Getting started with the STM32Cube function pack for IoT tracker node with LoRa® connectivity, GNSS and sensors

Description

FP-ATR-LORA1 is an STM32Cube function pack which lets you read data from environmental and motion sensors, retrieve geo-position from GNSS and send collected data via LoRaWAN connectivity.

The package implements low power profiles and related transitions to ensure long battery autonomy.

This software together with the suggested combination of STM32 and ST devices can be used, for example, to develop asset tracking, fleet management and pet/child tracking applications.

The software runs on the STM32 microcontroller and includes drivers for the LoRa radio, Teseo-LIV3F GNSS module, the motion and environmental sensors, and the power management.

RELATED LINKS

Visit the STM32Cube ecosystem web page on www.st.com for further information

1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
BSP	Board support package
GNSS	Global navigation satellite system
GPS	Global positioning system
HAL	Hardware abstraction layer
I2C	Inter-integrated circuit
IoT	Internet of Things
LoRa	Long range
MEMS	Micro electro-mechanical systems
RTC	Real-time clock
UART	Universal asynchronous receiver-transmitter
WAN	Wide area network

2 FP-ATR-LORA1 software expansion for STM32Cube

2.1 Overview

The FP-ATR-LORA1 software package expands STM32Cube functionality.

The key features of the package are:

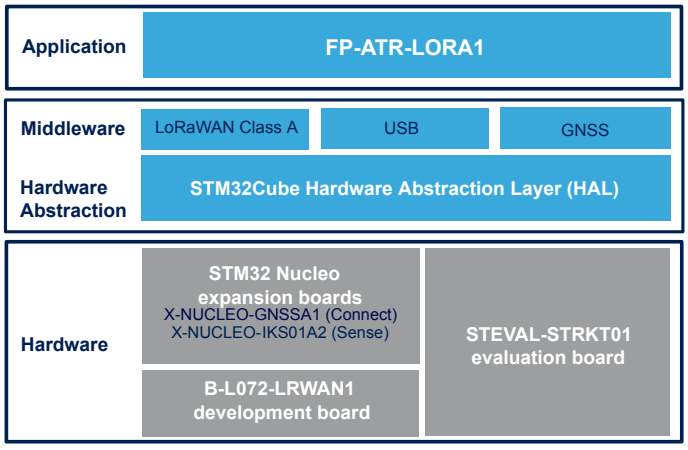
- Complete firmware to connect an IoT node to a LoRaWAN network, sending geo-position coming from GNSS and environmental and sensor data
- Middleware libraries supporting LoRaWAN specification 1.0.3 class A and USB 2.0
- Teseo-LIV3F based GNSS positioning and geofencing
- LoRaWAN keys provisioning via USB
- Power/battery management with low-power operating modes
- Datalogging on external EEPROM for STEVAL-STRKT01 and on internal RAM for B-L072Z-LRWAN1, with data download over-the-air or off-line via USB
- Sample implementation available for STEVAL-STRKT01 evaluation board and for X-NUCLEO-GNSS1A1 and X-NUCLEO-IKS01A2 expansion boards connected to a B-L072Z-LRWAN1 development board
- Easy portability across different MCU families, thanks to STM32Cube
- Free, user-friendly license terms

This software enables gathering environmental sensor data and global positioning coordinates to transmit via LoRa network connection. Collected data can be displayed on the chosen Internet of Things service provider dashboard.

2.2 Architecture

The FP-ATR-LORA1 software supports two hardware platforms: STEVAL-STRKT01 evaluation board and the stack of B-L072Z-LRWAN1, X-NUCLEO-GNSS1A1 and X-NUCLEO-IKS01A2 expansion boards.

Figure 1. FP-ATR-LORA1 software architecture



The FP-ATR-LORA1, compliant with STM32Cube architecture, is structured into a set of layers of increasing abstraction.

The hardware abstraction layer (HAL) interfaces with the hardware and provides the low level drivers and the hardware interfacing methods to interact with the upper layers (application, libraries and stacks). It provides APIs for the communication peripherals (I²C, SPI, UART, etc.) for initialization and configuration, data transfer and communication errors. There are two types of HAL driver APIs:

- generic APIs which provide common and generic functions to the entire STM32 series
- extension APIs which provide specific, customized functions for a particular family or a specific part number

The package extends STM32Cube by providing a board support package (BSP) which deals with board specific peripherals and functions (LED, user button, etc.).

The BSP structure follows the hardware structure, including a component management layer as well as the specific layers of the boards used.

Modules included in the BSP are selected according to the used hardware configuration as shown in the figures below.

Figure 2. FP-ATR-LORA1 software architecture for STEVAL-STRKT01 evaluation board configuration

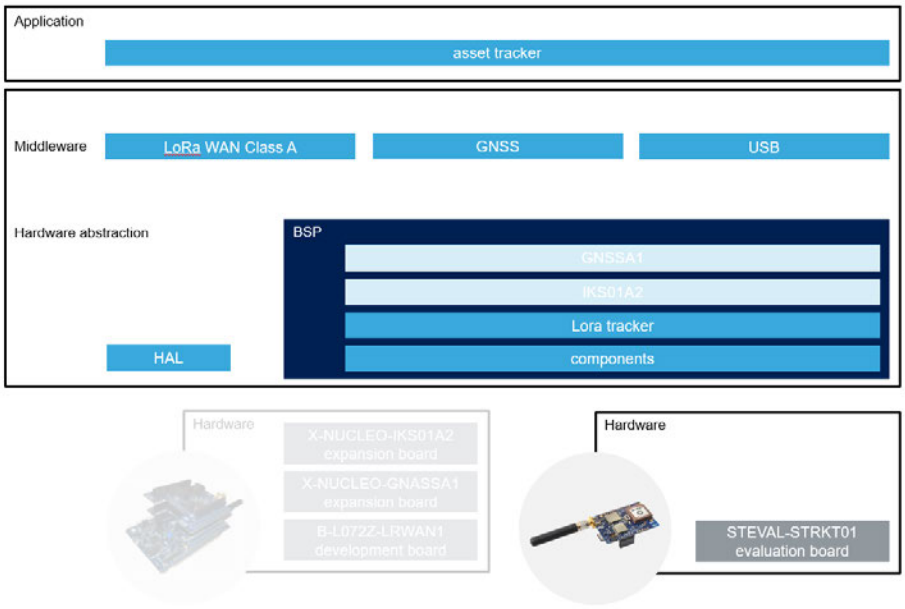
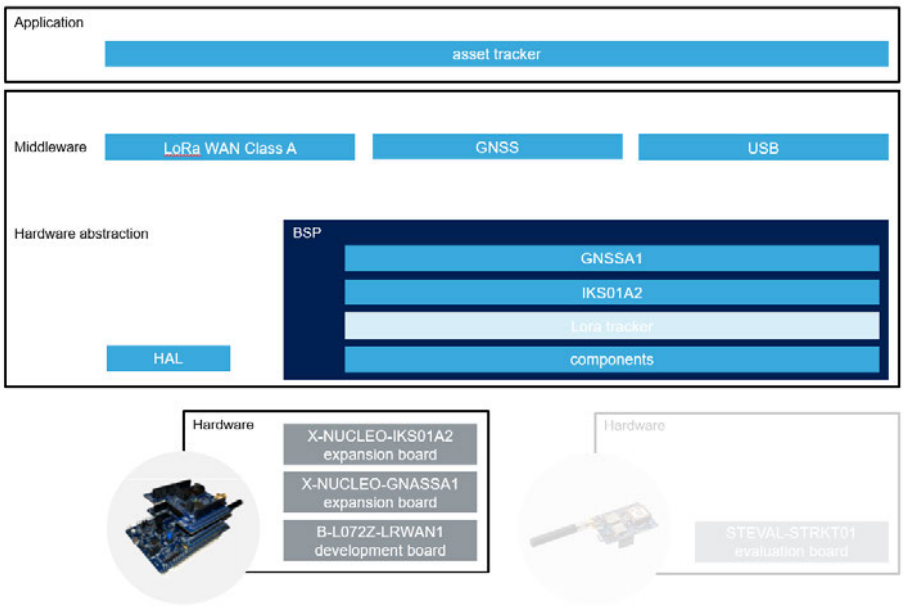


Figure 3. FP-ATR-LORA1 software architecture for B-L072-LRWAN1, X-NUCLEO-GNSSA1 and X-NUCLEO-IKS01A2 stack configuration



This structure improves library code reusability and guarantees easy portability on other devices.

The middleware layer is a set of libraries covering USB, GNSS and LoRaWAN Class A.

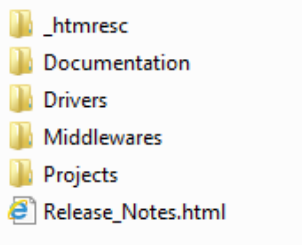
Horizontal interaction among layer components is handled directly by calling the feature APIs, while vertical interaction with the low level drivers is managed through specific callbacks and static macros implemented in the library system call interface.

On top, the application layer contains functions and procedures characterizing the application and which can be changed by the end user.

2.3 Folder structure

The FP-ATR-LORA1 firmware package folder structure follows the layer-based approach of the STM32Cube architecture.

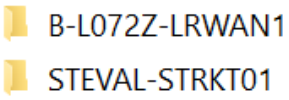
Figure 4. FP-ATR-LORA1 package folder structure



The folders included in the software package are:

- **Documentation**: contains a compiled HTML file generated from the source code which details the software components and APIs.
- **Drivers**: contains the HAL drivers and the board-specific drivers for each supported board or hardware platform, including the on-board components and the CMSIS vendor-independent hardware abstraction layer for ARM Cortex-M processor series.
- **Middlewares**: contains libraries and protocols related to the serial communication of sensor data with a connected PC application, GNSS and LoRaWAN Class A high level procedures.
- **Projects**: contains a sample application used to perform the LoRa asset tracker sample application. This application is provided for the STEVAL-STRKT01 evaluation board and the B-L072Z-LRWAN1 discovery kit platform with three development environments (IAR Embedded Workbench for ARM, RealView Microcontroller Development Kit (MDK-ARM), and System Workbench for STM32 (SW4STM32).

Figure 5. FP-ATR-LORA1: Projects subfolders



2.4 APIs

Detailed technical information with full user API function and parameter description are in a compiled HTML file in the “Documentation” folder.

