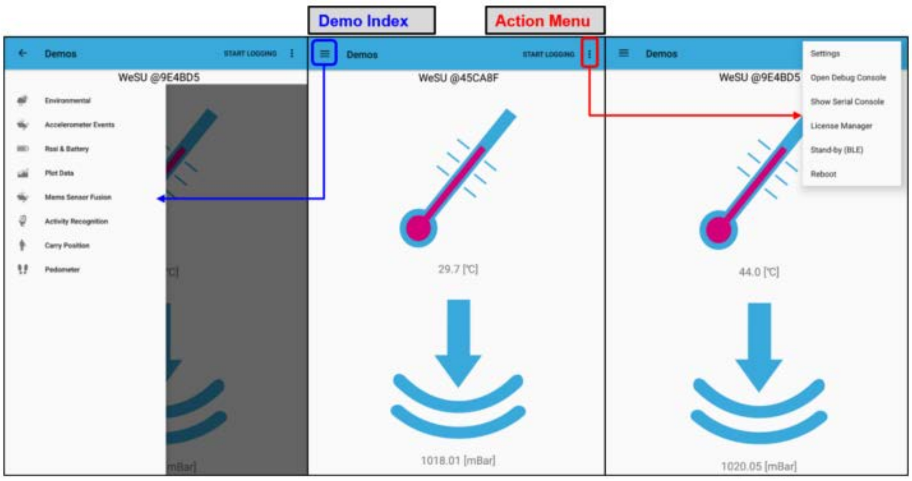
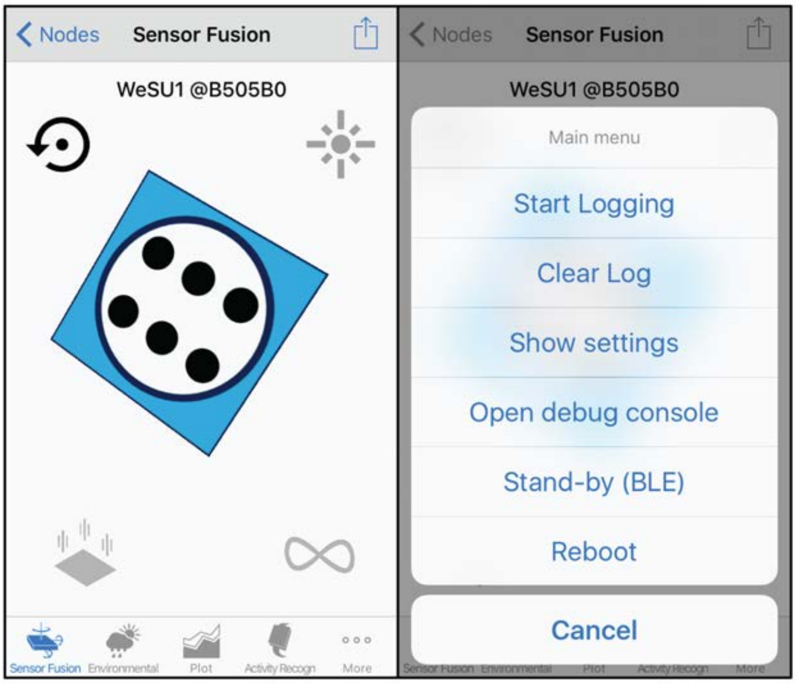


Figure 41. Demo index and Action menu (iOS)



4.1 Demo overview

From the first page, you can access the following:

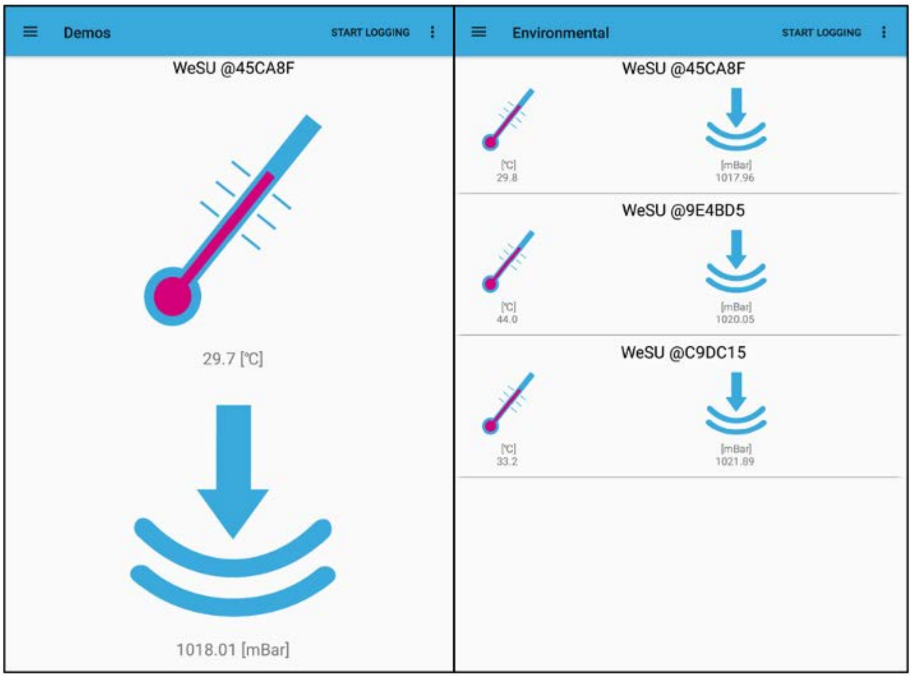
- **Environmental demo**
- **Accelerometer event demo**
  - Orientation 6D
  - Pedometer (HW)
  - Single/Double Tap
  - FreeFall
  - Wake Up
  - Tilt
- **RSSI and battery charging**:
  - RSSI value
  - TxPower
  - SOC battery in percentage
  - Battery status
  - Battery Voltage
  - Battery Current
- **Plot data demo:**
  - Can also configure plot length (time scale), and start/stop logging data
- **Algorithm demos**:
  - Sensor Fusion
  - Activity Recognition
  - Carry Position

The following sections relate to both Android and iOS versions, even if only Android screenshots are actually shown.

4.1.1 Environmental demo

On the next page (scrolling display right to left or use the demo selector button), you can run the environmental demo, with pressure and temperature feeds.

Figure 42. ST WeSU app (Android version) environmental demo

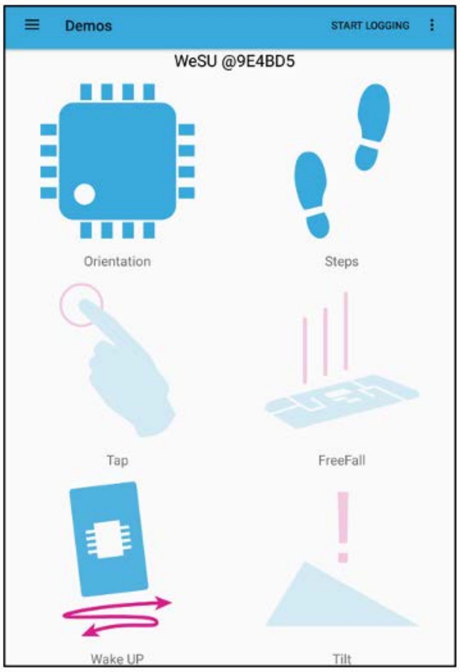


4.1.2 Accelerometer event demo

This demo shows the event coming from the on-board accelerometer (LSM6DS3), using the embedded features described below:

- **Orientation 6D** (always switched ON, with update on 6D device orientations)
- **Pedometer** (always switched ON, with update on stepcounter)
- **Single/Double Tap** (normally switched OFF, with update on event)
- **Free Fall** (normally switched OFF, with update on event)
- **Wake Up** (normally switched OFF, with update on event)
- **Tilt** (normally switched OFF, with update on event)

Figure 43. **ST WeSU app (Android version) Accelerometer demo**



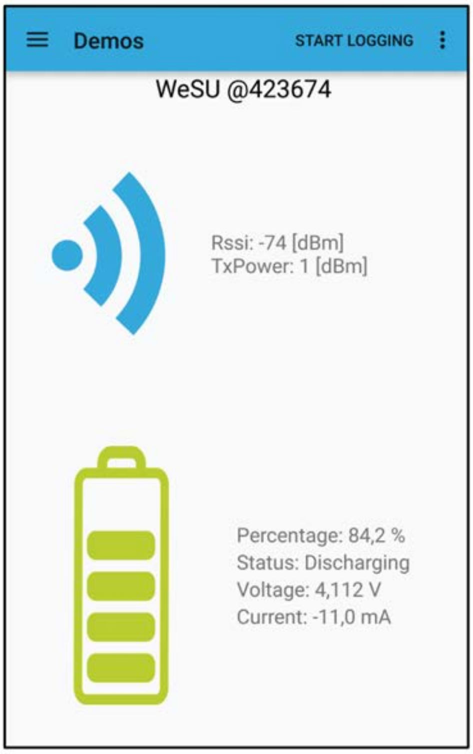
4.1.3 RSSI and battery

This simple demo shows the data directly from the on-board power management subsystem hardware. For the battery charging profile values, all the data comes directly from the STC3115 and STNS01 devices placed in the battery circuit path.

The values shown are:

- status of charge (SOC) in percentage
- battery status
- voltage
- current

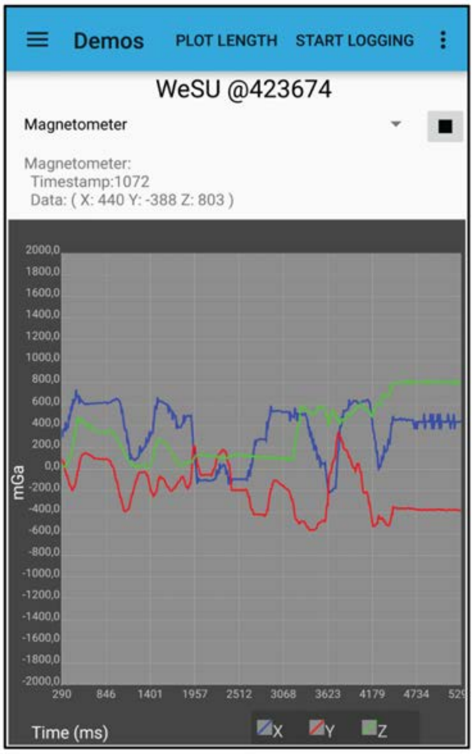
Figure 44. RSSI and battery charging demo



4.1.4 Plot data demo

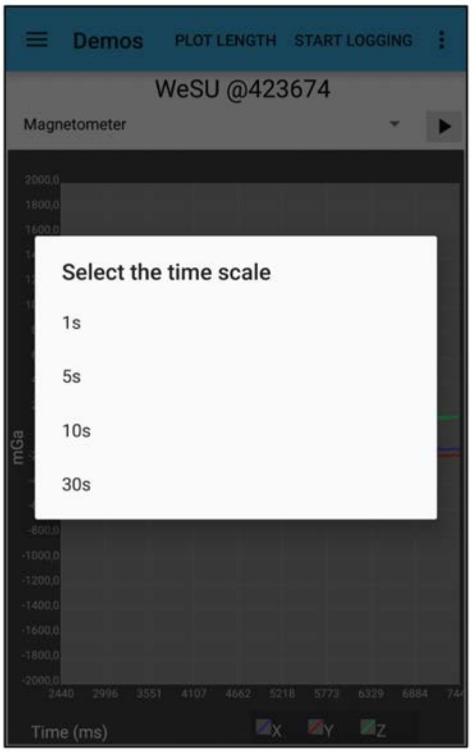
This demo includes the data plot from direct on-board sensor values, or values returned by algorithms run by the firmware.

Figure 45. ST WeSU app (Android version) example plot (magnetometer values)



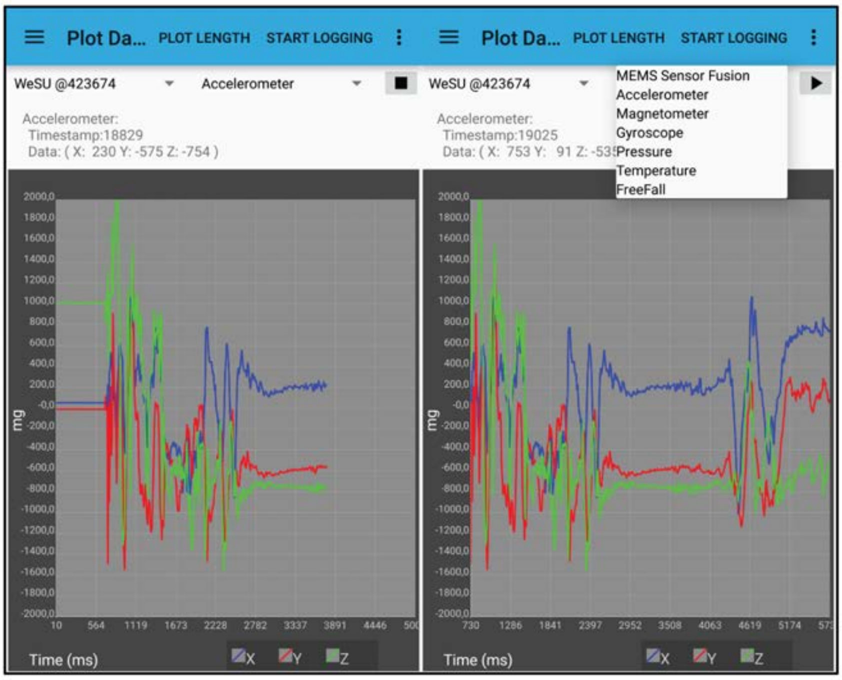
You can change the plot settings from the combo selection. From the Plot length menu, you can select the time frame from 1 s to 30 s, as shown below.

Figure 46. Plot length time scale selection



When multiple nodes are connected, it is also possible to select the device and feature to plot.

Figure 47. Features data plot selection

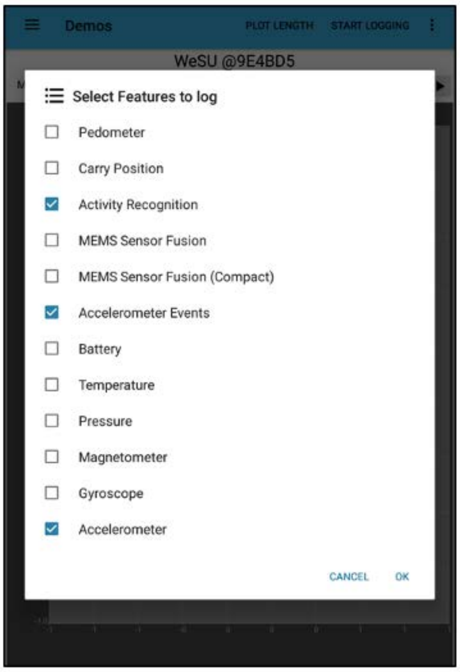


4.1.4.1 *Start/stop logging*

All the demos views have the start/stop logging command, which saves raw demo data in the .csv format. Data is stored for each connected device in files that contain logs for each feature.