2.5 Sample application description

An example application for asset tracking using the STEVAL-STRKT01 evaluation board or the B-L072Z-LRWAN1 Discovery kit together with X-NUCLEO-GNSS1A1 and X-NUCLEO-IKS01A2 expansion boards is provided in the “Projects” directory. Ready to be built projects are available for multiple IDEs.

Select the project related to your hardware:

- for B-L072Z-LRWAN1: Projects\B-L072Z-LRWAN1\Applications\LoRa\Asset_Tracker
- for STEVAL-STRKT01: Projects\STEVAL-STRKT01\Applications\LoRa\Asset_Tracker

The Binary folder (in Projects folder) contains precompiled binary files (for example, for REGION_EU868, REGION_US915 AND REGION_US915 HYBRID).

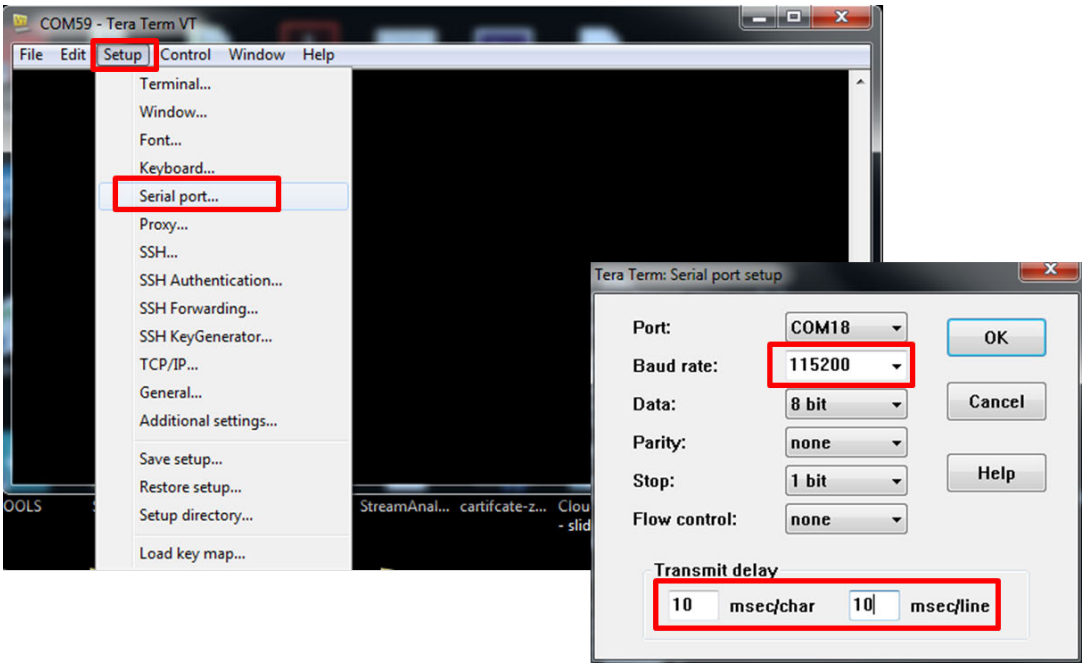
To use the board in a different region, you can use your IDE compiler pre-processor settings to change the selected REGION.

2.5.1 Serial port communication

The application layer provides a serial interface to the user to interact with the board (via USB for the STEVAL-STRKT01 application layer and via USART for the B-L072Z-LRWAN1 application).

The two boards use the same serial port setup as shown below.

Figure 6. Serial port setup



The serial communication allows the user to monitor the application status and the state machine evolution, as well as to send control and configuration commands.

The complete list of allowed commands and the related description are shown in Section 2.5.1 .

Figure 7. **Example of running application**

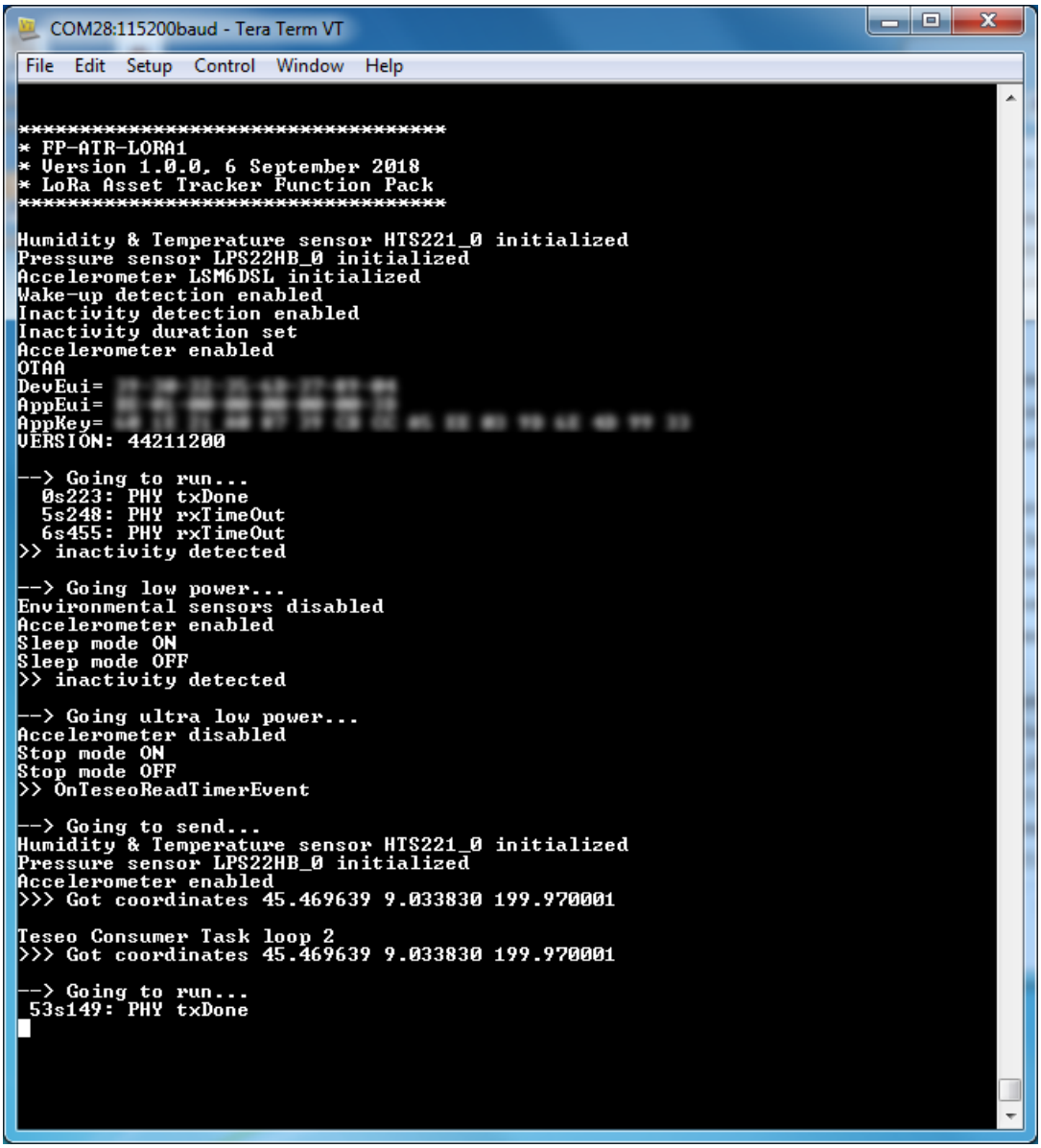


Table 2. **FP-ATR-LORA1 command list**

ASCII command[1]	Label	Get/Set	Description
?	help	G	It shows this command list.
?fwversion	View fw info	G	It shows information about firmware version.
?mcuid	View MCU ID	G	It shows information about microcontroller ID (96-bit unique ID).
!sysreset	System reset	S	It resets the system. Disconnect the VCP and wait for system restart.
!shutdown[2]	System shutdown	S	It switches the system in shutdown mode. Disconnect the USB cable and wait for system shutdown.

ASCII command[1]	Label	Get/Set	Description
?welcomemsg	Welcome message	G	It displays a welcome message, useful to test whether the USB connection has been established and VCP is open.
?platformstatus	Get the platform settings	G	It gets the platform settings.
!defaultsettings	Restore EEPROM default settings	S	It restores EEPROM default settings and has to be followed by `System reset` command.
!lpsensorevent-x	Set Low P on sensor event on/off	S	It enables or disables the system to go in low power mode after a sensor event (accelerometer inactivity). Replace x with 1 to activate the low power mode at sensor event, otherwise replace x with 0 to disable this feature.
!lpsleeptimer-x	Set Low P on sleep timer on/off	S	It enables or disables the system to go in low power mode after a timer event. Replace x with 1 to activate the low power mode at timer event, otherwise replace x with 0 to disable this feature.
!sendonwake-x[2]	Send data on sensor wakeup (on/off)	S	It enables or disables sending sensor data after the accelerometer wake-up event. Replace x with 1 to activate this feature, otherwise replace x with 0 to disable it.
!sendonthreshold-x[2]	Send data on sensor threshold (on/off)	S	It enables or disables sending data after a sensor overshoots the threshold event (low or high humidity, temperature or pressure). Replace x with 1 to activate this feature, otherwise replace x with 0 to disable it.
!loraadronoff-x	Set LoRa ADR on/off	S	The adaptive data rate (ADR) is a mechanism for optimizing network data rates, airtime and energy consumption. This command allows enabling or disabling this feature. Replace x with 1 to activate it, otherwise replace x with 0 to disable it.
!loradr-x	Set LoRa Data Rate	S	It sets the LoRa data rate (values should be between 0 and 5).
!lorainterval-xxxxx	Set LoRa send interval	S	It sets the LoRa sending interval (xxxxx is the interval expressed in ms). The syntax of the ASCII command which sets the LoRa sending interval is: ! lorainterval-xxxxx<CR><LF>. You have to change the xxxxxx digts with the chosen interval.
?loraack	Get LoRa ack variable status	G	It gets the LoRa ack variable status.
!loraack-x	Set LoRa ack variable status	S	It sets the LoRa ack variable status. Replace x with 1 to activate this feature, otherwise replace x with 0 to disable it.
!txtimerintv-xxx	Set tx timer interval	S	Set the system setting for tx timer interval. Replace xxx with the tx timer interval in ms.
!format[2]	Format EEPROM. It loses data	S	Format EEPROM to use with log manager. The EEPROM is prepared to be used with log manager. This command is mandatory before the very first activation of the log manager.
!pushlog[2]	Push current data to EEPROM	S	It stores current data to EEPROM (activity/inactivity of the accelerometer, T, P, H, latitude, longitude, altitude, battery level).
?getsingleitem[2]	Get 1 item from EEPROM	G	It gets one single item from EEPROM.
?getlogs[2]	Get all items from EEPROM	G	It gets all items from EEPROM.

ASCII command[1]	Label	Get/Set	Description
?getunsentlogs	Get items not sent from EEPROM		
?logmanager[2]	Get EEPROM datalog status	G	Get EEPROM log manager status and can be running or not. It also returns the amount of log manager events per type.
!logmanager-x[2]	Set EEPROM datalog ON/OFF	S	It enables or disables log manager in EEPROM. Replace x with 1 to activate this feature, otherwise replace x with 0 to disable it.
?gnssappconf	GNSS get app config data	G	It gets GNSS application configuration status.
!gnssappconf-x-y	GNSS set app config data	S	It configures the GNSS application layer: x is the activation for WAIT FOR FIX when sending data and y is the activation for WAIT FOR FIX when polling data from GNSS.
!powergnss-x[2]	GNSS VDD is switched on or off	S	This command has effect on the GNSS feeding line managed by STBC02 SW1_OA load switch. Replace x with 1 to activate the GNSS power line, otherwise replace x with 0 to break the feeding.
!powereeprom-x[2]	EEPROM VDD is switched on or off	S	This command has effect on the EEPROM feeding line managed by STBC02 SW1_OB load switch. Replace x with 1 to activate the EEPROM power line, otherwise replace x with 0 to break the feeding.
!powertcctrl-x[2]	Type-C controller VDD is switched on or off	S	This command has effect on the feeding line that enables the I²C communication with the USB Type-C port controller as well as its status management. It is managed by STBC02 SW2_OA load switch. Replace x with 1 to activate the Type-C controller power line, otherwise replace x with 0 to break the feeding.
!powersens-x[2]	Sensors VDD is switched on or off	S	This command has effect on the sensors (humidity, temperature and pressure) feeding line managed by STBC02 SW2_OB load switch. Replace x with 1 to activate the sensor power line, otherwise replace x with 0 to break the feeding.
?debugmode	Get the debug mode	G	It gets debug over USB mode status.
!debugmodeSs	Set the debug mode	S	It sets debug mode status. Replace last s character with e or E to activate the debug mode, otherwise replace it with d or D to disable this functionality.
?devicejoinstatus	Get the LoRa device join status	G	It gets the LoRa device join status.
?devicejoinparam	Get the LoRa join parameters	G	It gets the LoRa join parameters.
!deviceeui-xxxxxxxxxxxxxxx	Set the LoRa device EUI	S	It sets the device EUI. In the command syntax, replace each x character with one of the 16 nibbles composing the LoRa device EUI.
!joineui-xxxxxxxxxxxxxxx	Set the LoRa join EUI	S	It sets the join EUI. In the command syntax, replace each x character with one of the 16 nibbles composing the LoRa join EUI.
!appkey-xxxxxxxxxxxxxxx	Set the application key	S	It sets the application key. In the command syntax, replace each x character with one of the 32 nibbles composing the application key.
!ntwkkey-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	Set the network key	S	It sets the network key. In the command syntax, replace each x character with one of the 32 nibbles composing the network key.

ASCII command[1]	Label	Get/Set	Description
!eraselorakeys	Erase LoRa keys in EEPROM	S	It erases LoRa keys.
!triggerlora	Trigger a LoRa sending	S	It forces a LoRa data sending.
?includeepochtime	Get 'Include Epoch time' variable status	G	It gets the 'Include Epoch time' variable status.
!includeepochtime-x	Set 'Include Epoch time' variable status	S	It Sets the 'Include Epoch time' variable status. Replace x with 1 to include Epoch time, otherwise replace x with 0 to not include it.
?joinreqintvshort	Get 'Shorten join req intv' variable status	G	It gets the shorten join request interval variable status.
!joinreqintvshort-x	Set 'Shorten join req intv' variable status	S	It sets the shorten join request interval variable status Replace x with 1 to enable shorten join request interval, otherwise replace x with 0 to not disable it.
!sysrun	Set system state to run	S	It sets system state in run mode.
!syslp	Set system state to low power	S	It forces the system state to low power.
!sysulp	Set system state to ultra low power	S	It forces the system state to ultra low power.
!gpscoldstart	GPS cold start	S	It performs a GPS cold start initialization.
?gpsgetposition	Get GPS position	G	It gets the GPS position.
!geofence-p-rrrrr	Config geofence	S	The command manages the geofence functionality. You have to replace p with: • H if the geofence is centered on current GPS coordinates • L if the geofence is centered on ST Catania site coordinates • C if the geofence is centered on ST Lecce site coordinates • X to disable geofence functionality Moreover if H, L or C is selected, the command must be completed with the '-' character, replacing rrrrrr with the radius expressed in meters.
?geofence	Get geofence status	G	It gets the GPS geofence status.
?sensordata	Get sensors data	G	It gets the sensors data.

1. *All ASCII commands must end with the <CR> <LF> characters (where <CR> is the 0x0D carriage return byte and <LF> is the 0x0A line feed byte).*
2. *Not available for B-L072Z-LRWAN1.*

2.5.2 FP-ATR-LORA1 state machine

The FP-ATR-LORA1 application firmware implements a state machine that can be used to develop, for example, asset tracking, fleet management and pet/child tracking applications.

The state machine is composed of three main states: Run, Low Power and Ultra Low Power. However, other temporary states (Read, Send, Check Acknowledgement, Retrieve Data, Prepare Low Power, Prepare Ultra Low Power) can be set to perform customized actions, such as reading sensors or sending data through the LoRaWAN network.

At system reset, the device remains running to collect data from sensors and GNSS and monitor the sensor values to detect if some thresholds are overshoot or a geofence alarm is reported by the GNSS.

When the accelerometer internal motion detection algorithm signals that the device is at rest, the device is put in low power mode (inactivity), disabling some sensors and/or power domains; the device can be woken up again by the accelerometer itself when it is moved (motion).

If the device remains in low power mode for a long time, a timer will trigger it to enter ultra-low power mode; in this state, the power consumption is reduced at the minimum. After entering the ultra-low power mode, the system wake ups periodically only to read sensors and send data. You can implement a check on the sensor data (acceleration, environmental data, geofence, etc.) to exit the ultra-low power mode and enter run mode (see `CheckSensorsData()` function in main.c).

Run: is the default state at system reset. All the devices are initialized and the application polls the environmental sensors, the accelerometer, the GNSS sensor and battery ADC converter for updated data.

Sensors and GNSS receiver are continuously monitored, to detect whether the sensor thresholds are overshoot or the GNSS sends geofence alerts. These events cause the system to go to Read And Send mode.

The accelerometer is also monitored to check:

- inactivity: no movement is detected for a certain amount of time
- wake-up: a movement is detected (the event can be triggered by briefly shaking the board)

Three timers are set to:

- switch to Read state at a user-configurable time interval
- switch to Send state
- switch to Ultra Low Power mode after a certain interval

All subsystems, sensors, GNSS and MEM_VDD are powered and the devices are running. The MCU is in run mode. STUSB1600_VDD is activated depending on the USB cable connection (for more details, see Table 1).

Low Power: the MCU enters sleep mode to reduce power consumption and can be woken up by accelerometer interrupt events; the environmental sensors are disabled. If accelerometer wake-up event or timer event are detected, the system switches to Read state and, subsequently, to Send state. If accelerometer inactivity event is detected, the system switches to Ultra Low Power state. Before entering this state, a Prepare Low Power state is executed to disable unused power paths (no GNSS and no environmental sensors) and set a wakeup timer.

Ultra Low Power: the MCU enters Stop mode to reduce the power consumption at the minimum.

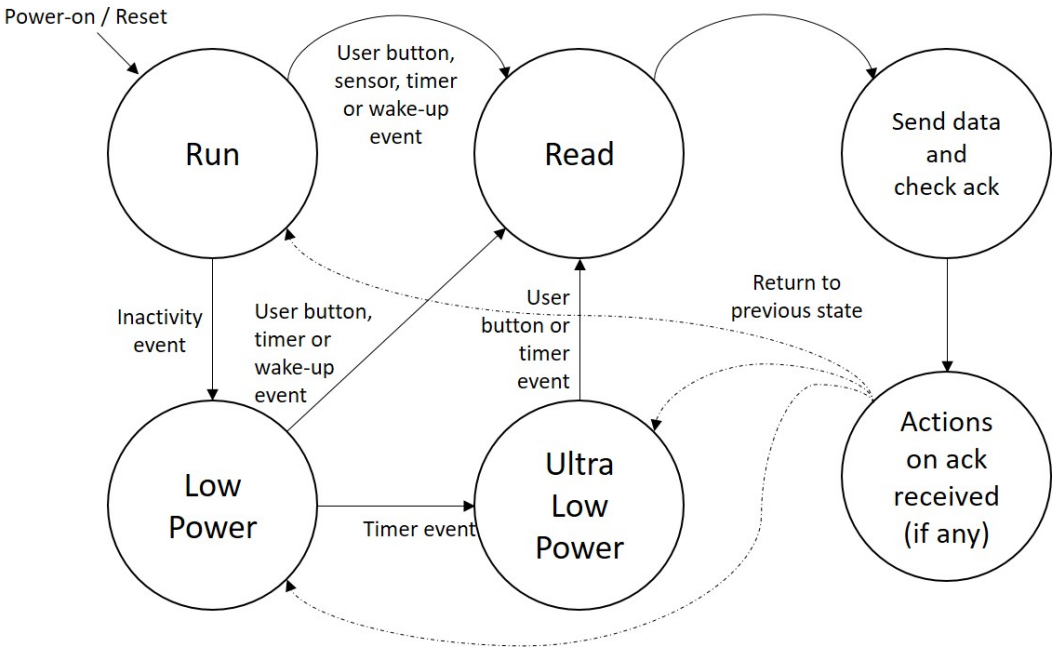
Environmental and accelerometer sensors are disabled and the MCU can be woken up only by the timer interrupt event. When this event is triggered, the system switches to Read state and, subsequently, to Send state.