The START LOGGING command lets you select any measurement to store, as shown in the next figure.

Figure 48. ST WeSU app (Android version) LOG setting menu



For example, logging environmental data results in yyyymmdd_HHmm_Temperature.csv and yyyymmdd_HHmm_Pressure.csv files, where the prefix represents the date and time when the log starts.

The file includes:

- header information with date time of log start, devices connected, the logged feature, data header and units
- the raw data with the logged host date time stamp (relative to start time), node name and feature timestamp.

Stopping the log triggers the prompt to send all available logged data by mail.

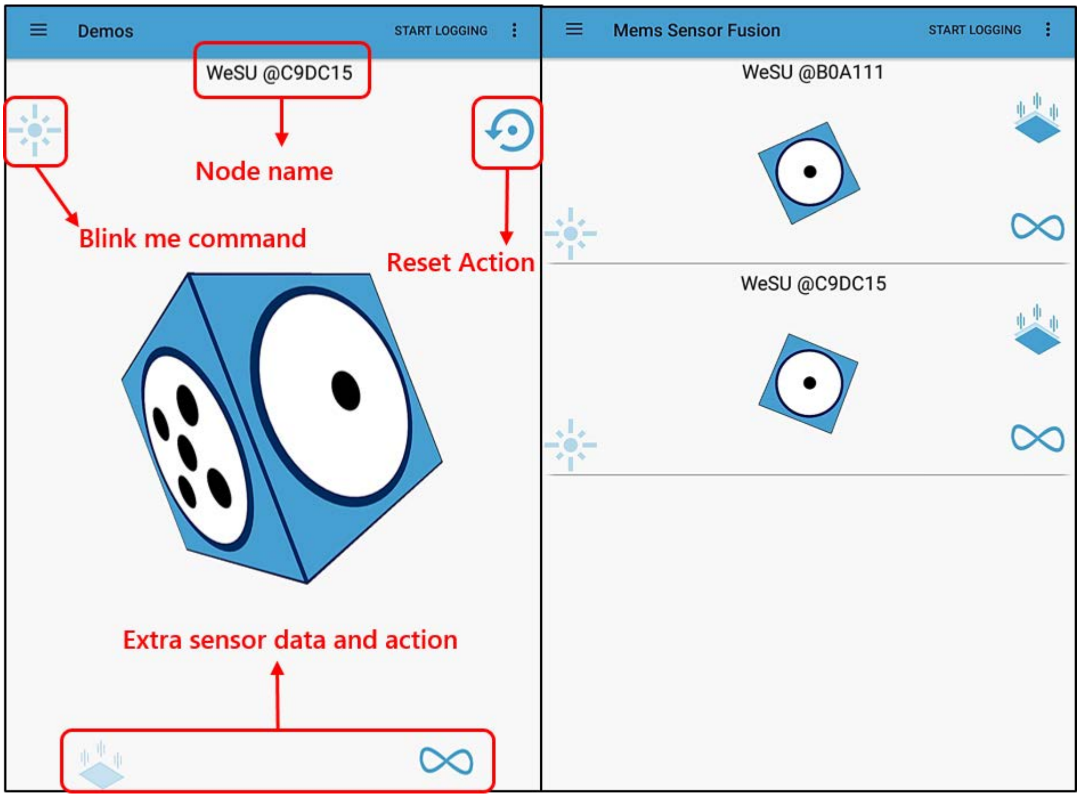
4.1.5 Algorithm demos

Apart from the sensor fusion demo already discussed, the following motion-related demos are also available.

4.1.5.1 *MotionFX suite*

The MEMS sensor fusion demo is automatically launched after board selection.

Figure 49. **ST WeSU app (Android version) motion sensor fusion demo**



It directly feeds all the available information associated with the board motion features and actions.

Node name indicates which node is running, the start/stop logging command is discussed below, and the reset command aligns the cube position

The extra sensor data and action area shows the extra available features and commands:

- Freefall (if available) shows the freefall status:
  - no freefall
  - currently in freefall
  - Saved freefalls – if shown, it displays the list of the last 10 freefalls when clicked; from here you can choose to either continue adding freefall events or clearing the list entirely.

Figure 50. **Freefall icons**

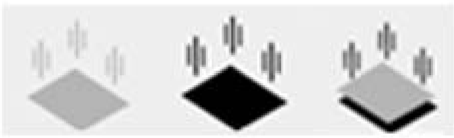
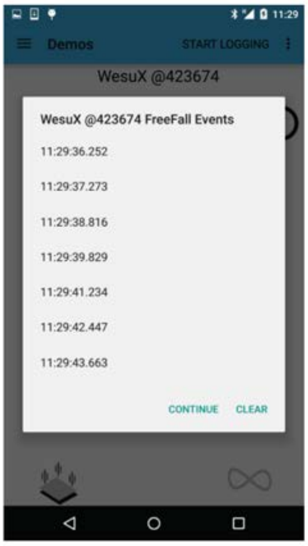
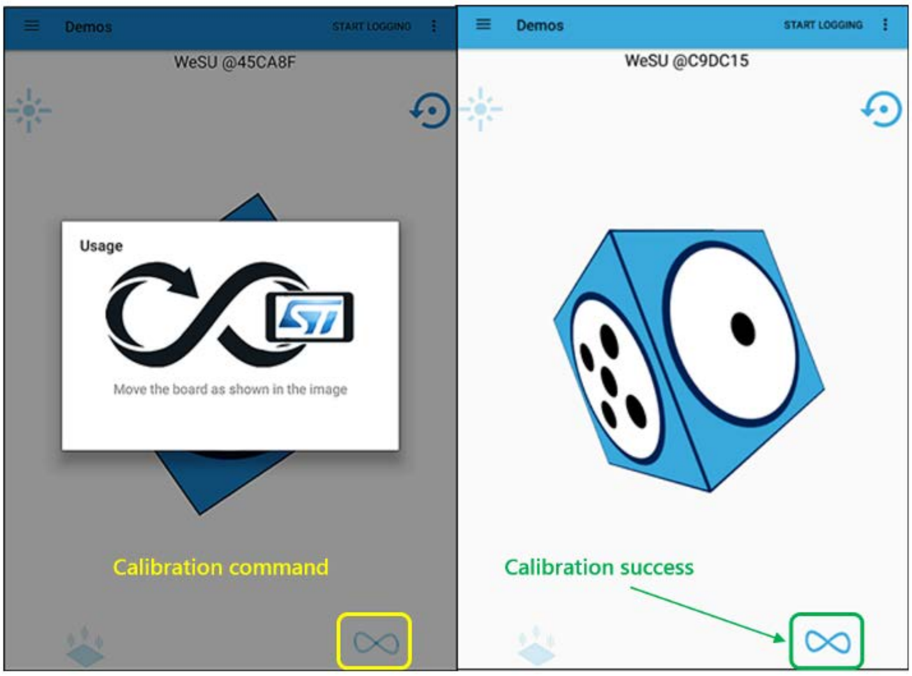


Figure 51. **Freefall events list**



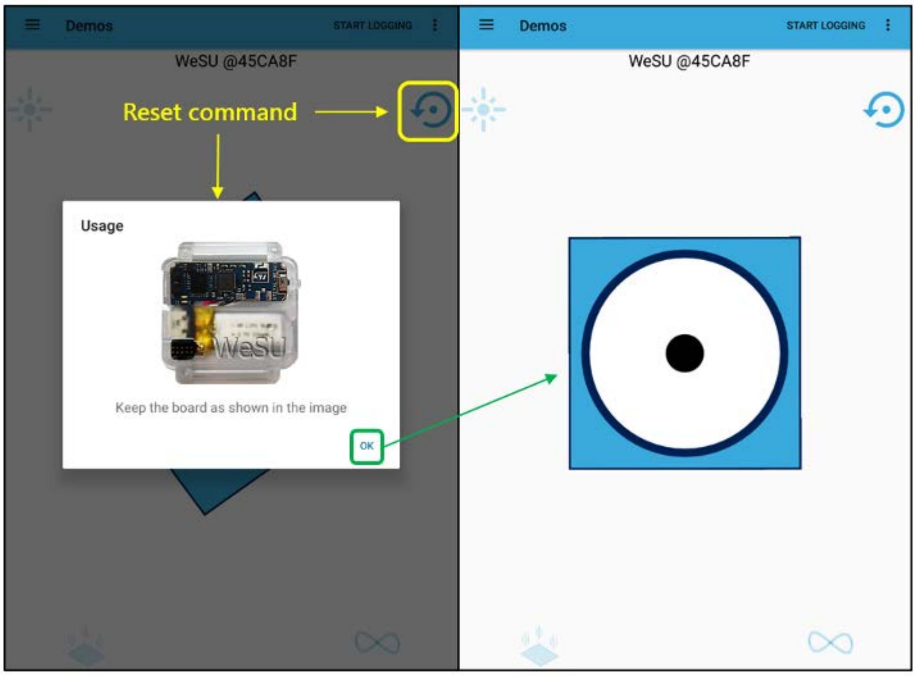
- MEMS sensor fusion calibration (if available). The calibration status of the sensor fusion library is normally gray before calibration. When pushed, a popup window instructs you how to move the board to facilitate calibration; the symbol turns black when calibration has been completed successfully.
  – calibration available (calibration status unknown)
  – node sensor fusion calibrated

Figure 52. **ST WeSU (Android) calibration**



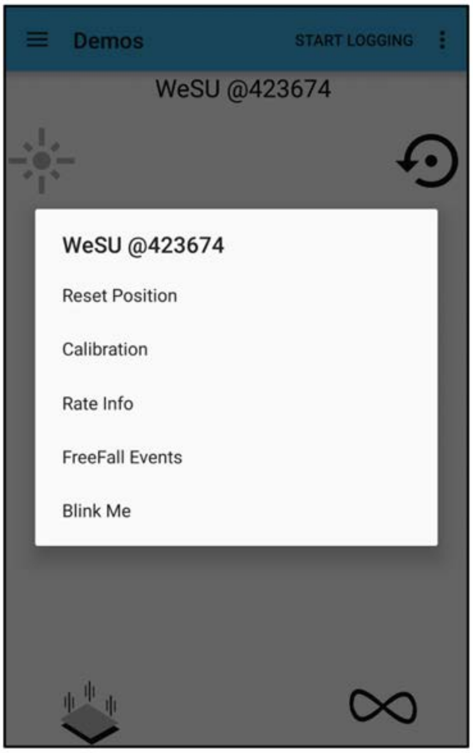
- The reset action button allows resetting the cube position; when pushed, a popup window explains how to physically move the board to the default position. Reset is useful after calibration.

Figure 53. **ST WeSU (Android) reset position**



All demo commands are also available by tapping the demo area with a context menu, as shown below.

Figure 54. **Commands context menu**



4.1.5.2 The MotionAR suite

The MotionAR suite features predictive software which recognizes common activities and states, like:

- stationary

- walking
- fast walking
- jogging
- biking
- driving

Real-time activity recognition can significantly improve the user experience in advanced motion-based applications for consumer, computer, industrial and medical purposes.

The algorithm exclusively manages the data acquired from the accelerometer at a low sampling frequency (16 Hz) to minimize power consumption.

Figure 55. Activity recognition demo

The MotionAR engine is provided as a node-locked library which allows derivative firmware images to run on a specific STM32-based board only.

Licensing activation code requests must be forwarded to ST and these codes must then be included in the project prior to usage - failure to do so will prevent proper API execution.

The resulting firmware binary image will therefore be node-locked.

4.1.5.3 *The MotionCP suite*

This predictive software recognizes the carry position of the board using exclusively the data acquired from the accelerometer; carry positions include:

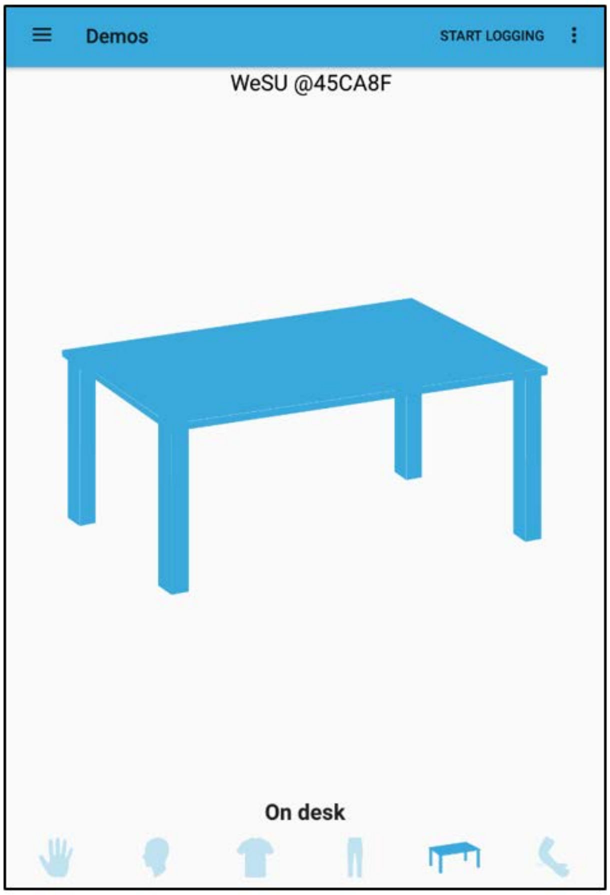
- on desk
- in hand
- near head
- shirt pocket

- trouser pocket
- arm swinging

The real-time carry position algorithm can significantly improve motion-based applications in the consumer, computer, industrial and medical fields.

The algorithm exclusively manages the data acquired from the accelerometer at a low sampling frequency (50 Hz) to minimize power consumption.

Figure 56. **Carry position demo**



The MotionCP engine is provided as a node-locked library which allows derivative firmware images to run on a specific STM32-based board only.

Licensing activation codes must be requested from ST and included in the project prior to attempting its usage - failure to do so will prevent proper API execution.

The resulting firmware binary image will therefore be node-locked.

4.2 Action list overview

From the action selector button at the top right of the screen, you can access:

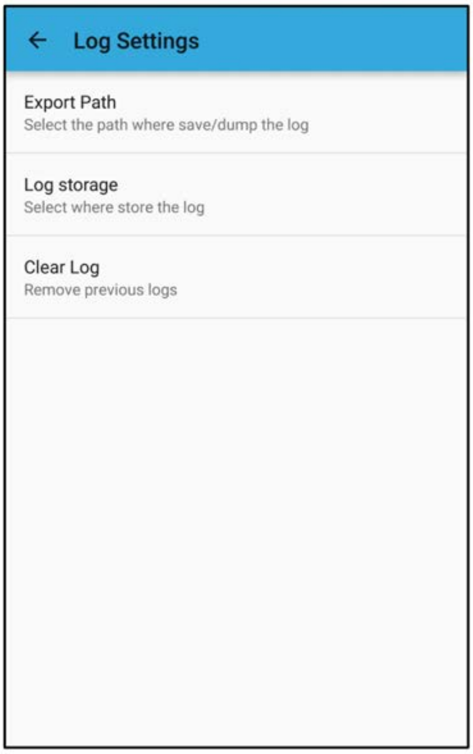
- Start/stop logging
- Settings
- Debug console
- Serial console
- BLE standby functions
- Reboot functions

4.2.1 Settings

This menu provides the following settings groups:

- **General Settings:** to enable free fall signaling and node firmware checking
- **Log Settings:** to set your export path and determine how logged data is treated:
  - saved directly in .csv files
  - saved in an sql db and exported to .csv once logging has finished in order to send the data by mail
  - not saved, but sent to the logCat (Android Studio debug view)
  - Clear log: lets you select specific session logs and remove them
  - Export log: lets you select specific session logs and export them

Figure 57. **Log settings**

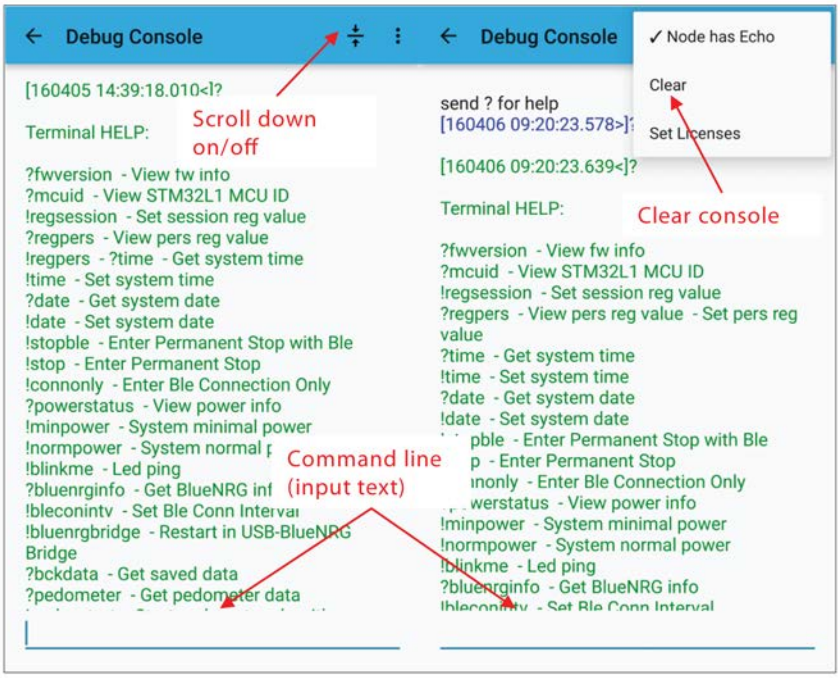


- **Node Configuration:** is divided into the following subgroups:
  1. Node general settings: for Name, Address, FW Version and BlueNRG.
  2. Session settings: configure the firmware session registers to control high level sensor and algorithm features, and low power mode.
  3. Persistent settings: configure the firmware persistent registers to control the same features as the session settings, but with default values. It also contains other low level sensor parameters like full scale (FS), output data rate (ODR) and BLE output power.
  4. System settings: includes RTC configuration, device firmware upgrade (DFU) selector and six different Power-OFF modes (See Section 4.2.5 Reboot).

4.2.2 Debug console

The Debug console can be opened from the actions menu to allow management of several board features and command functions (sensor selection, read frequency, BlueNRG communication subsampling) through a command line interface, and also to read register values.

Figure 58. Debug console command list



When the Debug console is open, all the functions implemented in the firmware can be manipulated through a simple command line interface:

- To get information, status and internal register in reading: `?command_name`
- To set a value, an internal register or other in writing mode: `!command_name`

4.2.3 Serial console

This is a terminal window for debugging, where you can read messages from the firmware and control critical conditions.

4.2.4 BLE standby

This is a command to force the Bluetooth standby condition, causing demo interruption and forcing the entire STEVAL-WESU1 system into STOP mode.

4.2.5 Reboot

Force a system reboot to guarantee control in case of problems with communications or sensor data, or low power modes.

4.2.5.1 *Low power mode selection*

Selecting Power Mode from the node configuration settings lets you reduce the system current consumption, keeping some features available if needed.

This function is available for a current session via the Session Settings or for all running sessions via the Persistent Settings.

These functions differ from the Power-OFF features available in System Settings.

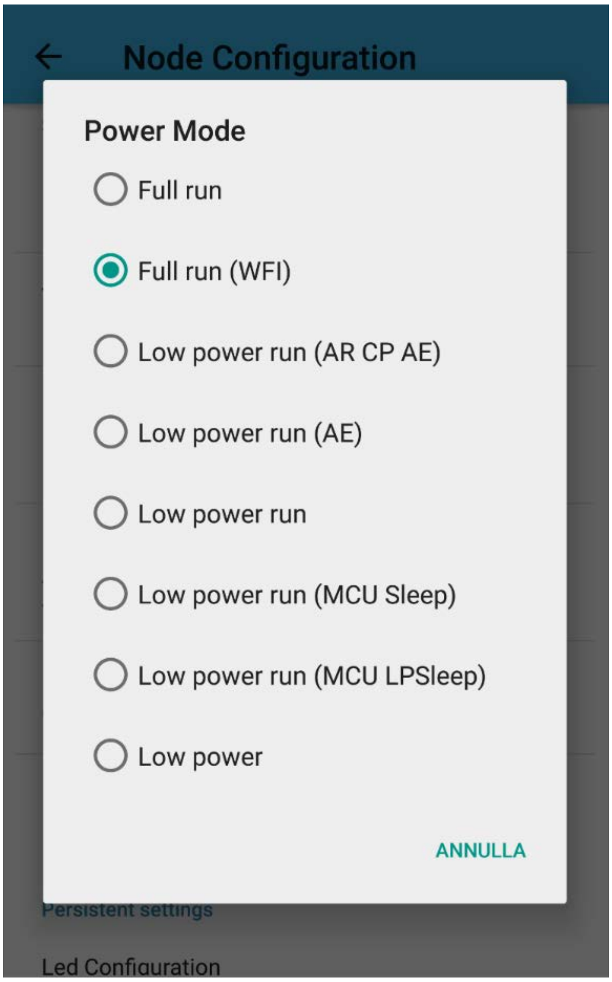
The available running low power modes are:

- **Full run**: high level performance mode, without any constraints associated with clock frequency, device available, BLE connection, and microcontroller features.
- **Low power run (WFI)**: standard power mode selected when the node starts. It uses the same configurations and functions as Full run mode, but forces the microcontroller to perform a WFI instruction every main cycle; all the features, algorithms and high level functions are retained and the system is totally configurable.
- **Low power run (AR CP AE)**: like low power run (WFI), but with the accelerometer enabled and the other sensors left in low power mode. The frequency is adjusted to suit the algorithms that use the accelerometer

data (AR and CP) only or its embedded hardware features (AE) such as Pedometer, FreeFall, or 6D orientations using the minimum ODR value.

- **Low power run (AE)**: the same as low power run (AR CP AE) without the SW algorithms. The system frequency is reduced and the MCU clock is 4 MHz.
- **Low power run (MCU Sleep)**: only the accelerometer and the gas gauge are kept active, the system frequency is reduced (1 Hz) and the MCU clock is 4 MHz, with the core forced into SLEEP mode.
- **Low power run (MCU LPSleep)**: like low power run (MCU Sleep), but with the core forced into LPSLEEP mode.
- **Low power**: system frequency is 25 Hz, MCU clock is 4 MHz, and all the sensors are set to low power mode.

Figure 59. Set running low power modes



5 Factory settings restore and firmware upgrade

The firmware can be updated using the IDE toolchain projects available in the package; however, this procedure is only really useful to verify the code in debug mode. If this is not required, the user can just download the available binary files to the FLASH memory and proceed to test the application.

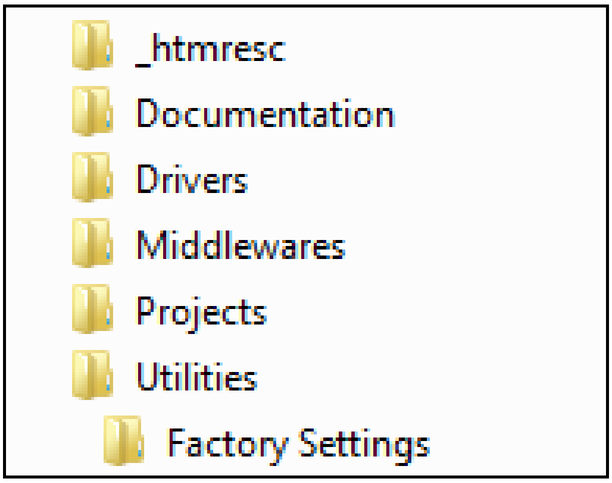
5.1 Restoring the factory settings with ST-Link

To use the binary firmware code, there are files dedicated to restoring functionality, including the default settings; however, to update it directly, you must:

- download and install "STM32 ST-LINK Utility" from www.st.com and run it
- click “Target” -> “Settings” and select SWD Connection
- Click “File” -> “Open” and select the binary file to be programmed
- Click “Target” -> “Program and verify” and then “Start”
- Wait for the “Verification… OK” message in the log window and disconnect the programming cable

The binary files are available in the “Utilities/Factory Settings” folder.

Figure 60. Factory settings folder



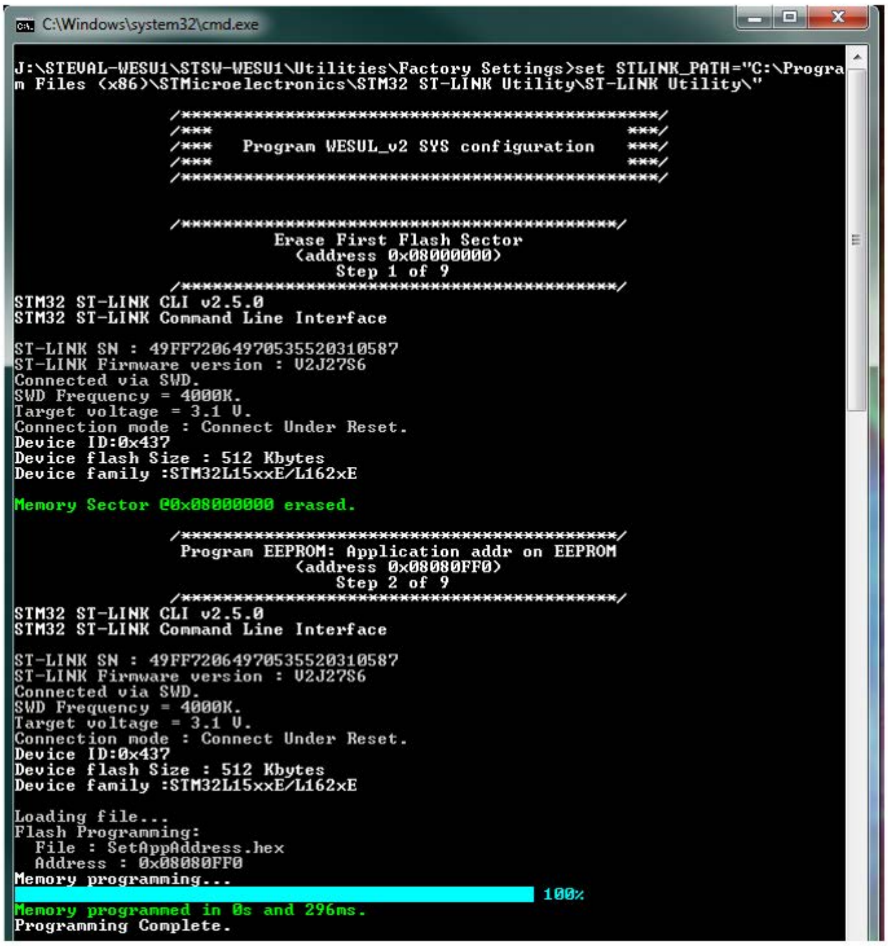
There are different batch files in the same folder that allow you to work with the ST-LINK programmer, without using the GUI.

The readme.txt file inside the folder will help you choose the appropriate batch file for the desired activity. The main ones are:

1. **STEVAL-WESU1_FACTORY.bat:** restore all the memory content to factory conditions.
2. **ProgApplication.bat:** update the DEMO application code only.
3. **ProgApplicationAndOTA.bat:** update the DEMO application code plus the OTA manager.
4. **ProgExamples.bat:** update the EXAMPLE application code only.

When launching one of these batch files, a DOS command window opens with a list of programming steps, as shown below.

Figure 61. Batch file execution in DOS command window



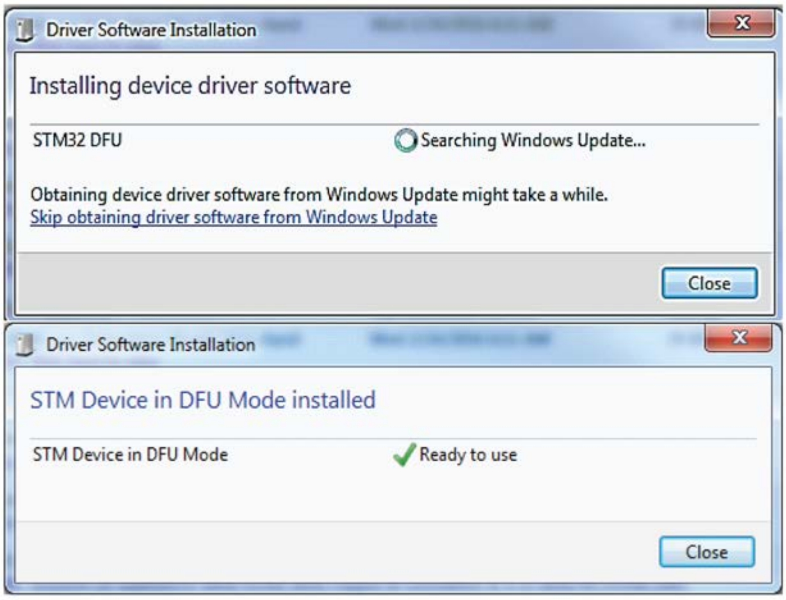
5.2 Device firmware upgrade using USB

Before connecting the USB cable, download and install the DfuSe demonstration software from www.st.com.