Before entering Ultra Low Power state, a Prepare Ultra Low Power state is set to disable unused power paths: the sensors subsystem, SENS_VDD, the GNSS subsystem, LIV3_VDD and MEM_VDD, are not powered. The accelerometer is put in shutdown mode by software command. The MCU is in stop mode. STUSB1600_VDD is activated depending on the USB cable connection (for more details, see Table 1).

Read: the application gathers data from environmental, accelerometer and GNSS. After the Read state, data are ready to be saved in the internal EEPROM and/or sent over the LoRaWAN network. The sensors are read and evaluated also in run mode. Anyway, this dedicated “temporary state” is implemented to have a coherent packet of data acquired from sensors at a specific timestamp.

Send: the application sends data to the LoRaWAN network packed in a single message. After the message is sent, the application performs the check on the acknowledgement (if this feature is enabled, i. e. `LORAWAN_DEFAULT_CONFIRM_MSG_STATE` define set to `LORAWAN_CONFIRMED_MSG`) and then the state machine switches back to the previous state.

Figure 8. **STEVAL-STRKT01: application state machine**



2.5.2.1 *Enabling confirmed LoRaMAC frame*

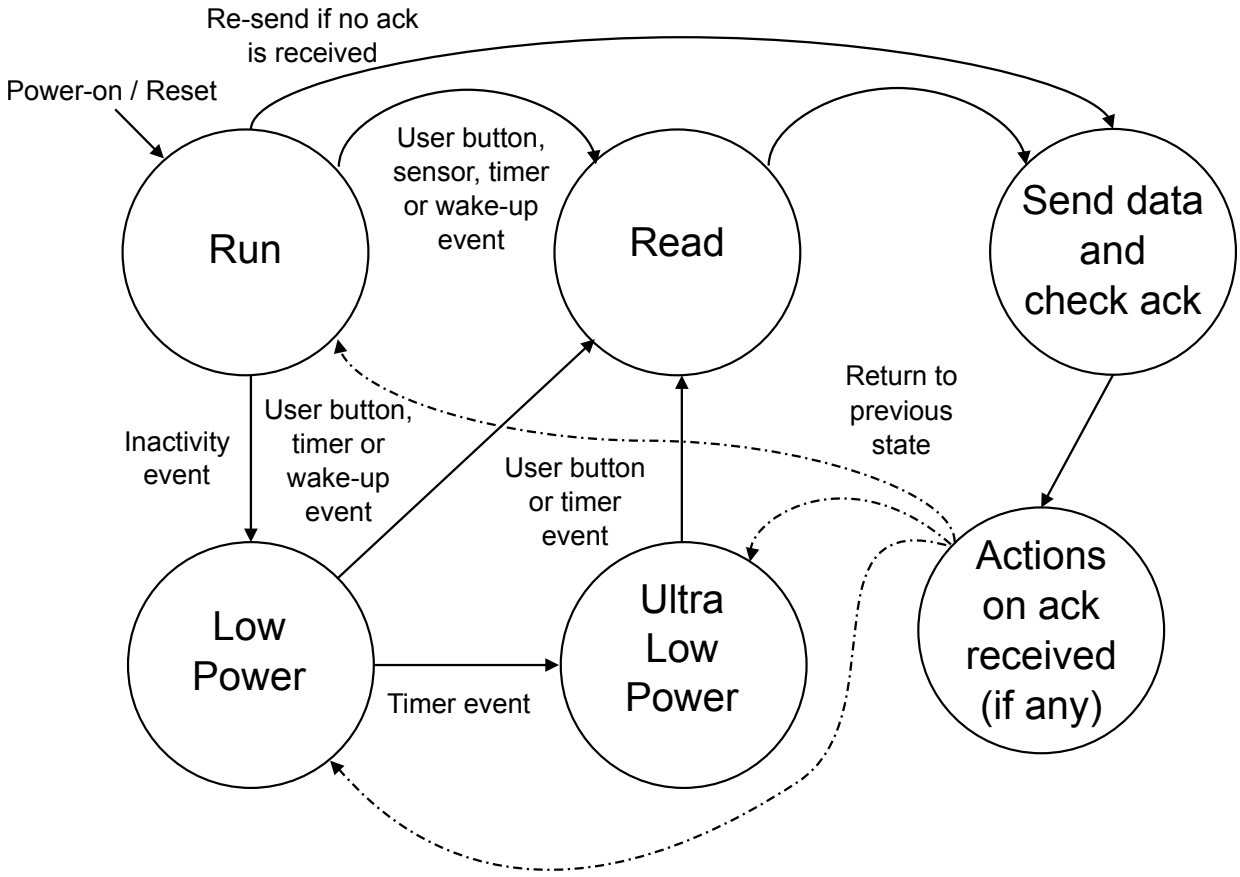
Using acknowledged messages, the device recognizes if a network is available, that is a LoRaWAN gateway is in range.

After the device tries sending a data payload, the ack reception is verified. If no acknowledgement is received, the device tries sending the data again.

Refer to `LORAWAN_DEFAULT_CONFIRM_MSG_STATE` define in the firmware and set to `LORAWAN_CONFIRMED_MSG` to enable this feature, `LORAWAN_UNCONFIRMED_MSG` otherwise.

For STEVAL-STRKT01 only, you can use the `!loraack` USB command. Refer to Section 2.5.1 for further details.

Figure 9. **Application state machine with confirmed LoRaMAC frame**



2.5.2.2 *Data download over the air*

The confirmed LoRaMAC frame makes the device detect if a gateway is in range. When no gateway is in range for a long time and data cannot be delivered, the device can save the data in the internal EEPROM and send it when the network becomes available again.

This feature can be activated by `LOG_MANAGER_DEF` define set to 1 and/or `!logmanager-1` command (see Section 2.5.1 for details); the data are stored in the EEPROM after the Read state; then, in the Send state, the device tries to send the oldest data payload. If no ack is received, the device tries to send the data once again. If the ack is received, the device increases the frequency of the transmissions to download the entire datalog over the air.

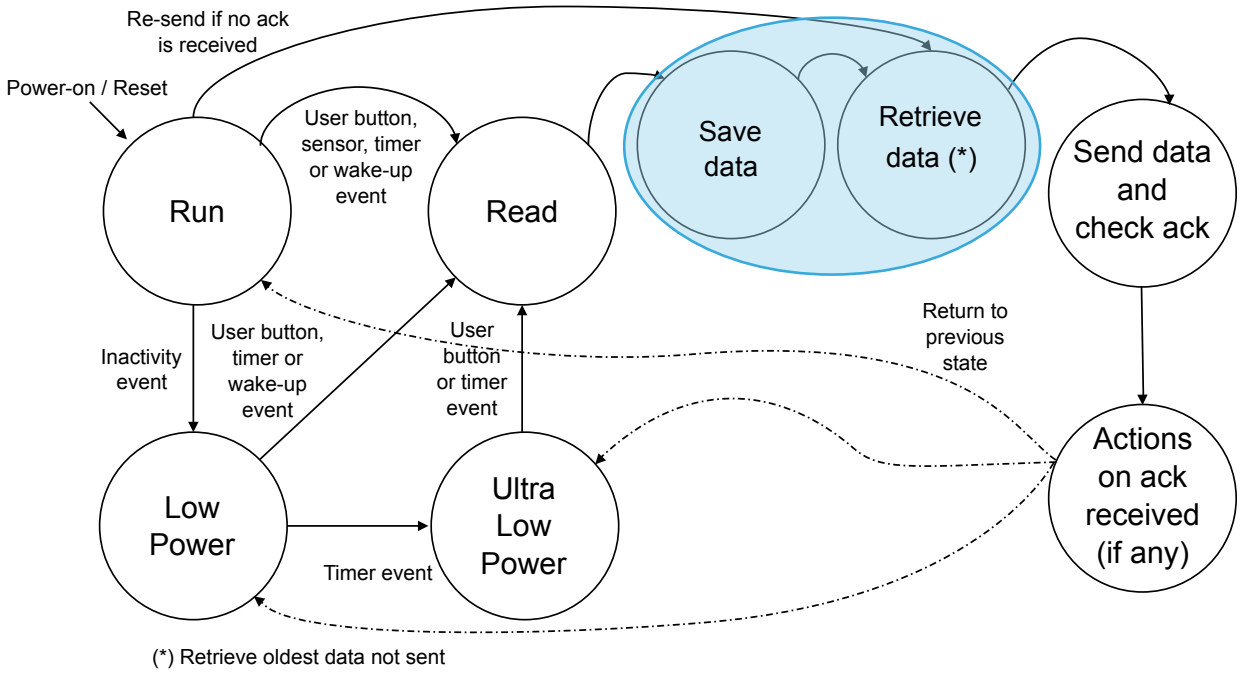
Note: For US915 region, due to FCC requirements, the data sending is allowed only if DR is higher than 1 (due to payload size limitation).

For STEVAL-STRKT01 only, use the following USB commands to activate this feature:

- mandatory:
  - `!defaultsettings`
  - `!sysreset`
  - `!format`
  - `!includeepochtime-1`
  - `!loraack-1`
  - `!txtimerintv-10000`
  - `!joinreqintvshort-0`
  - `!logmanager-1`
- mandatory in US915 region:
  - `!loraadronoff-1`
- optional:
  - `!gnssappconf-1-1`
  - `!lpsensorevent-0`
  - `!lpsleeptimer-0`
  - `!sendonwake-0`
  - `!sendonthreshold-0`
  - `!sysreset`

Refer to Section 2.5.1 for further details on the commands to use.

Figure 10. Application state machine with data download over the air



2.5.3 LoRa packet format

The data sent to the LoRaWAN network is packed according to the Cayenne low power payload (LPP) data format.

The application sends:

- epoch time - optional (see command list: if it is included, the application looses compatibility with Cayenne LPP payload, so the Cayenne dashboard cannot be used)
- pressure
- temperature
- humidity
- GPS coordinates (latitude, longitude, altitude)
- accelerometer data
- battery level information

For more information see http://mydevices.com/cayenne/docs/lora/#lora-cayenne-low-power-payload-uplink-payload-structure.

You can implement your own packet format (see FP-ATR-LORA1, file *main.c*, function `Send(void)` for details).