Following installation, set the board to DFU mode thus:

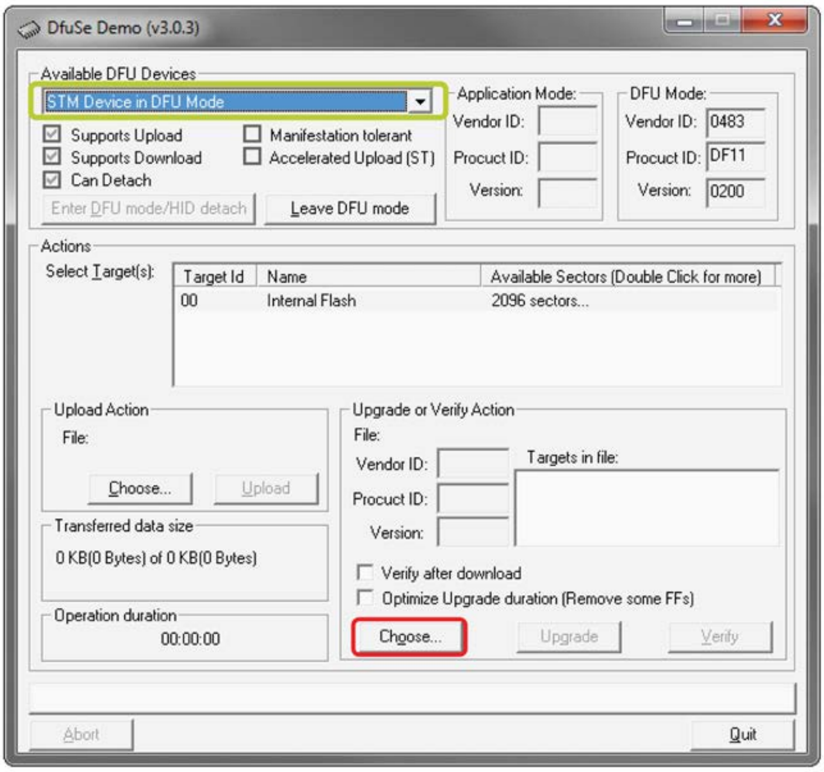
- Plug the USB cable, the board starts in normal mode running the demo application, and the LED should blink every two seconds (at 0.5Hz).
- Set the board to standby mode by pressing the user button. Wait until the LED goes off.
- With the USB cable still plugged, press and hold the user button for at least 3 seconds, until the LED turns on. The board then enters DFU Mode and the LED blinks at 2.5 Hz. You can also use the ‘settings’ menu in the ST WeSU app (Settings -> Node Configuration -> Device Firmware Upgrade -> USB DFU).
- With the USB connection still active, you will see the following message:

Figure 62. **DFU Driver installation**



- Now run the "DfuSe Demonstration" software, see "STM Device in DFU Mode" in the "Available DFU Devices" drop down list.

Figure 63. **DfuSe Demo**



The user can upgrade the application firmware directly by choosing the *.dfu file.

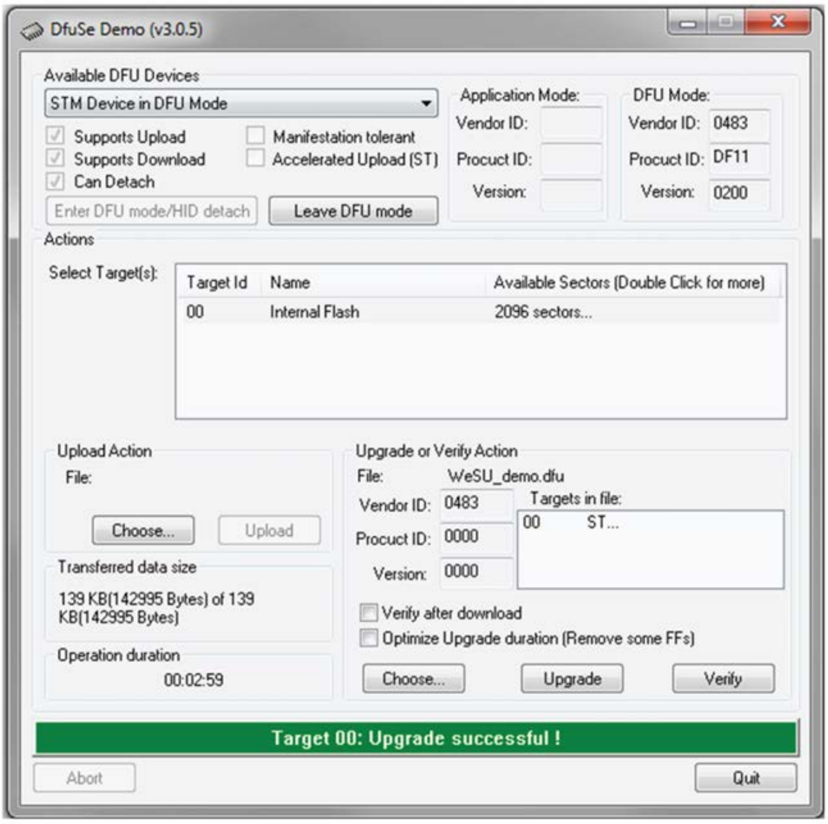
- Click on "Choose" button and select the file to update the desired firmware:

Table 6. **Package file list**

DFU Files	Action
WeSU_demo.dfu WeSU_examples.dfu	Program Flash DEMO or EXAMPLES APPLICATION (@0x08020000)
WeSUandOTA.dfu	Program Flash APPLICATION (@0x08020000) + OTA (@0x08003800)
WeSU_OTA_ServiceManager_App.dfu	Program Flash OTA SERVICE MANAGER (address 0x08003800)
SetAppAddress.dfu	Set Application address on EEPROM location (@0x08080FF0)
WESU_BlueNRG_VCOM_1_8.dfu	Program Usb BlueNRG-MS Bridge (@0x08008000)
SetBluenrgUsbBridgeAddress.dfu	Program EEPROM: BlueNRG Bridge address on EEPROM
ResetManager.dfu	Program Flash RESET MANAGER (address 0x08003000)
ResetRegs.dfu	Program EEPROM Delete Registers

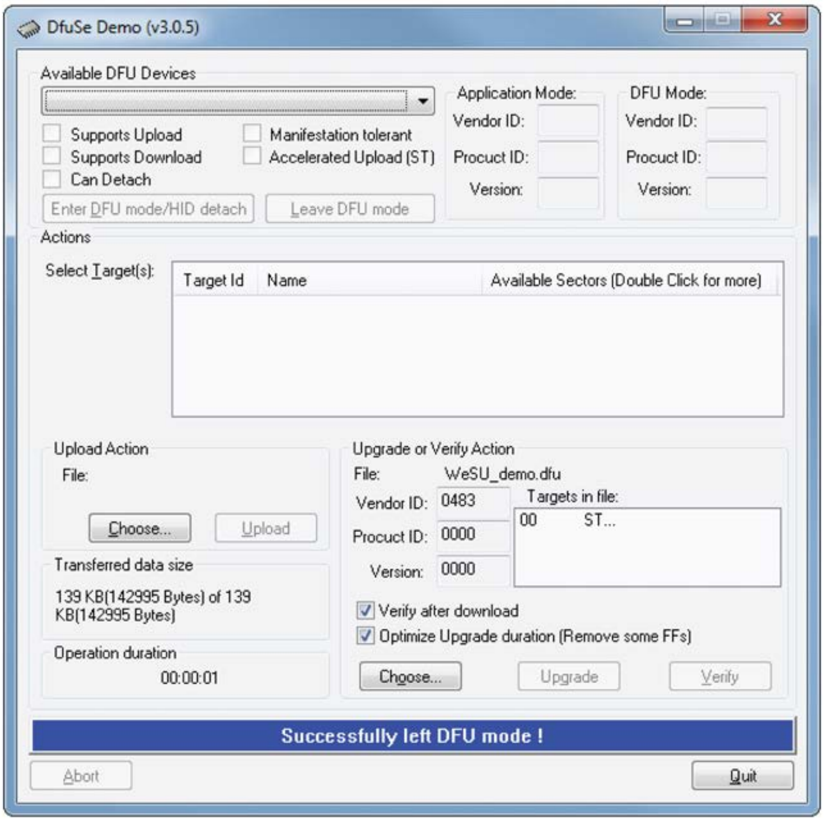
- Once the dfu file is selected click on “Upgrade” and wait for the completion confirmation

Figure 64. **DfuSe Demo upgrade successful**



- When finished, click “Leave DFU mode”; the board restarts and runs the loaded firmware.

Figure 65. DfuSe Demo leave DFU mode



5.3 Device firmware upgrade OTA via BLE

Before performing the OTA update, download and install the dedicated app on the mobile devices used: **ST BlueDFU**, available on Google Play (soon to be available on Apple Store).

Figure 66. **ST BlueDFU start page**

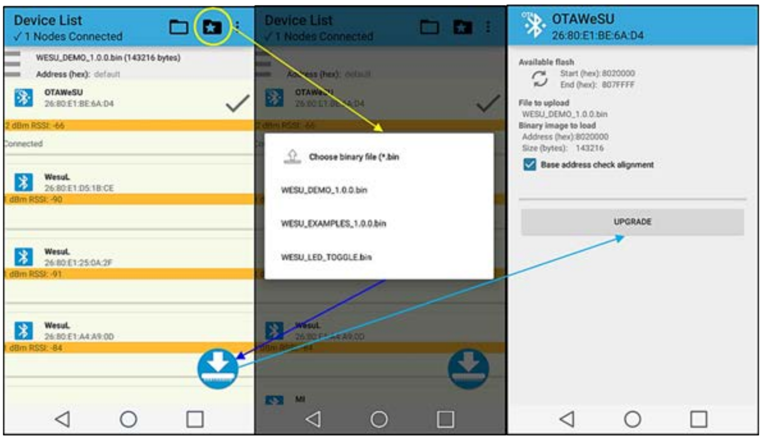
Following installation, set the board in DFU mode thus:

- Push the user button to start the board in normal mode running the demo application; the LED blinks every two seconds (at 0.5 Hz).
- Set the board in standby mode by pressing the user button. Wait until the LED goes off.
- Without the USB cable plugged, press and hold the user button for at least for 3 seconds until the LED turns on. The board then enters OTA mode and the LED toggles at 1 Hz.

Note: If you want to exit OTA mode without updating the firmware, press the user button three times (each time the LED performs a soft blink); the board will restart in normal mode running the demo application.

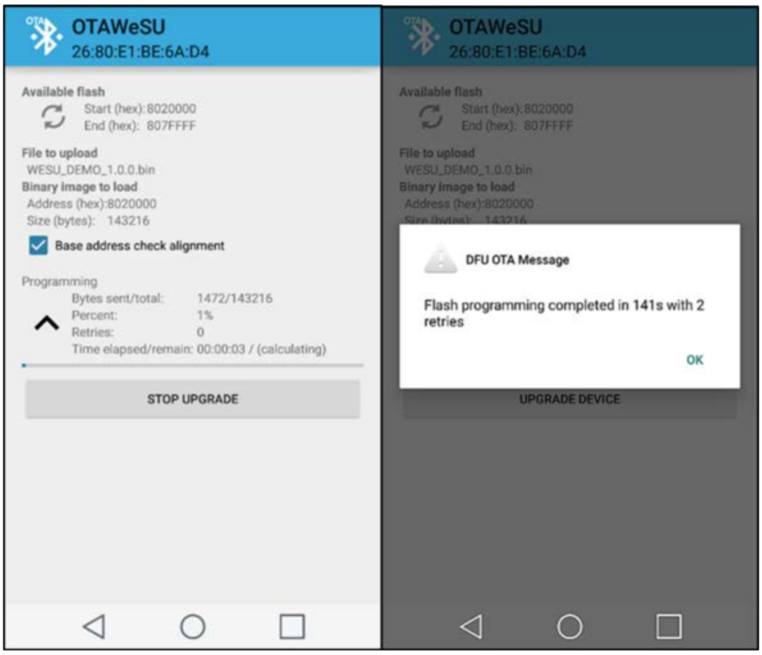
- When the board is in DFU OTA mode, run the **ST BlueDFU** application on your mobile device and check the device list.
- The board in OTA is recognizable by the name OTAWeSU.
- Select your board (the LED toggles faster at 4 Hz); the BLE connection is ready to download the firmware binary file. If there are different boards in OTA, you can check the address or simply ‘try’ the connection and verify which one accelerates the LED toggle frequency
- Select the *.bin file from a dedicated folder in the mobile device or use the default file already available in the folder with the star.
- Click the download icon, and double check the address values, filename and size of firmware code.

Figure 67. **ST BlueDFU working flow (Android Version)**



- Run the update and wait until it finishes after more or less 2.5 minutes; after which, the board automatically restarts with the newly loaded firmware.

Figure 68. **ST BlueDFU device update**



It is possible to upgrade more than one device simultaneously with the same binary firmware selected. Ensure you select the same device type (hardware configuration) and download the right firmware to avoid rendering the device unusable.

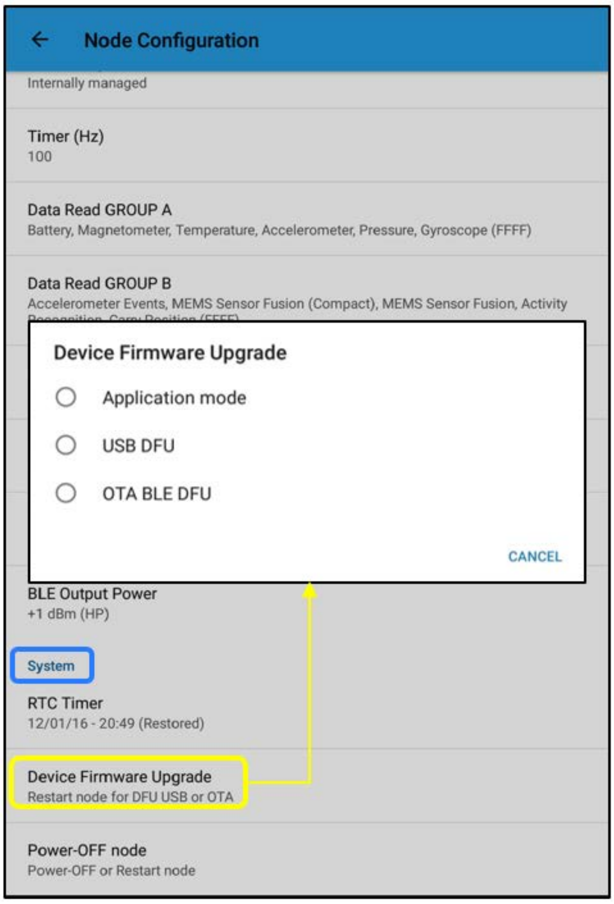
5.4 Entering DFU mode directly from the App

A custom function to enter DFU mode directly inside the mobile App is also available for boards connected in application mode. In this mode, with the board connected and visible by App:

1. open the action menu (top right)
2. Select Settings > Node Configuration
3. Select Device Firmware Upgrade in the System subgroup
4. Select the DFU mode:
   – Application mode - to confirm the current mode (only useful for debugging)
   – USB DFU - to restart the board directly in USB DFU mode

– OTA BLE DFU - to restart the board directly in OTA BLE DFU mode

Figure 69. DFU selection from App settings



6 BlueST Protocol SDK

The Bluetooth low energy interface protocol is used by the BlueST app (iOS and Android) with the STEVAL-WESU1 reference design. The sensor board exposes its functions and communicates with the host through structured services and characteristics.

Each data characteristic can be signaled with different timings (setting the respective register and therefore subsampling the application timer) and can be read asynchronously.

6.1 Advertising format

According to the Bluetooth 4.0 core specification Vol.3 part C, the 0xFF identifies vendor-specific information.

6.1.1 Bluetooth low energy AD structure (max. 31 bytes)

Table 7. BLE advertising structure

AD Field Name	ADType	AD Length	Record size
TX_POWER_LEVEL	0x0A	2	3
COMPLETE_NAME	0x09	Max. 10/16 (1)	11/17(1)
MANUF_SPECIFIC	0xFF	13/7(1)	14/8(1)
FLAGS	0x01	2	3

1. *If the public device address is not set in the MANUF_SPECIFIC field, then the COMPLETE_NAME can be maximum 16 bytes*

6.1.2 AD structures

6.1.2.1 *TX_POWER_LEVEL advertising item*

Table 8. BLE TX power level advertising field

Octets LSB	0	1	2
Name	Len	Type	-100 a + 20 dBm
Value	0x02	0x0A	0xXX

6.1.2.2 *MANUF_SPECIFIC advertising item*

Table 9. BLE advertising manuf.-specific advertising field

<table>
<tr><th>Octets LSB</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th></tr>
<tr><td>Name</td><td>Len</td><td>Type</td><td>Ver</td><td>Dev ID</td><td colspan="2">Group A Features</td><td colspan="2">Group B Features</td><td colspan="3">Company assigned</td><td colspan="3">Company id (proposal)</td></tr>
<tr><td>Value</td><td>0x0D</td><td>0xFF</td><td>0x01</td><td>0xXX</td><td colspan="2">0xXXXX</td><td colspan="2">0xXXXX</td><td colspan="3">0xXXXXXX</td><td colspan="3">0x2680E1</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="6">Public device address (48 bits) (optional)</td></tr>
</table>

Table 10. Group A features map

N	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU	ACC	GYRO	MAG	PRESS	RFU	TEMP	BATT	RFU

Table 11. Group B features map

N	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	RFU	RFU	RFU	RFU	RFU	RFU	Free Fall	RFU	Sensor Fusion	RFU	RFU	Activity Recognition	Carry Position	RFU	RFU	Pedometer

6.1.2.3 *Device ID enum*

Table 12. Device ID enumeration

ID	HW
0x00	Generic
0x01	WeSU
0x02 – 0x7F	RFU
0x80 – 0xFF	Nucleo Map

Note: If GPA or GPB bits are set, then the general purpose characteristics have to be defined

Note: The msb in the Device ID enum is used to indicate a Nucleo-based system

6.2 Services/characteristics

Table 13. Services/characteristics allocation map

Groups	Service	Char	Max Size	Mode	UUID	Note
Data	Features				0x0000 0000 0001	
		Group A 16 single features	n/a	r/n	0xXXXX 0000 0001	TS + Value, 0xXXXX only one bit, (e.g. Accelerometer bit 7 => 0x00800001-)
		Group B 16 single features	n/a	r/n	0x0000 XXXX 0001	TS + Value, Only one bit
	General Purpose				0x0000 0000 0003	must use the characteristic descriptor to configure the data Unit, Name, Type (Size), Format (Precision)
		GP XXXX XXXX	n/a	r/n	0xXXXXX XXXX 0003	TS + Value[s]
Debug	Debug				0x0000 0000 000E	
		Term	n/a	r/w/n	0x0000 0001 000E	This characteristic is used for debug purpose as a hyper-terminal like connection (Stdio)
		StdErr	n/a	r/n	0x0000 0002 000E	This characteristic is used for debug purpose as an output only terminal in order for the BLE device to send textual info on errors (StdErr)

Groups	Service	Char	Max Size	Mode	UUID	Note
Config	Control		64		0x0000 0000 000F	
		Registers access			0x0000 0001 000F	
		Feature Command	64	w/n	0x0000 0002 000F	

Note:
1. BLUETOOTH SPECIFICATION Version 4.2 [Vol 3, Part B] 2.5.1 UUID
2. Service UUID -11e1-9ab4-0002a5d5c51b
3. Char UUID -11e1-ac36-0002a5d5c51b
4. All data char contains TS [2 bytes] (first field)
5. TS is timestamp (uint16) relative to the board and valid for all features

6.2.1 HW single feature packet

All packets start with a uint16 timestamp (TS)

6.2.1.1 *Generic packet format*

Table 14. **Generic packet format**

Octets LSB	0	1	2	3	…	N
Name	TS		Payload: Value[s]			
Value	0xXXXX		0xXXXXXXXX			

6.2.1.2 *Motion sensors packet format*

Accelerometer payload: mg, signed int16
Gyroscope payload: dps, signed int16
Magnetometer payload: mGa, signed int16

Table 15. **Motion sensors packet format**

Octets LSB	0	1	2	3	4	5
Name	X		Y		Z	
Value	0xXXXX		0xXXXX		0xXXXX	

6.2.1.3 *Battery packet format*

Battery payload:
- **Battery level**: 0.1%, signed int16 (multiply by 10)
- **Battery Voltage**: mV, signed int16
- **Average current**: mA, signed int16
- **Power mgmt. Status**: enum, unsigned int8

Table 16. **Battery packet format**

Octets LSB	0	1	2	3	4	5	6
Name	Battery level		Battery Voltage		Average current		Power Mng Status
Value	0xXXXX		0xXXXX		0xXXXX		0xXX

Table 17. **Pressure packet format**

Octets LSB	0	1	2	3
Name	Pressure Value			
Value	0xXXXXXXXX			

Pressure payload: mbar, signed int32, multiply by 100

6.2.1.4 *Temperature packet format*

Temperature payload: Celsius multiplied by 10

Table 18. **Temperature packet format**

Octets LSB	0	1
Name	Value	
Value	0xXXXX	

6.2.2 Aggregate feature packet

To optimize the data throughput, you can aggregate multiple features in a single characteristic. The resulting UUID is the OR of the UUID of the single features.

The features data must follow the feature mask order.

Example: Motion packet (0x00700000) = Accelerometer (0x00400000) + Gyroscope (0x00200000) + Magnetometer (0x00100000)

Table 19. **Motion packet format**

Octets LSB	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Name	TS		Accelerometer						Gyroscope						Magnetometer					
Axis	-		X		Y		Z		X		Y		Z		X		Y		Z	
Value	0xXXXX		0xXXXX		0xXXXX		0xXXXX		0xXXXX		0xXXXX		0xXXXX		0xXXXX		0xXXXX		0xXXXX	

Accelerometer payload: mg, signed int16
Gyroscope payload: tenth of dps, signed int16
Magnetometer payload: mGa, signed int16

6.2.3 SW single feature packet 1/2

All packets start with a uint16 timestamp (TS)

6.2.3.1 *Generic packet format*

Table 20. Generic packet format

Octets LSB	0	1	2	3	...	N
Name	TS		Payload:Value[s]			
Value	0xXXXX		0xXXXXXXXX			

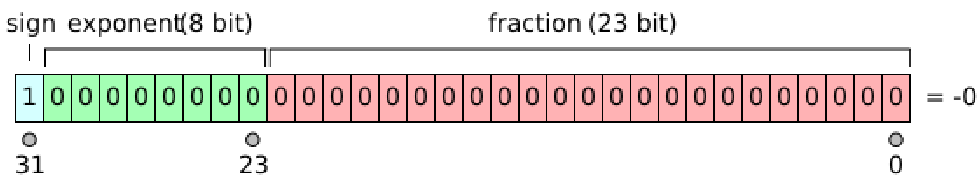
6.2.3.2 *Sensor fusion payload: 4x float (IEEE 754 single)*

Table 21. Sensor fusion packet format

Octets LSB	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Name	TS		qi				qj				qk				qs			
Value	0xXXXX		0xXXXXXXXX				0xXXXXXXXX				0xXXXXXXXX				0xXXXXXXXX			
Field	-		Vector coefficients												Scalar coefficient (optional)			

If it only has 3 fields (qi,qj,qk), the vector coefficients are normalized
IEEE 754 single (Reference: AN4044 on www.st.com)
(Using floating-point unit (FPU) with STM32F405/07xx and STM32F415/417xx microcontrollers)

Figure 70. IEEE 754 single reference



6.2.3.3 *Free fall payload*

Free fall payload: 1 byte is needed if used in an aggregated fashion

- 1 = free fall event
- 0 = no free fall event

Table 22. Free fall packet format

Octets LSB	0
Name	Value
Value	0 or 1

6.2.3.4 *Activity recognition payload*

Activity recognition payload: 1 byte

Table 23. Activity recognition packet format

Octets LSB	0
Name	Value
Value	0 …. 6

Table 24. Valid activity types

Activity type	
0x00	No activity (no enough data for decide)
0x01	Stationary
0x02	Walking
0x03	Fast walking
0x04	jogging
0x05	Biking
0x06	driving

6.2.3.5 *Carry position payload*

Carry position payload: 1 byte

Table 25. Carry position packet format

Octets LSB	0
Name	Value
Value	0 …. 6

Table 26. Carry position type

Carry position type	
0x00	Unknown
0x01	On Desk
0x02	In Hand
0x03	Near Head
0x04	Shirt Pocket
0x05	Trouser Pocket
0x06	Arm Swing

6.2.4 Configuration settings

6.2.4.1 *Register access*

Table 27. Register access packet format

Octets LSB	0	1	2	3	4	5	…	64
Name	CTRL	ADDR	ERR	LEN	Payload			

- CTRL field

Table 28. **Control field**

N	7	6	5	4	3	2	1	0
Mode	Pending	Mode	Type	Error	Ack	RFU	RFU	RFU
1	Exec op	Persistent	Write	Error	Ack required	RFU	RFU	RFU
0	No op	Session	Read	No error	No ack	RFU	RFU	RFU

- ADDR: register address (0x00 – 0xFF)
- ERR: error code (0x00, no ERROR – 0x01-0xFF specific error code)
- LEN: register number (len * 2 = payload size in byte)

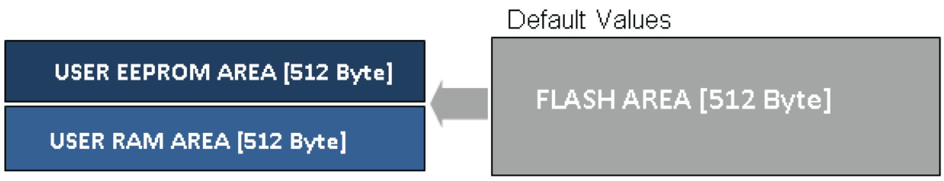
6.2.5 Debug

- The package doesn't contain a timestap
- If the package doesn't end in '\0', the message finishes in the next package
- Max package size is 20 bytes

6.3 Registers

The control registers (16 bit) are for hardware configuration and runtime operation. The default configuration is stored in FLASH memory and loaded into RAM and EEPROM during runtime.

Figure 71. **Memory register mapping**



Access at this memory area is regulated through write and read operations in the Memory Access characteristic and through notification events.

6.3.1 Register types

Persistent registers are stored in EEPROM, data is preserved in case of power loss (battery discharge or failure).

Session registers are stored in RAM, data is preserved as long as the system is supplied (battery or external power).