2.5.4 Storing and retrieving sensor data

For the STEVAL-STRKT01, the sensor data can be stored in the eeprom when in Send state via a firmware module, called log manager that manages formatting, saving and retrieving data.

The log manager deals with three types of data: normal, system and critical.

In the developed application, only the normal data storage is implemented for data logging.

The eeprom must be formatted, using the "`!format`" command (see Section 2.5 Sample application description).

Then, you can enable the log manager by issuing the `!logmanager` command setting 1 as parameter.

At each sending, a set of data, containing Timestamp, sensor and GNSS data, is saved (see the table below).

To manually save an item, without enabling the log manager, use the `!pushlog` command. The commands related to log manager are: `!format`, `!logmanager`, `?logmanager`, `!pushlog`, `?getsingleitem`, `?getlogs` (refer to Section 2.5.1 Serial port communication).

The log manager data format is detailed in the table below.

Table 3. Datalog format details

Size	Data	Units
32b	Timestamp	s, internal RTC value or epochtime if it is enabled
16b	Temperature	°C * 100
16b	Pressure	hPa/10
16b	Humidity	percentage * 10
32b	Latitude	Sexagesimal degree converted to decimal.(1)
32b	Longitude	Sexagesimal degree converted to decimal. (1)
32b	Altitude	m

1. *For details see FP-ATR-LORA1, file main.c, function* `convertCoord`.

2.6 Configuration and registration

LoRa® is a long range wireless area network allowing low-power sensors to report over ranges of up to dozen (or hundreds) kilometers.

The sample application collects environmental, motion and geo-position data and sends them over the LoRaWAN network. To connect to LoRaWAN network, you need a LoRa gateway.

Through the gateway, data can be sent to a LoRaWAN network server and then forwarded to an application network, where they are finally displayed in a dashboard.

To join a LoRaWAN network, each end-device has to be personalized and activated in two ways: via Over-The-Air Activation (OTAA) or via Activation By Personalization (ABP).

The FP-ATR-LORA1 package can handle both methods, but in the default example only OTAA is taken into consideration.

All the keys can be set by using the USB command interface.

2.6.1 LoRaWAN keys

For over-the-air activation, end-devices must follow a join procedure before exchanging data with the network server.

An end-device has to go through a new join procedure every time it loses the session context information.

The join procedure requires the end-device to be customized with:

- **DevEUI** – for theFP-ATR-LORA1, we suggest to use the default DevEUI related to the STM32 MCU ID
- **JoinEUI** - application EUI for v LoRaWAn 1.0.3 Specification or older
- **NwkKey**
- **AppKey** - set at the same value of NwkKey for compatibility with Specs 1.0.2 and 1.1

Usually the JoinEUI (AppEUI), NwkKey and AppKey are provided by the chosen network server once a new device is enrolled.

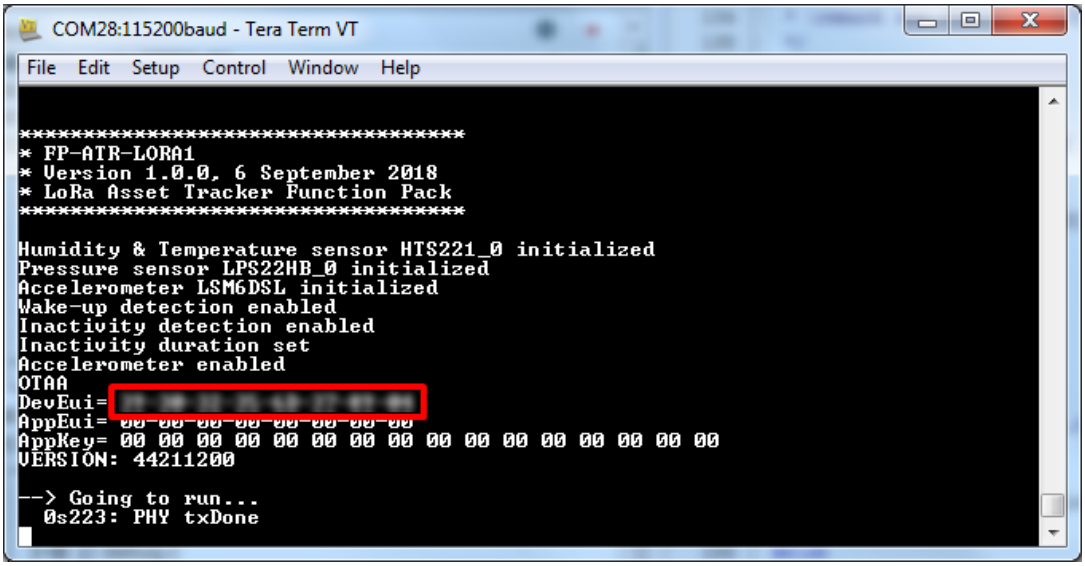
2.6.2 B-L072Z-LRWAN1 Discovery kit plus STM32 Nucleo expansion board setup

To configure and register your device, follow the steps below.

Step 1. Flash the application binary *Projects\B-L072Z-LRWAN1\Applications\LoRa\Asset_Tracker\Binary\mlm32l07x01.bin* on the B-L072Z-LRWAN1.

Step 2. Open a serial connection with 115200 baud speed and 8N1 settings.
On the first run, take note of the DevEUI string necessary to identify the device.

Figure 11. DevEUI string highlighted



2.6.3 STEVAL-STRKT01 setup

To configure and register your device, follow the steps below.

Step 1. Flash the application binary *Projects\B-L072Z-LRWAN1\Applications\LoRa\Asset_Tracker\Binary\STEVAL_STRKT01_GetDevEui.bin* on the STEVAL-STRKT01.

Step 2. Open a serial connection with 115200 baud speed and 8N1 settings.
On the first run, take note of the DevEUI string necessary to identify the device (refer to Figure 11. DevEUI string highlighted).

2.6.4 Gateway setup

The LoRa gateway allows end-nodes to send and receive packets to/from LoRa network servers.

The application described herein requires a working LoRa gateway which has to be registered on the network server used by the end-node.

RELATED LINKS

The Things Network Seed Gateway

Loriot Seed Gateway

2.6.5 STEVAL-STRKT01 registration on LoRaWAN network server

Before using the STEVAL-STRKT01 and FP-ATR-LORA1, you have to register the device on a network server and an application server.

Important: *The system works with 868 Mhz and 915 Hz, thus the network server region has to be set accordingly.*

Registration instructions on LoRiot can be found here. When using ST Asset Tracking web dashboard, a TTN network server must to be used and the registration instructions are available on st.com (https://dsh-assetracking.st.com/#/howto).

Step 1. After enrolling a new device into the network server, set the received keys on the same device.

Note: *Instead of recompiling and flashing the project, the LoRa keys can also be set at runtime and stored in the board EEPROM memory by the following commands used in the serial interface:*

- `!deviceeui-xxxxxxxxxxxxxxxx` *to set Device EUI*
- `!joineui-xxxxxxxxxxxxxxxx` *to set Application EUI*
- `!appkey-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx` *to set Application Key*
- `!ntwkkey-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx` *to set Network Key*

A complete set of keys (deviceeui, joineui, appkey and ntwkkey) must be sent to the device. For example, if you want to use the device eui generated by the device itself, you also have to issue the command `!deviceeui` *with the device eui from the logs*

After the four commands, restart the board with the `!sysreset` *command.*

If you are using 1.0.3 LoRaWAN versions (or below) of the specifications, the below table shows the name equivalence between the versions.

Table 4. ***LoRaWAN specifications: key name equivalence among 1.0.x, 1.0.4 and 1.1.x versions***

1.0.x LoRaWAN versions	*1.0.4 or 1.1.x LoRaWAN versions*
LORAWAN_DEVICE_EUI	*LORAWAN_DEVICE_EUI*
LORAWAN_APP_EUI	*LORAWAN_JOIN_EUI*
N/A	*LORAWAN_APP_KEY*
LORAWAN_APP_KEY	*LORAWAN_NWK_KEY*

Step 2. Recompile and flash the project on the chosen board (B-L072-LRWAN1 or STEVAL-STRKT01).
Skip this step if you set the keys by USB commands.

2.6.6 Application server setup

ST provides an Asset tracking dashboard (see Figure 12) that allows the user to discover Asset tracking ST solution, including STEVAL-STRK01. To use FP-ATR-LORA1 with ST dashboard, go to https://dsh-assetracking.st.com/#/howto and follow the procedure.

Several application servers are also available on the web to offer asset tracking services, for example Tago.io reports here the steps to register the STEVAL-STRK01 on its dashboard.