6.3.2 Errors

Table 29. **Error code mapping**

Name	Error code	Session
NO_ERROR_CODE	0x00	No error
ERROR_LENGHT	0x01	Max payload length is 16 (TBC)
ERROR_WRONG_FORMAT	0x02	Incorrect payload data format
ERROR_NOT_IMPLEMENTED	0x03	Register not implemented (optional)
ERROR_ACTION_NOT_ALLOWED	0x04	Action not allowed
ERROR_REG_IS_READ_ONLY	0x05	Read only register
ERROR_NOT_ALLOWED	0x06	Ctrl field mask is not allowed

7 Bill of materials

Table 30. **STEVAL-WESU1 bill of materials**

Item	Q.ty	Ref.	Part/Value	Description	Manufacturer	Order code
1	1	ANT1	2.4 GHz	Chip Antenna,SMD	Pulse	W3008C
2	1	CN500	Micro_USB_AB		Molex	47590-0001
3	5	C110, C200, C216, C401, C411	1uF, 6.3V, ±10%	Ceramic X5R,SMD 0402	Murata	GRM155R60J105KE19D
4	23	C111, C112, C113, C114, C115, C116, C201, C219, C222, C225, C228, C302, C304, C305, C306, C307, C308, C309, C406, C407, C412, C504, C505	100nF, 16V, ±10%	Ceramic X5R,SMD 0201	Murata	GRM033R61C104KE84D
5	4	C117, C303, C400, C404	10uF, 6.3V, ±20%	Ceramic X5R,SMD 0402	TDK	C1005X5R0J106M050BC
6	2	C118, C119	10pF, 25V, ±0.5pF	Ceramic C0G, NP0,SMD 0201	TDK	C0603C0G1E100D030BA
7	1	C206	51pF, not mounted, 50V, ±10%	Ceramic C0G,SMD 0402	any	GRM1555C1H510GA01D
8	2	C212, C213	12pF, 50V, ±0.1pF	CH,SMD 0201	Murata	GRM0335C1H120GA01
9	2	C214, C221	100pF, 16V, ±10%	Ceramic X7R,SMD 0201	Murata	GRM033R71C101KD01D
10	2	C223, C224	not mounted, not mounted	not mounted,SMD 0402	any	not mounted
11	4	C226, C227, C22, C23	15pF, 25V, ±0.1pF	C0G,SMD 0201	Murata	GJM0336C1E150FB01D
12	1	C229	150nF, 6.3V, ±10%	Ceramic X5R,SMD 0402	Murata	GRM155R60J154KE01D

Item	Q.ty	Ref.	Part/Value	Description	Manufacturer	Order code
13	2	C402, C403	2.2µF, 6.3V, ±20%	Ceramic X5R(EIA),SMD 0402	Murata	GRM155R60J225ME95D
14	3	C405, C408, C409	10nF, 10V, ±10%	Ceramic X7R,SMD 0201	Murata	GRM033R71A103KA01D
15	1	C410	220nF,16V, ±10%	Ceramic X7R,SMD 0402	Murata	GRM155R71C224KA12D
16	1	C500	4.7nF, 50V, ±10%	Ceramic X7R,SMD 0402	Murata	GRM155R71H472KA01D
17	1	C501	47µF, 10V, ±20%	Ceramic X5R,SMD 0805	TDK	C2012X5R1A476M125AC
18	1	D401	RED	LED	VISHAY	VLMS1500-GS08
19	1	D500	WHITE	LED	VISHAY	VLMW1500-GS08
20	1	D501	ESDALC6V1-1U2	ST0201	ST	ESDALC6V1-1U2
21	1	D502	ESDA7P60-1U1M	QFN	ST	ESDA7P60-1U1M
22	1	J500	SWD/JTAG	THR 1.27mm 2x5	SAMTEC	FTSH-105-01-F-D-K
23	1	J501	CON3	SMT 3W 1.2mm pitch	Molex	78171-0003
24	1	L203	10µH, 20%	SMD 0805	Murata	LQM21FN100M70L
25	1	L204	0Ω, ±0, 1nH	SMD 0402	Murata	any
26	2	L205, L206	3.9nH, ± 0.3nH	SMD 0402	Murata	LQG15HN3N9SO2D
27	3	L401, L402, L403	1.5Ω 215 mA	SMD 0201	Murata	BLM03BD471SN1D
28	4	R113, R309, R405, R506	0Ω, ±1%	SMD 0201	any	any
29	10	R116, R117, R119, R120, R121, R122, R123, R200, R201, R501	10kΩ, ±1%	SMD 0201	any	any
30	5	R301, R302, R303, R307, R308	0Ω, ±1%	SMD 0201	any	any
31	5	R114, R118, R304, R305, R306	0Ω, not mounted, ±1%	SMD 0201	any	any
32	1	R400	2kΩ, ±1%	SMD 0201	any	any

Item	Q.ty	Ref.	Part/Value	Description	Manufacturer	Order code
33	1	R401	1Ω, ±1%	SMD 0201	any	any
34	1	R402	60 - 0.05W, ±1%	SMD 0201	any	any
35	2	R403, R408	1kΩ, ±1%	SMD 0201	any	any
36	1	R404	10kΩ NTC - not mounted, ±1%	SMD 0402	Vishay	NTCS0402E3103FLT
37	1	R412	1MΩ, ±1%	SMD 0201	any	any
38	1	R413	33kΩ, ±1%	SMD 0201	any	any
39	1	R500	1MΩ, ±1%	SMD 0402	any	any
40	2	R502, R503	100kΩ, ±1%	SMD 0201	any	any
41	1	R504	100Ω, ±1%	SMD 0201	any	any
42	1	R505	0.05Ω, ±1%	SMD 0402	WELWYN	LRCS0402-0R05FT5
43	1	SW500	SW PUSHBUTTON-DPST		Omron	B3U-3000P
44	1	U201	uFL connector - not mounted, 50Ω - 6 GHz	SMD Coaxial Connector,SMT	Hirose	U.FL-R-SMT-1(10)
45	1	U100	STM32L151VEY6	WLCSP104	ST	STM32L151VEY6
46	1	U200	BLUENRG-MS	VFQPN32 5x5mm	ST	BLUENRG-MSQTR
47	1	U202	BALF-NRG-01D3	FLIP CHIP 4 ball	ST	BALF-NRG-01D3
48	1	U301	LPS25HB	HLGA-10L (2.5 x 2.5x 0.76mm)	ST	LPS25HB
49	1	U302	LSM6DS3	LGA-14L (2.5x3x0.83mm)	ST	LSM6DS3
50	1	U303	LIS3MDL	VFLGA-12 (2.0x2.0x1.0mm)	ST	LIS3MDLTR
51	1	U400	STNS01	DFN12L (3x3mm)	ST	STNS01
52	1	U401	STC3115	CSP (1.4 x 2.0mm)	ST	STC3115
53	1	U402	Voltage Regulator 3.1V	SOT666	ST	STLQ015XG31R
54	1	U502	USBULC6-2M6	uQFN	ST	USBULC6-2M6(uQFN)
55	2	Y2, Y201	NX2012SA 32kHz EXS00A-MU00389		NDK	
56	1	Y101	NX2016SA 24MHZ EXS00A-CS05544		NDK	

Item	Q.ty	Ref.	Part/Value	Description	Manufacturer	Order code
57	1	Y202	NX2016SA 32MHz EXS00A-CS06644		NDK	

Table 31. **Adapter board bill of materials**

Item	Q.ty	Ref.	Part/Value	Description	Manufacturer	Order code
1	2	J1-J2	Con3	Strip Line Male THR 2.54 mm 1x8	Any	Any
2	1	J5	External connector	Strip Line Male THR 2.54 mm 1x8	Any	Any
3	1	J9	JTAG connector	THR 2.54 mm 2x10	Tyco Electronics	2-1634688
4	1	J10	Jump_JTD	Strip Line Male THR 2.54 mm 1x2	Any	Any
5	1	J12	SWD/JTAG	THR 1.27 mm 2x5	Samtec	FTSH-105-01-F-D-K
6	1	C1	100nF, 16V, ±10%	Ceramic X5R, SMD 0201	Murata	GRM033R61C104KE84D
7	1	R1	10 kΩ, ±1%	SMD 0201	Any	Any
8	1	S1	SW pushbutton adapter		C&K	KMR221GLFS
9	1		SWD flat cable	Cable	SAMTEC	FFSD-05-D-08.00-01-N

8 Schematic diagrams

Figure 72. **Main system blocks**

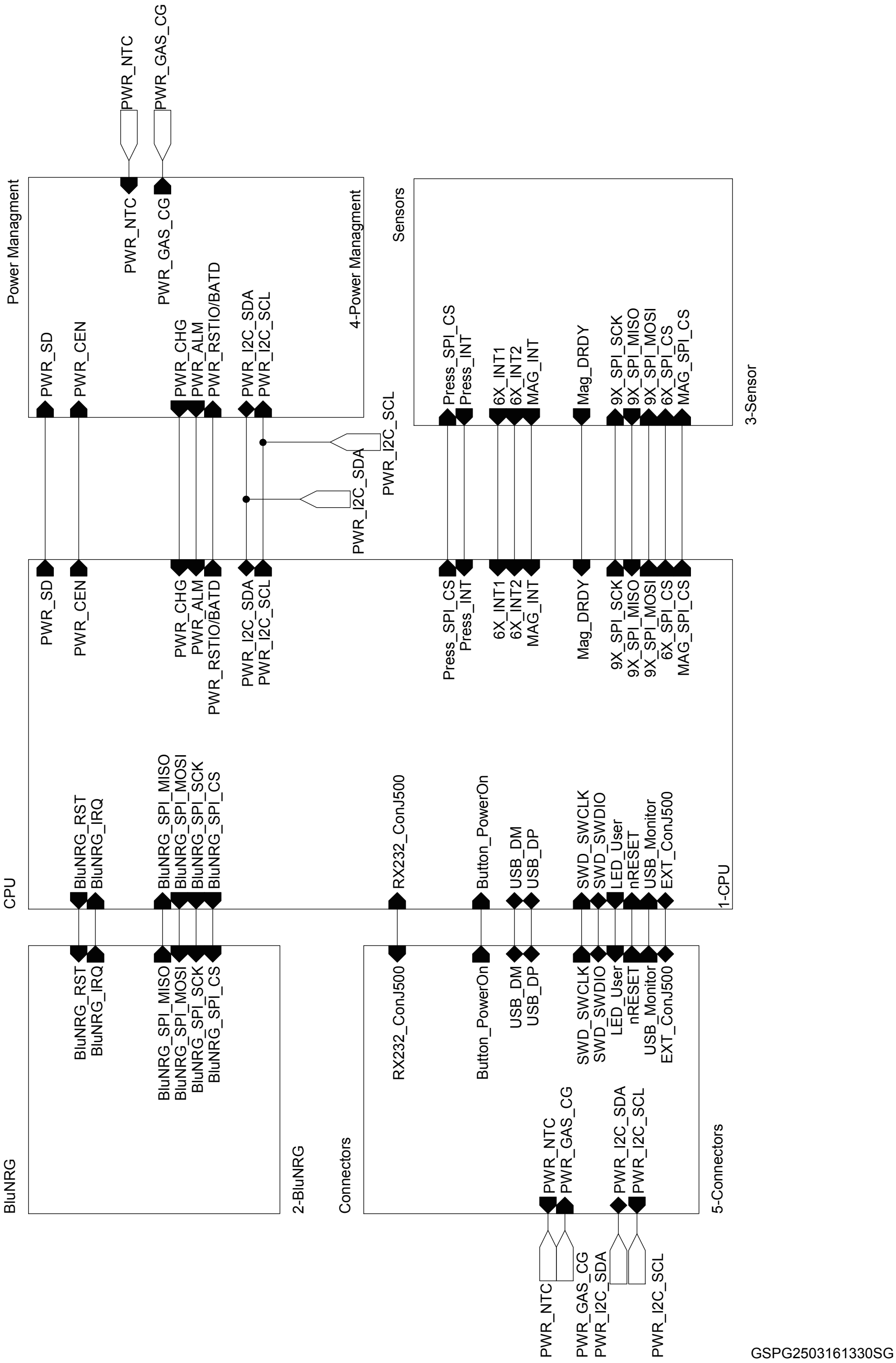
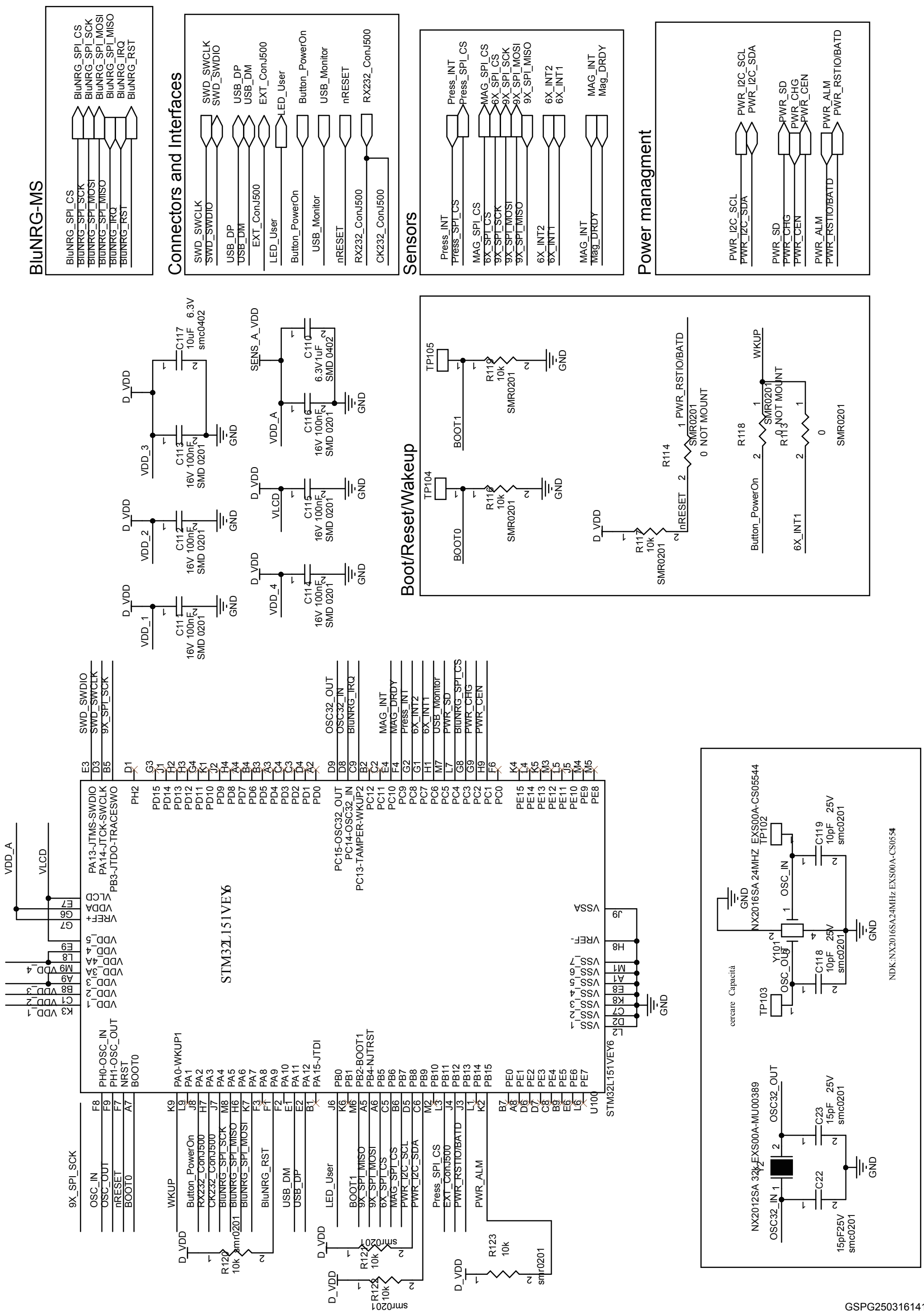