Figure 12. **ST Asset tracking dashboard - main page**

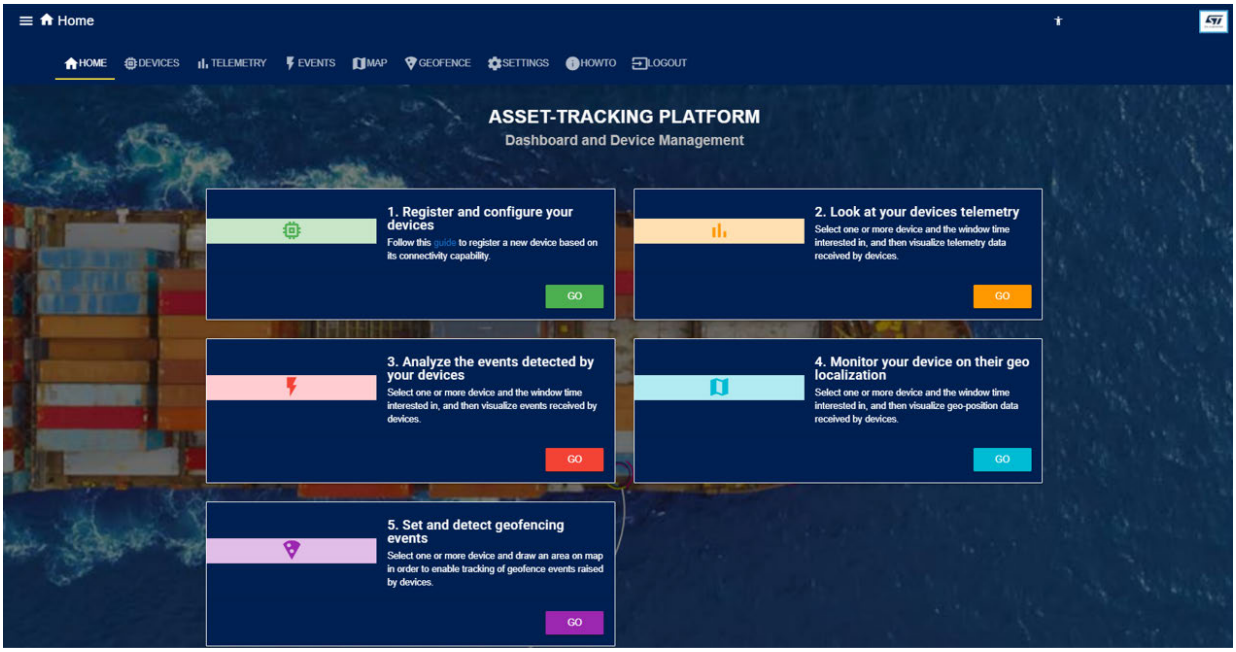


Figure 13. **ST Asset tracking dashboard - device view**

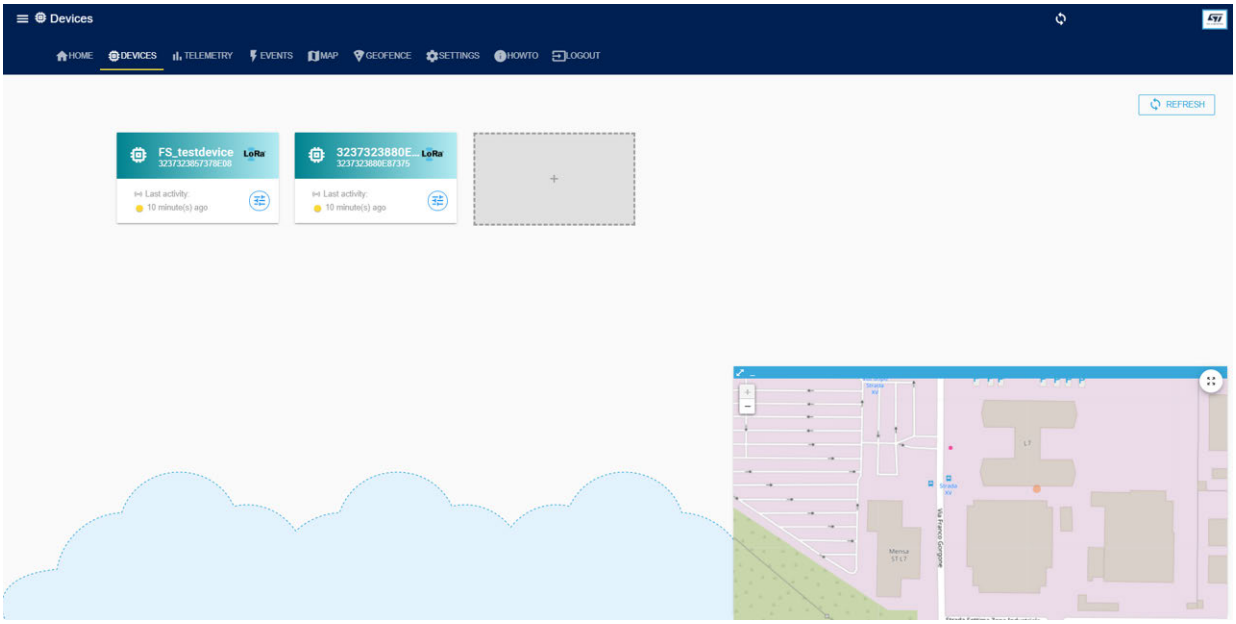


Figure 14. **ST Asset tracking dashboard - telemetry view**

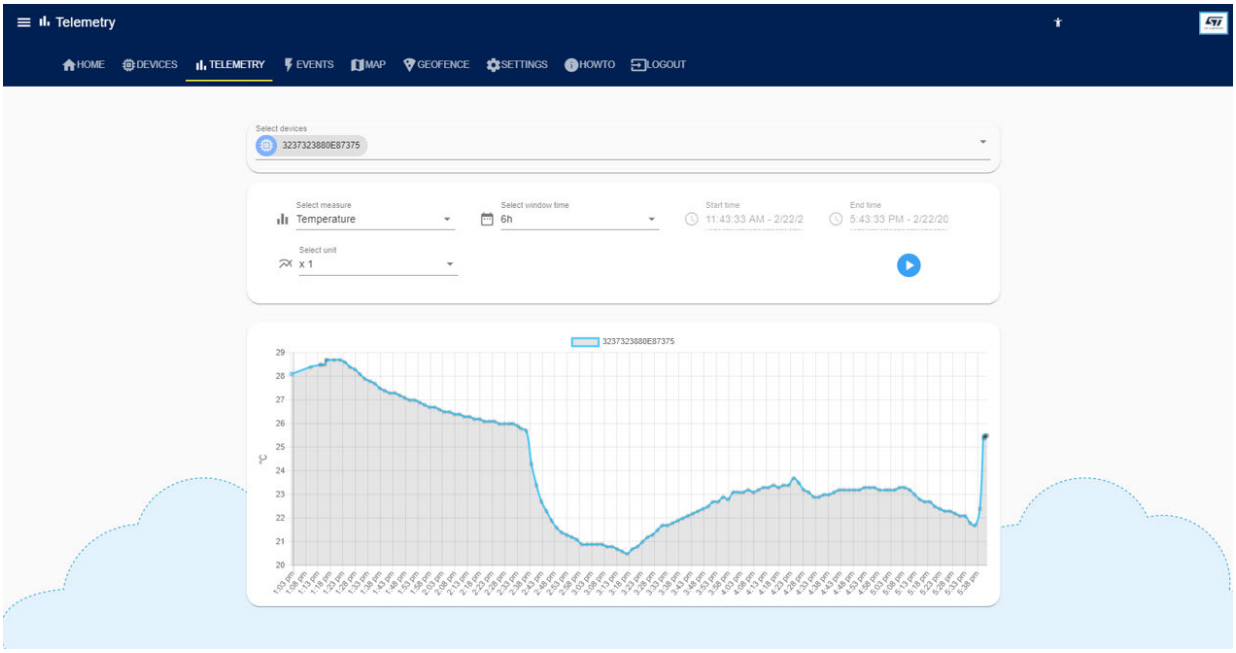
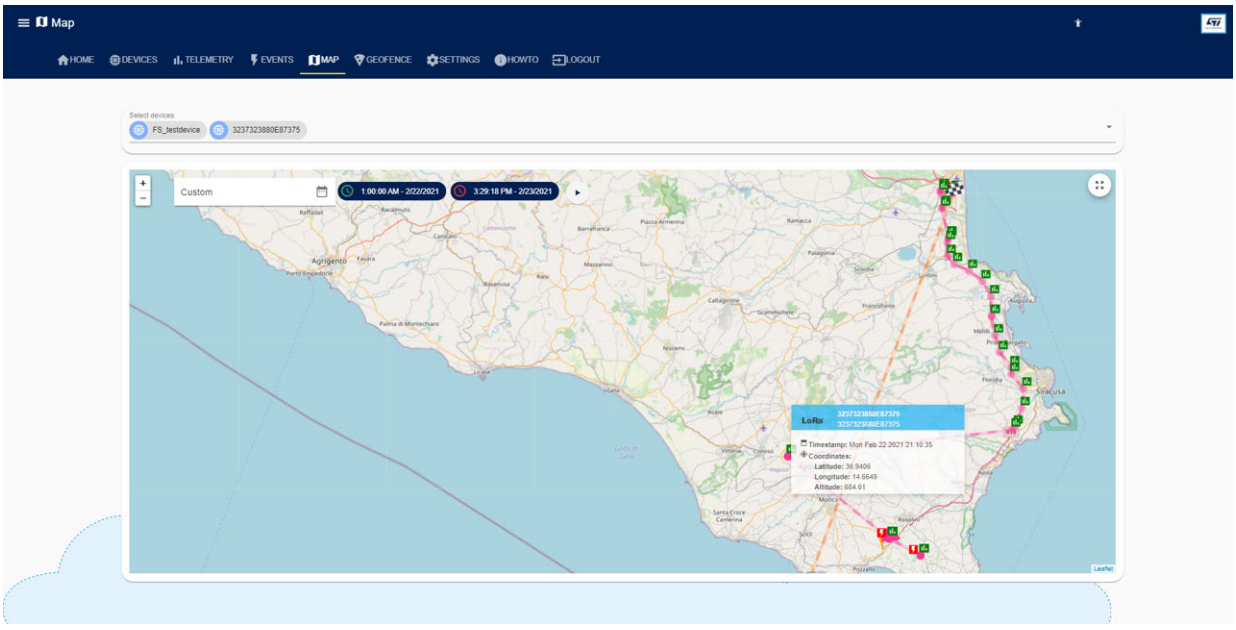


Figure 15. **ST Asset tracking dashboard - map view**



3 System setup guide

The FP-ATR-LORA1 firmware package can be used by the end user to build customized applications.

Hardware, firmware development environments and connection setup (described in the following sections) allows developing and transferring your own firmware to the hardware.

3.1 Hardware description

3.1.1 STEVAL-STRKT01 evaluation board

The STEVAL-STRKT01 LoRa® IoT tracker is designed and optimized to implement the latest technologies in IoT tracker applications such as asset, people and animal tracking as well as fleet management.

The evaluation board simplifies prototyping, evaluation and development of tracker innovative solutions. It comes with comprehensive software, firmware libraries, tools, battery, cables and plastic case.

Thanks to the STM32L072CZ embedded in the CMWX1ZZABZ-091 LoRa® module (by Murata), the STEVAL-STRKT01 allows acquiring position, managing geofence and data logging from Teseo-LIV3F GNSS module and monitoring motion (LIS2DW12) and environmental (HTS221 and LPS22HB) sensors.

The board also transmits and receives data, configurations and events to and from the cloud over a LoRaWAN™ network, or stores data locally in the M95M02-DR EEPROM.

The STEVAL-STRKT01 is a LiPo battery operated solution and implements low power strategies thanks to an enhanced power/battery management design, based on the STBC02 battery charger and the ST1PS01 step-down converter, to ensure long battery autonomy. The STUSB1600A addresses 5 V USB Type-C port management and offers high voltage protection pins.