Figure 73. Microcontroller schematics



GSPG2503161415SG

Figure 74. Sensors subsystem schematics

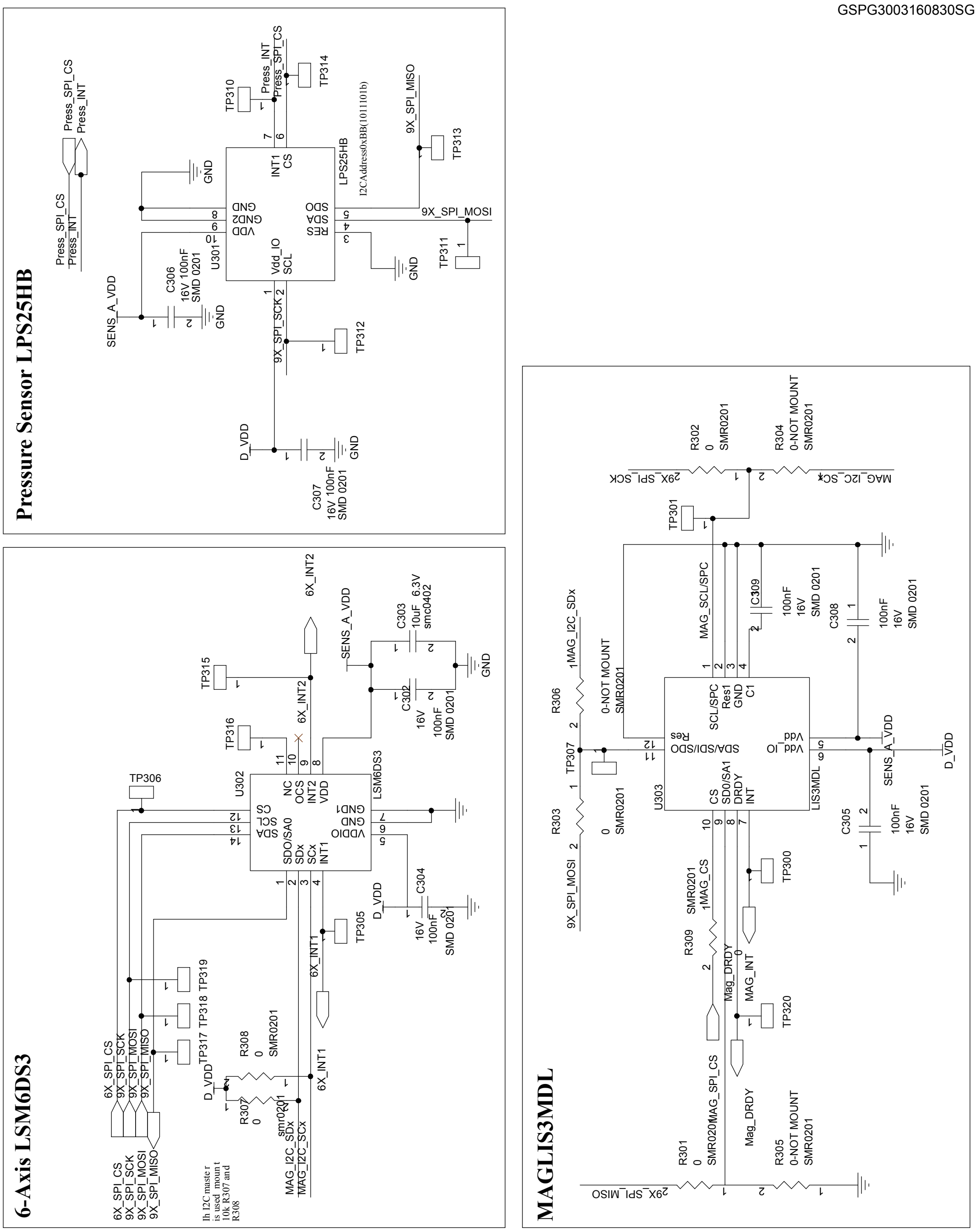


Figure 75. Connectivity subsystem schematics

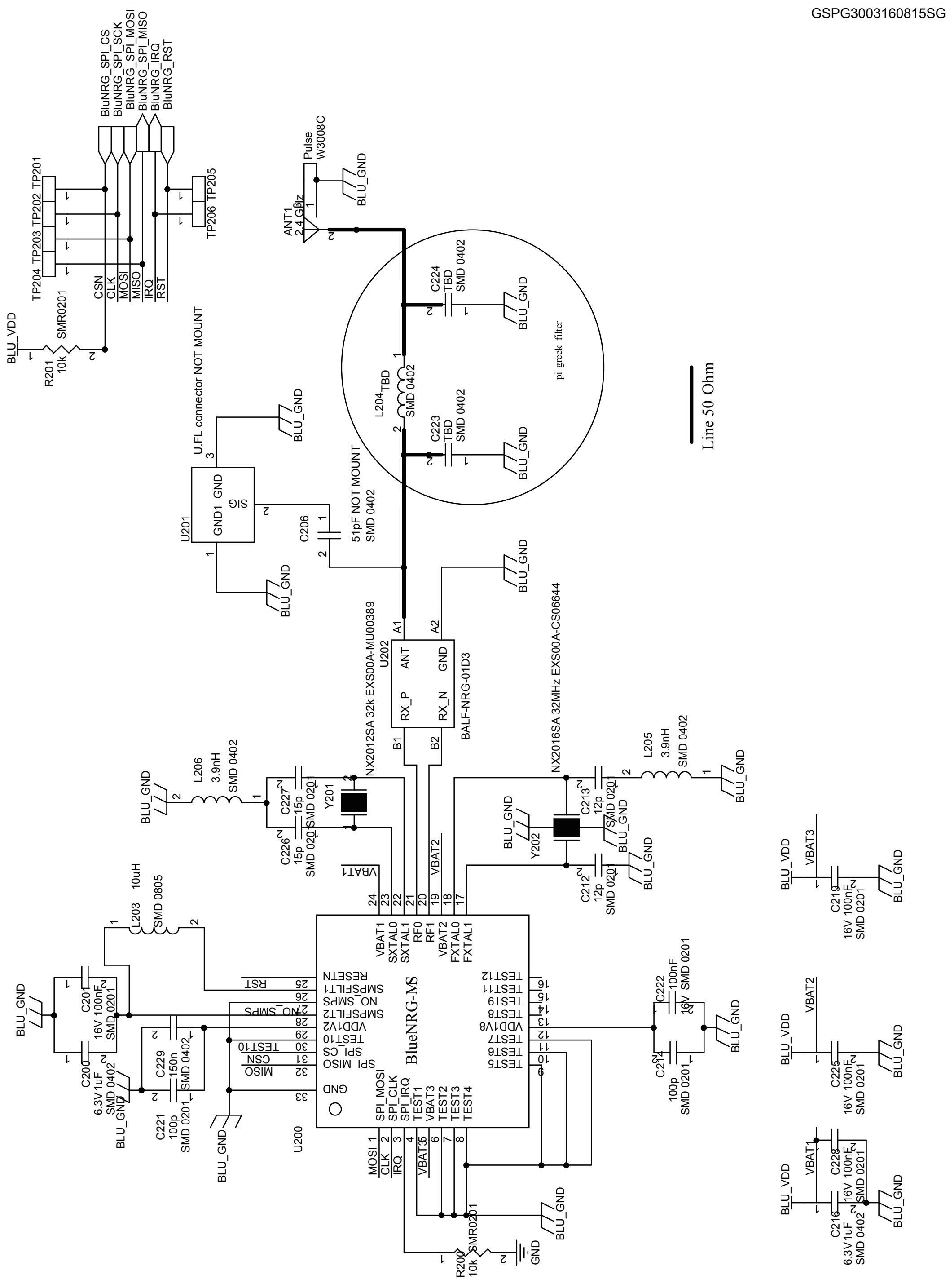
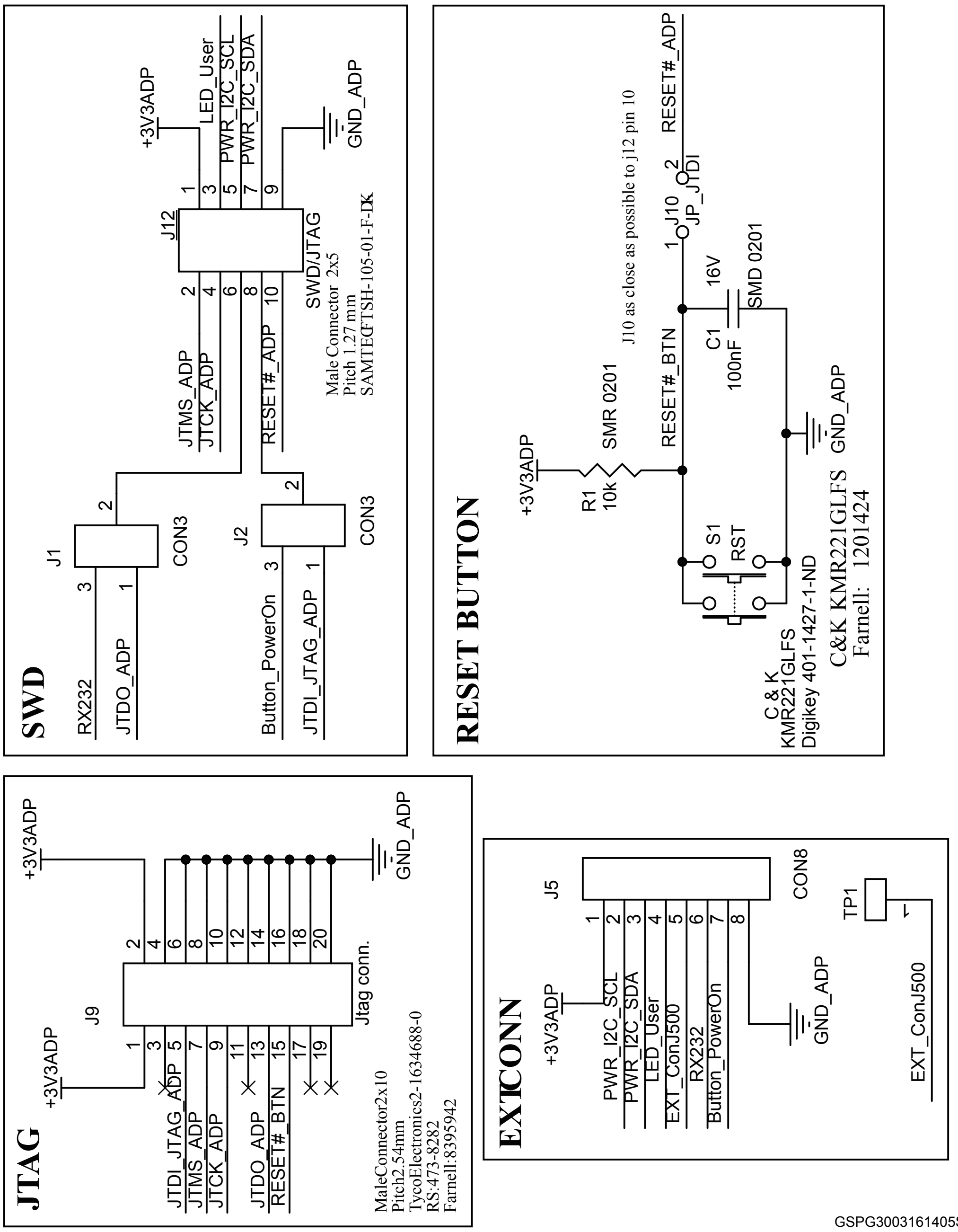
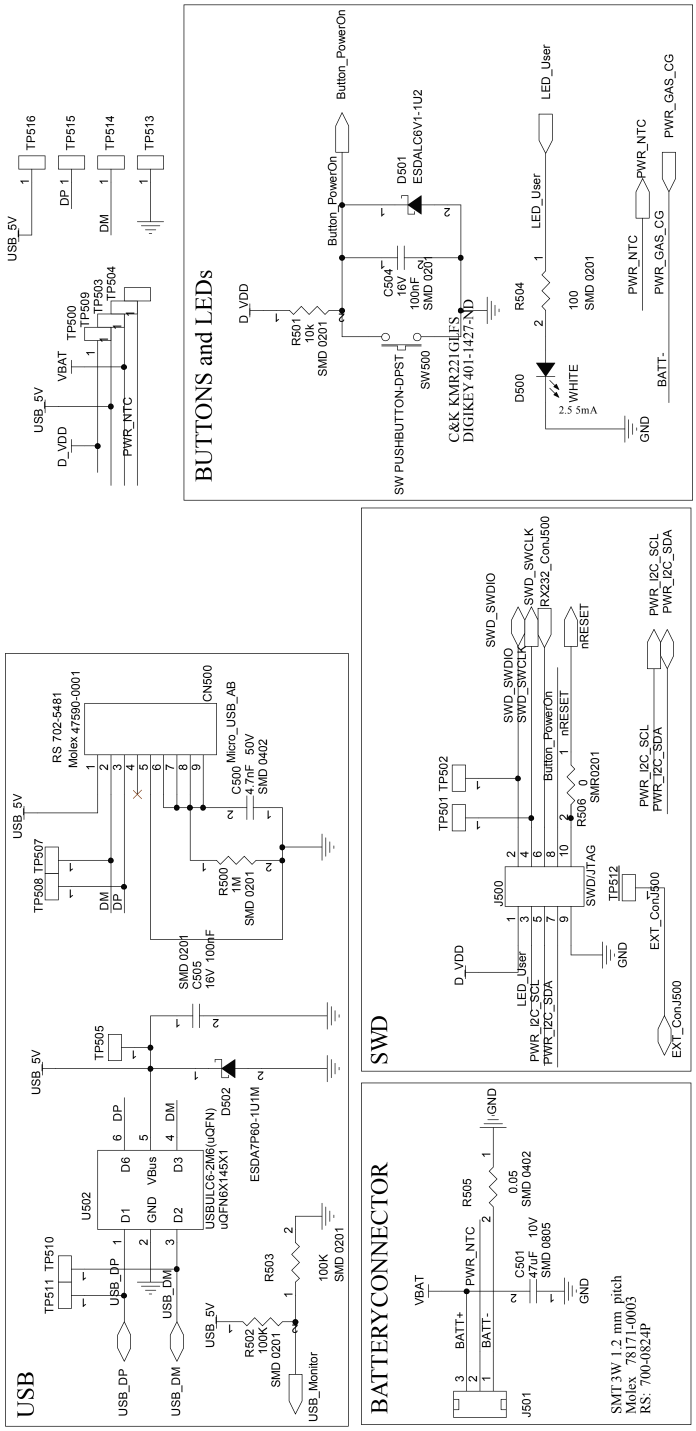