Figure 16. STEVAL-STRKT01 evaluation package

3.1.2 B-L072Z-LRWAN1 Discovery kit

The B-L072Z-LRWAN1 Discovery kit embeds the CMWX1ZZABZ-091 LoRa® module (by Murata). It allows users to develop easily applications with the STM32L072CZ and the LoRa® RF connectivity in one single module.

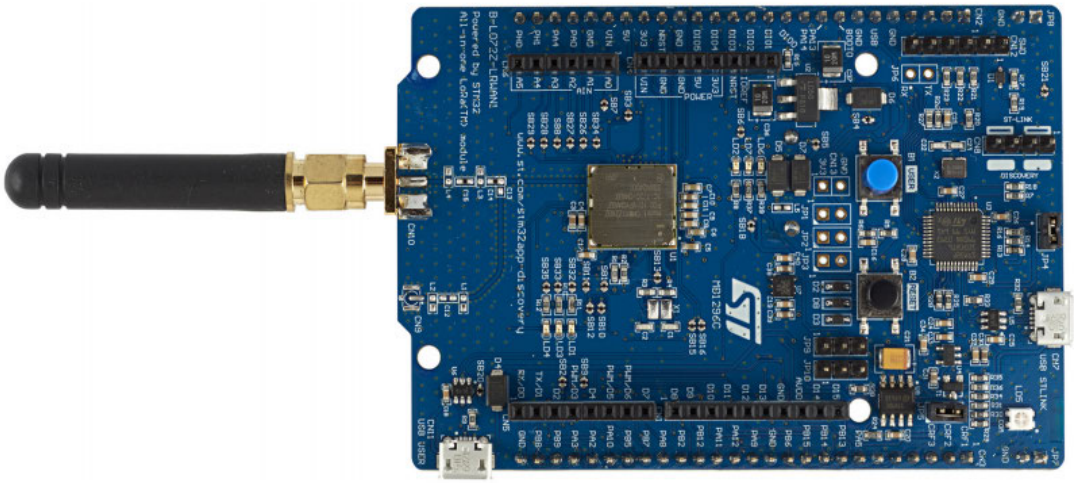
The B-L072Z-LRWAN1 Discovery kit has the full set of features available in the STM32L0 series and offers ultra-low-power and LoRa® RF features, including also LEDs, push buttons, antenna, Arduino™ Uno V3 connectors, USB 2.0 FS connector in Micro-B format.

The integrated ST-LINK/V2-1 provides an embedded in-circuit debugger and programmer for the STM32L0 MCUs.

The LoRaWAN™ stack is certified class A and C compliant and is available in the I-CUBE-LRWAN firmware package.

To help users setting up a complete node, the B-L072Z-LRWAN1 Discovery kit comes with the STM32 comprehensive free software libraries and examples available with the STM32Cube package, as well as a direct access to the Arm® Mbed Enabled™ resources at the http://mbed.org website.

Figure 17. B-L072Z-LRWAN1 Discovery kit



3.1.3 X-NUCLEO-GNSS1A1 expansion board

The X-NUCLEO-GNSS1A1 expansion board is based on the Teseo-LIV3F tiny GNSS module.

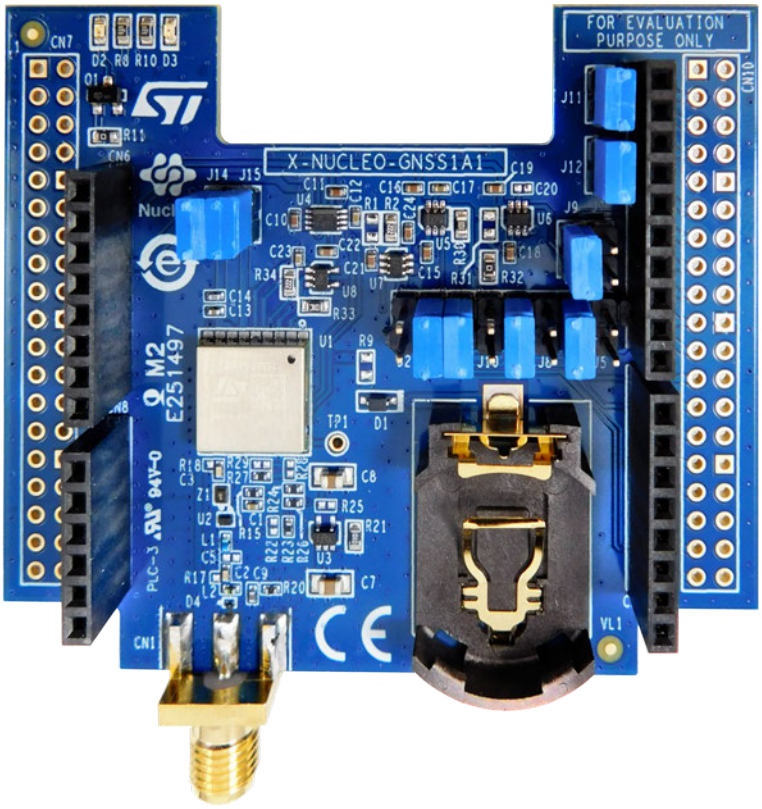
It represents an affordable, easy-to-use, global navigation satellite system (GNSS) module, embedding a TeseoIII single die standalone positioning receiver IC, usable in different configurations in your STM32 Nucleo project.

The Teseo-LIV3F is a compact (9.7x10.1 mm) module that provides superior accuracy thanks to the on-board 26 MHz temperature compensated crystal oscillator (TCXO) and a reduced time-to-first fix (TTFF) with its dedicated 32 KHz real-time clock (RTC) oscillator.

The Teseo-LIV3F module runs the GNSS firmware (X-CUBE-GNSS1) to perform all GNSS operations including acquisition, tracking, navigation and data output without external memory support.

The X-NUCLEO-GNSS1A1 expansion board is compatible with the Arduino™ UNO R3 connector and the ST morpho connector, so it can be plugged to the STM32 Nucleo development board and stacked with additional STM32 Nucleo expansion boards.

Figure 18. X-NUCLEO-GNSS1A1 expansion board



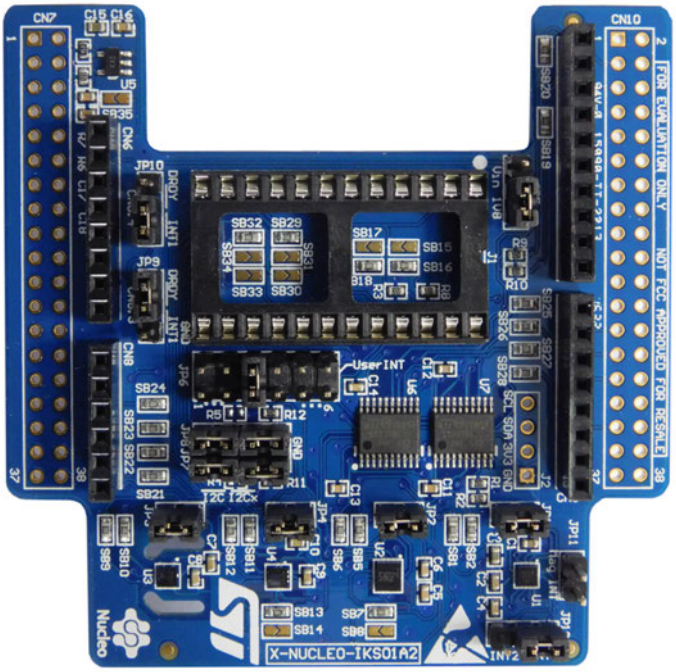
3.1.4 X-NUCLEO-IKS01A2 expansion board

The X-NUCLEO-IKS01A2 is a motion MEMS and environmental sensor expansion board for STM32 Nucleo.

It is compatible with the Arduino UNO R3 connector layout, and is designed around the LSM6DSL 3D accelerometer and 3D gyroscope, the LSM303AGR 3D accelerometer and 3D magnetometer, the HTS221 humidity and temperature sensor and the LPS22HB pressure sensor.

The X-NUCLEO-IKS01A2 interfaces with the STM32 microcontroller via the I²C pin, and it is possible to change the default I²C port.

Figure 19. X-NUCLEO-IKS01A2 MEMS and environmental sensor expansion board



3.2 Hardware setup

3.2.1 STEVAL-STRKT01 setup

The following hardware components are needed:

- One STM32 Nucleo board with USB cable (type A to mini B) for programming.
- One STEVAL-STRKT01 discovery kit (order code: STEVAL-STRKT01), containing:
  - one STEVAL-STRKT01 evaluation board;
  - a type-C M-M cable;
  - a USB A to type-C adapter;
  - a 5-pin flat cable to connect the STEVAL-STRKT01 to the STM32 Nucleo board for programming;
  - a battery;
  - an antenna.

3.2.2 B-L072Z-LRWAN1, X-NUCLEO-GNSS1A1 and X-NUCLEO-IKS01A2 setup

The following hardware components are needed:

1. One STM32 Discovery kit for LoRaWAN™ with STM32L0 (order code: B-L072Z-LRWAN1).
2. One Teseo-LIV3F expansion board (order code: X-NUCLEO-GNSS1A1).
3. One motion MEMS and environmental sensor expansion board (order code: X-NUCLEO-IKS01A2).
4. One USB type A to Micro-B USB cable to connect the STM32 Nucleo board to the PC.

The following jumpers settings need to be made on the boards:

1. On the B-L072Z-LRWAN1 discovery board, the solder bridges SB32 and SB35 must be unsoldered.
2. On the X-NUCLEO-GNSS1A1 expansion board, the following jumpers must be open: J3, J5, J6, J7, J8 and J10. The following jumpers must be closed: J2, J4, J9, J11, J12, J13, J14 and J15.
3. On the X-NUCLEO-IKS01A2 expansion board, the solder bridge SB25 must be unsoldered.

3.3 Software setup

The following software components are required for the setup of a suitable development environment to create applications for the STM32 Nucleo board with the sensor expansion board:

- FP-ATR-LORA1: an STM32Cube function pack dedicated to asset tracking applications development. The firmware and related documentation are available on www.st.com.
- Development tool-chain and Compiler. The STM32Cube expansion software supports the three following environments to select from:
  - IAR Embedded Workbench for ARM® toolchain + ST-LINK
  - RealView Microcontroller Development Kit (MDK-ARM-STR) toolchain + ST-LINK
  - System Workbench for STM32 (SW4STM32) + ST-LINK

3.4 System setup

To update the firmware running on the STEVAL-STRKT01 and the B-L072Z-LRWAN1 boards, set up the hardware as described below.