Figure 76. SWD and Reset connection schematics



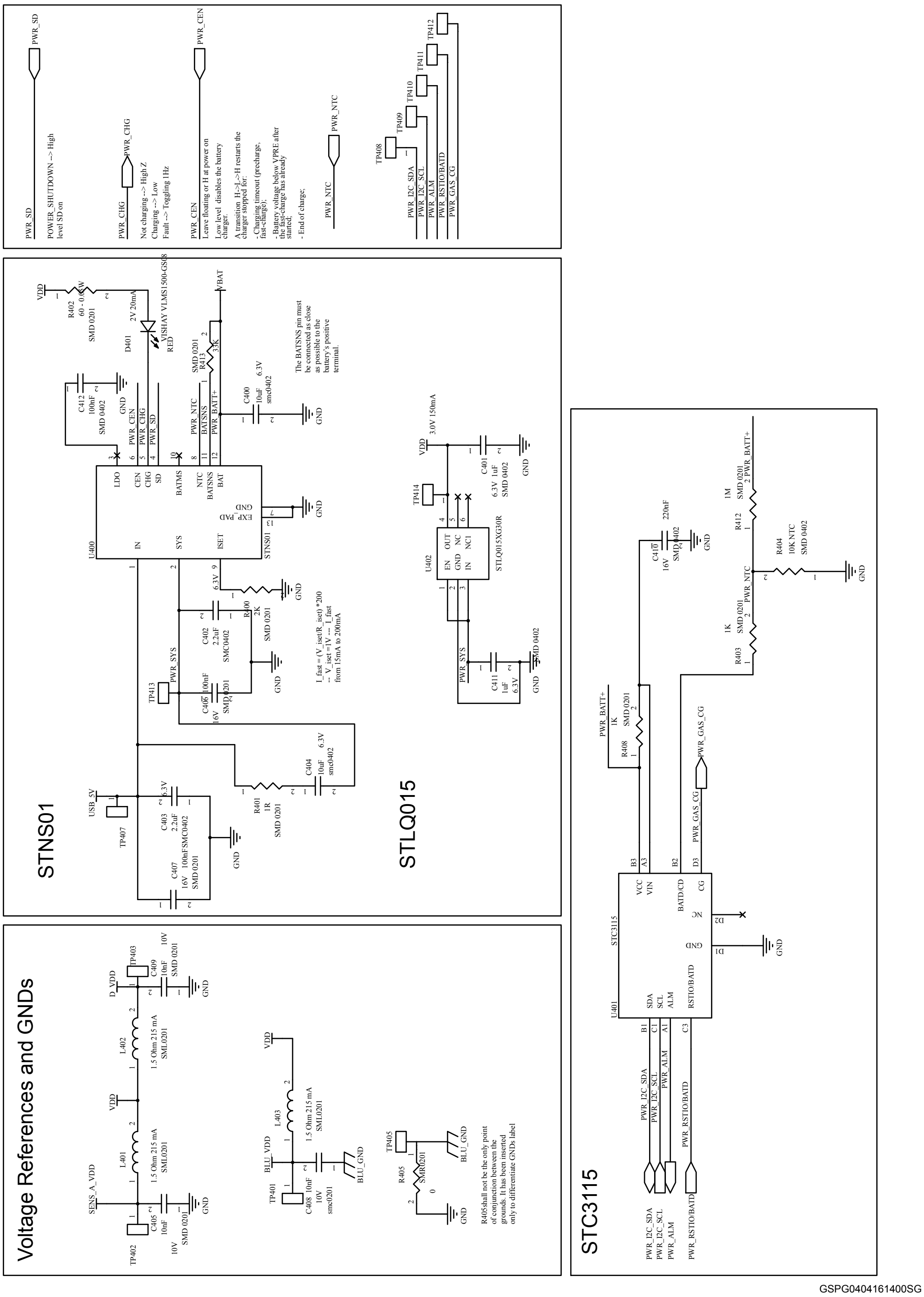
GSPG3003161405SG

Figure 77. External connector schematics



GSPG3003161355SG

Figure 78. Battery and power management subsystem schematics



9 Formal notices required by the U.S. Federal Communications Commission ("FCC")

Model: STEVAL-WESU1

FCC ID: S9NWESU1

Any changes or modifications to this equipment not expressly approved by STMicroelectronics may cause harmful interference and void the user’s authority to operate this equipment.

This device complies with part 15 of the FCC rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference, and
2. This device must accept any interference received, including interference that may cause undesired operation.

For Class A Digital Devices

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

For Class B Digital Devices

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference's by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and the receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

10 Formal notices required by the Industry Canada ("IC")

Model: STEVAL-WESU1

IC: 8976C-WESU1

English:

This Class A or B digital apparatus complies with Canadian CS-03.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

French:

Cet appareil numérique de la classe A ou B est conforme à la norme CS-03 du Canada.

Les changements ou les modifications pas expressément approuvés par la partie responsable de la conformité ont pu vider l'autorité de l'utilisateur pour actionner l'équipement.

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. 'exploitation est autorisée aux deux conditions suivantes: (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

11 Acronyms and abbreviations

Table 32. List of acronyms

Acronym	Description
API	application programming interface
BLE	Bluetooth low energy
BSP	board support package
CLI	command line interface
DFU	device firmware upgrade
FS	full scale
HAL	hardware abstraction layer
HCI	host command interface
IDE	integrated development environment
ODR	output data rate
OTA	over the air
RFU	reserved for future use
SOC	status of charge

Revision history

Table 33. **Document revision history**

Date	Version	Changes
07-Apr-2016	1	Initial release.
17-May-2016	2	Minor text edits Updated Section 3.4: "Toolchains"
01-Feb-2017	3	Text and formatting changes throughout document Updated Figure 8: "ST WeSU Android iOS start page", Figure 26: "STEVALWESU1 firmware architecture", Figure 30: "Middleware folder", Figure 32: "Demo application folder", Figure 34: "Example application files package" Updated Section 3.4: "Toolchains" Figure 35: "ST WeSU app start page (Android and iOS)", Figure 40: "Demo index and Action menu (Android)" Updated Section 4.1: "Demo overview" Updated Figure 42: "ST WeSU app (Android version) environmental demo" Added Section 4.1.2: "Accelerometer event demo" Updated Section 4.1.4.1: "Start/stop logging" Renamed and updated Section 4.1.5.1: "osxMotionFX suite" Updated Figure 56: "Carry position demo" Updated Section 4.2.1: "Settings" and Section 4.2.2: "Debug console" Added Section 4.2.4: "License manager" and Section 4.2.6.1: "Low power mode selection" Added Section 5.4: "Entering DFU mode directly from the App"
10-May-2017	4	Updated Section 2.4.1: "Microcontroller".
17-Jan-2019	5	Document updated according to firmware v1.2.0. Updated Figure 26. STEVAL-WESU1 firmware architecture, Figure 30. Middleware folder and Section 7 Bill of materials. Minor text changes.

Contents

1 Getting started 2
1.1 System setup guide 4
1.2 ST WeSU app setup 4
1.3 System requirements 5
2 STEVAL-WESU1 hardware description 6
2.1 STEVAL-WESU1 board connections 7
2.2 ST-LINK connections 7
2.3 Exposed pad connectors 8
2.4 Hardware architecture 9
2.4.1 Microcontroller 10
2.4.2 Sensors 15
2.4.3 Bluetooth low energy connectivity 15
2.4.4 Battery and power management 16
3 STEVAL-WESU1 Firmware 19
3.1 Overview 19
3.2 Architecture 19
3.3 Folder structure 20
3.3.1 Documentation 20
3.3.2 Drivers 20
3.3.3 Middleware 21
3.3.4 Projects folder 22
3.3.5 Demonstration firmware overview 23
3.3.6 Sample firmware overview 27
3.4 Toolchains 28
4 ST WeSU app 29
4.1 Demo overview 33
4.1.1 Environmental demo 33
4.1.2 Accelerometer event demo 34
4.1.3 RSSI and battery 35
4.1.4 Plot data demo 36

4.1.5 Algorithm demos 38
4.2 Action list overview 43
4.2.1 Settings 43
4.2.2 Debug console 44
4.2.3 Serial console 45
4.2.4 BLE standby 45
4.2.5 Reboot 45
5 Factory settings restore and firmware upgrade 47
5.1 Restoring the factory settings with ST-Link 47
5.2 Device firmware upgrade using USB 48
5.3 Device firmware upgrade OTA via BLE 51
5.4 Entering DFU mode directly from the App 53
6 BlueST Protocol SDK 55
6.1 Advertising format 55
6.1.1 Bluetooth low energy AD structure (max. 31 bytes) 55
6.1.2 AD structures 55
6.2 Services/characteristics 56
6.2.1 HW single feature packet 57
6.2.2 Aggregate feature packet 58
6.2.3 SW single feature packet 1/2 58
6.2.4 Configuration settings 60
6.2.5 Debug 61
6.3 Registers 61
6.3.1 Register types 61
6.3.2 Errors 61
7 Bill of materials 62
8 Schematic diagrams 66
9 Formal notices required by the U.S. Federal Communications Commission ("FCC") 73
10 Formal notices required by the Industry Canada ("IC") 74
11 Acronyms and abbreviations 75

Revision history 76

List of figures

Figure 1. STEVAL-WESU1 package . . . 1
Figure 2. Inside the STEVAL-WESU1 package . . . 2
Figure 3. STEVAL-WESU1 board . . . 2
Figure 4. LiPO battery . . . 3
Figure 5. STEVAL-WESU1 SWD adapter . . . 3
Figure 6. STEVAL-WESU1 silicone wristband . . . 3
Figure 7. Plastic case housing the STEVAL-WESU1 board . . . 4
Figure 8. ST WeSU Android iOS start page . . . 5
Figure 9. STEVAL-WESU1 top side components . . . 6
Figure 10. STEVAL-WESU1 bottom side components . . . 7
Figure 11. STEVAL-WESU1 top and bottom side board connections . . . 7
Figure 12. ST-Link connection using the adapter . . . 8
Figure 13. STEVAL-WESU1 exposed pad connections for battery charging . . . 8
Figure 14. USB plug with external USB connector for DFU update . . . 9
Figure 15. USB connection with external plug after cutting . . . 9
Figure 16. STEVAL-WESU1 functional block diagram . . . 10
Figure 17. Microcontroller subsystem . . . 10
Figure 18. Current consumption in three typical running modes . . . 12
Figure 19. SWD connector and external peripheral connections . . . 13
Figure 20. Programming adapter description . . . 14
Figure 21. Sensor array . . . 15
Figure 22. BLE connectivity subsystem . . . 16
Figure 23. Battery and power management subsystem . . . 17
Figure 24. Battery connector . . . 18
Figure 25. USB connector with ESD protection . . . 18
Figure 26. STEVAL-WESU1 firmware architecture . . . 20
Figure 27. STEVAL-WESU1 firmware package folder structure . . . 20
Figure 28. firmware drivers folder . . . 21
Figure 29. BSP folders . . . 21
Figure 30. Middleware folder . . . 22
Figure 31. Demonstrations and Examples folders . . . 22
Figure 32. Demo application folder . . . 23
Figure 33. Main function with internal infinite loop . . . 25
Figure 34. Example application files package . . . 28
Figure 35. ST WeSU app start page (Android and iOS) . . . 29
Figure 36. ST WeSU app (Android and iOS) device list . . . 30
Figure 37. Node list (Android and iOS) commands . . . 30
Figure 38. Node status (Android and iOS) indication . . . 31
Figure 39. Nodes list (Android and iOS) menu . . . 32
Figure 40. Demo index and Action menu (Android) . . . 32
Figure 41. Demo index and Action menu (iOS) . . . 33
Figure 42. ST WeSU app (Android version) environmental demo . . . 34
Figure 43. ST WeSU app (Android version) Accelerometer demo . . . 35
Figure 44. RSSI and battery charging demo . . . 36
Figure 45. ST WeSU app (Android version) example plot (magnetometer values) . . . 36
Figure 46. Plot length time scale selection . . . 37
Figure 47. Features data plot selection . . . 37
Figure 48. ST WeSU app (Android version) LOG setting menu . . . 38
Figure 49. ST WeSU app (Android version) motion sensor fusion demo . . . 39
Figure 50. Freefall icons . . . 39
Figure 51. Freefall events list . . . 40
Figure 52. ST WeSU (Android) calibration . . . 40

Figure 53. ST WeSU (Android) reset position 41
Figure 54. Commands context menu 41
Figure 55. Activity recognition demo 42
Figure 56. Carry position demo 43
Figure 57. Log settings 44
Figure 58. Debug console command list 45
Figure 59. Set running low power modes 46
Figure 60. Factory settings folder 47
Figure 61. Batch file execution in DOS command window 48
Figure 62. DFU Driver installation 49
Figure 63. DfuSe Demo 49
Figure 64. DfuSe Demo upgrade successful 50
Figure 65. DfuSe Demo leave DFU mode 51
Figure 66. ST BlueDFU start page 52
Figure 67. ST BlueDFU working flow (Android Version) 53
Figure 68. ST BlueDFU device update 53
Figure 69. DFU selection from App settings 54
Figure 70. IEEE 754 single reference 59
Figure 71. Memory register mapping 61
Figure 72. Main system blocks 66
Figure 73. Microcontroller schematics 67
Figure 74. Sensors subsystem schematics 68
Figure 75. Connectivity subsystem schematics 69
Figure 76. SWD and Reset connection schematics 70
Figure 77. External connector schematics 71
Figure 78. Battery and power management subsystem schematics 72

List of tables

Table 1. System current consumption values . . . 11
Table 2. Expansion connector GPIO description . . . 13
Table 3. Memory mapping . . . 26
Table 4. Permanent register location . . . 27
Table 5. Session register location . . . 27
Table 6. Package file list . . . 50
Table 7. BLE advertising structure . . . 55
Table 8. BLE TX power level advertising field . . . 55
Table 9. BLE advertising manuf.-specific advertising field . . . 55
Table 10. Group A features map . . . 56
Table 11. Group B features map . . . 56
Table 12. Device ID enumeration . . . 56
Table 13. Services/characteristics allocation map . . . 56
Table 14. Generic packet format . . . 57
Table 15. Motion sensors packet format . . . 57
Table 16. Battery packet format . . . 58
Table 17. Pressure packet format . . . 58
Table 18. Temperature packet format . . . 58
Table 19. Motion packet format . . . 58
Table 20. Generic packet format . . . 59
Table 21. Sensor fusion packet format . . . 59
Table 22. Free fall packet format . . . 59
Table 23. Activity recognition packet format . . . 59
Table 24. Valid activity types . . . 60
Table 25. Carry position packet format . . . 60
Table 26. Carry position type . . . 60
Table 27. Register access packet format . . . 60
Table 28. Control field . . . 61
Table 29. Error code mapping . . . 61
Table 30. STEVAL-WESU1 bill of materials . . . 62
Table 31. Adapter board bill of materials . . . 65
Table 32. List of acronyms . . . 75
Table 33. Document revision history . . . 76