3.4.1 Update via ST-LINK/V2 in-circuit debugger/programmer

Step 1. Connect the 5-pin flat cable (male side) to STEVAL-STRKT01 CN501 connector as per Table 5

Step 2. Connect the 5-pin flat cable (female side) to ST-LINK/V2 pins 1-5 as per Table 5

Step 3.

Step 3a. Connect the battery and press SW400 button to power the board on (for 1.250 s at least)

or

Step 3b. Supply the STEVAL-STRKT01 via a Type-C USB cable (and a Type-C to Type-A adapter if needed)

Step 4. Connect the ST-LINK/V2 to a PC via a Type-A/mini B cable

Figure 20. **Hardware configuration for STEVAL-STRKT01 firmware update using an ST-LINK/V2 programmer**

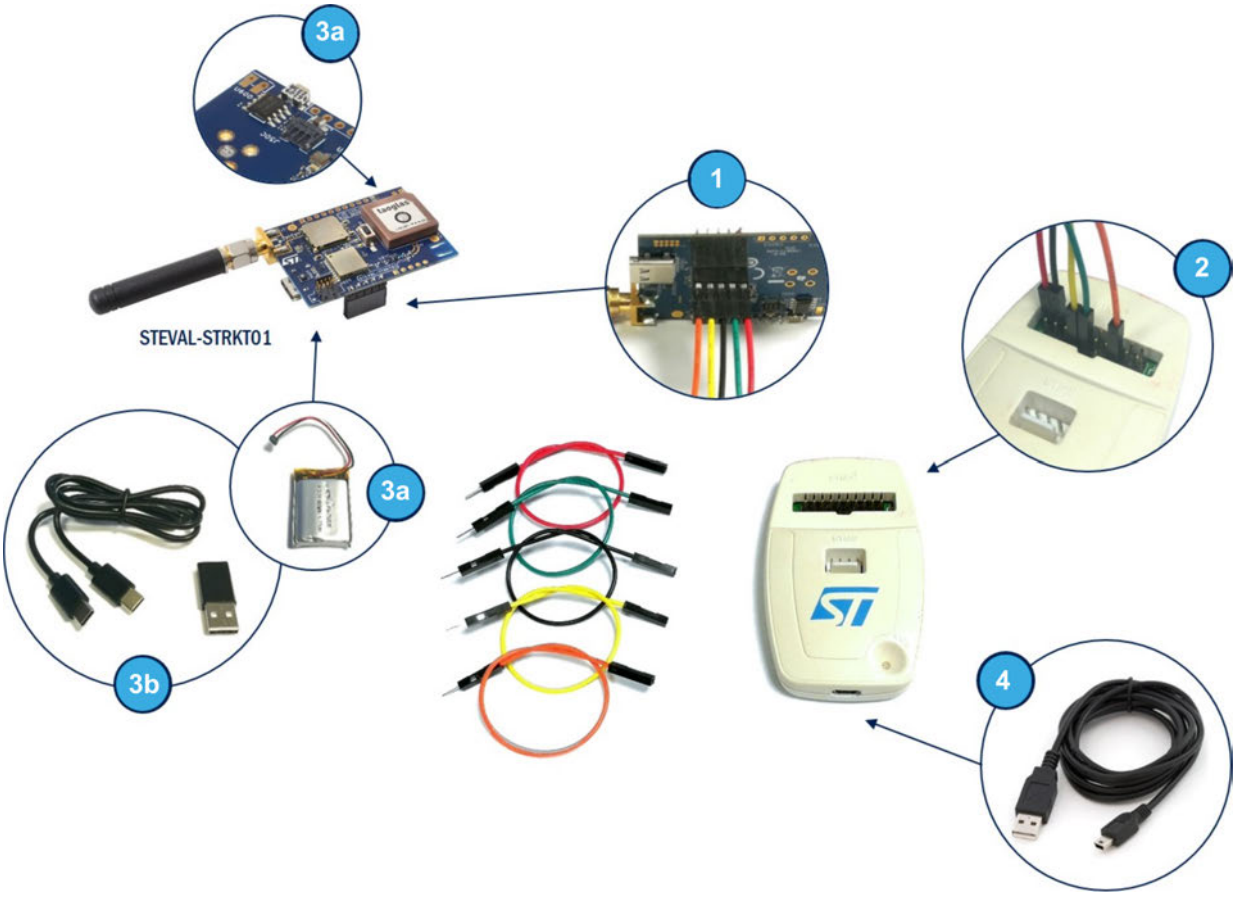
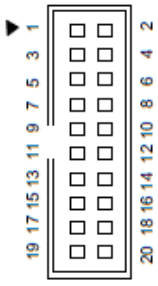


Table 5. **ST-LINK/V2 programmer and STEVAL-STRKT01 pinout**

ST-LINK/V2 connector (pin and label)		STEVAL-STRKT01 CN501 (pin and label)
	2, VAPP	5, D_VDD
	4, GND	3, GND
	7, TMS_SWDIO	2, SWD_SWDIO
	9, TCK_SWCLK	4, SWD_SWCLK
	15, NRST	1, nRESET

3.4.2 Update via ST-LINK/V3 in-circuit debugger/programmer

To perform the update via ST-LINK/V3 you first have to combine it with the adapter (MB1441 plus MB1440, shown in Figure 21. Hardware configuration for STEVAL-STRKT01 firmware update using an ST-LINK/V3 programmer).

Step 1. Connect the 5-pin flat cable (male side) to STEVAL-STRKT01 CN501 connector

Step 2. Connect the 5-pin flat cable (female side) to MB1440 CN6 connector pins 1-5 and leave pin 6 unconnected

Step 3.

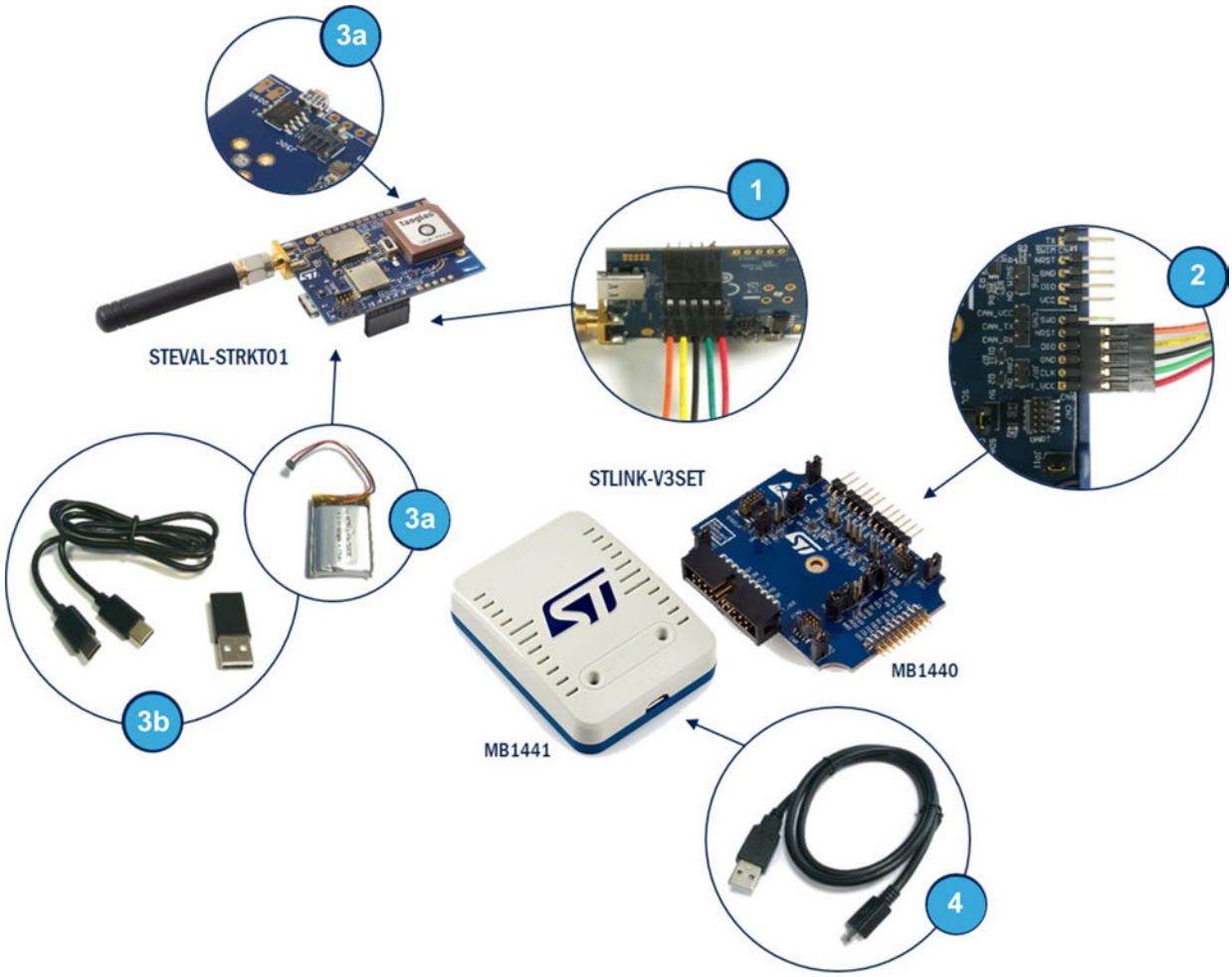
Step 3a. Connect the battery and press SW400 button to power the board on (for 1.250 s at least)

or

Step 3b. Supply the STEVAL-STRKT01 via a Type-C USB cable (and a Type-C to Type-A adapter if needed)

Step 4. Connect the ST-LINK/V3 to a PC via a Type-A/micro B cable

Figure 21. Hardware configuration for STEVAL-STRKT01 firmware update using an ST-LINK/V3 programmer



3.4.3 Update via STM32 Nucleo-64 on-board ST-LINK programmer

Step 1. Connect the 5-pin flat cable (male side) to STEVAL-STRKT01 CN501 connector

Step 2. Connect the 5-pin flat cable (female side) to the STM32 Nucleo SWD connector pins 1-5, leave pin 6 unconnected and remove CN2 jumpers

Step 3.

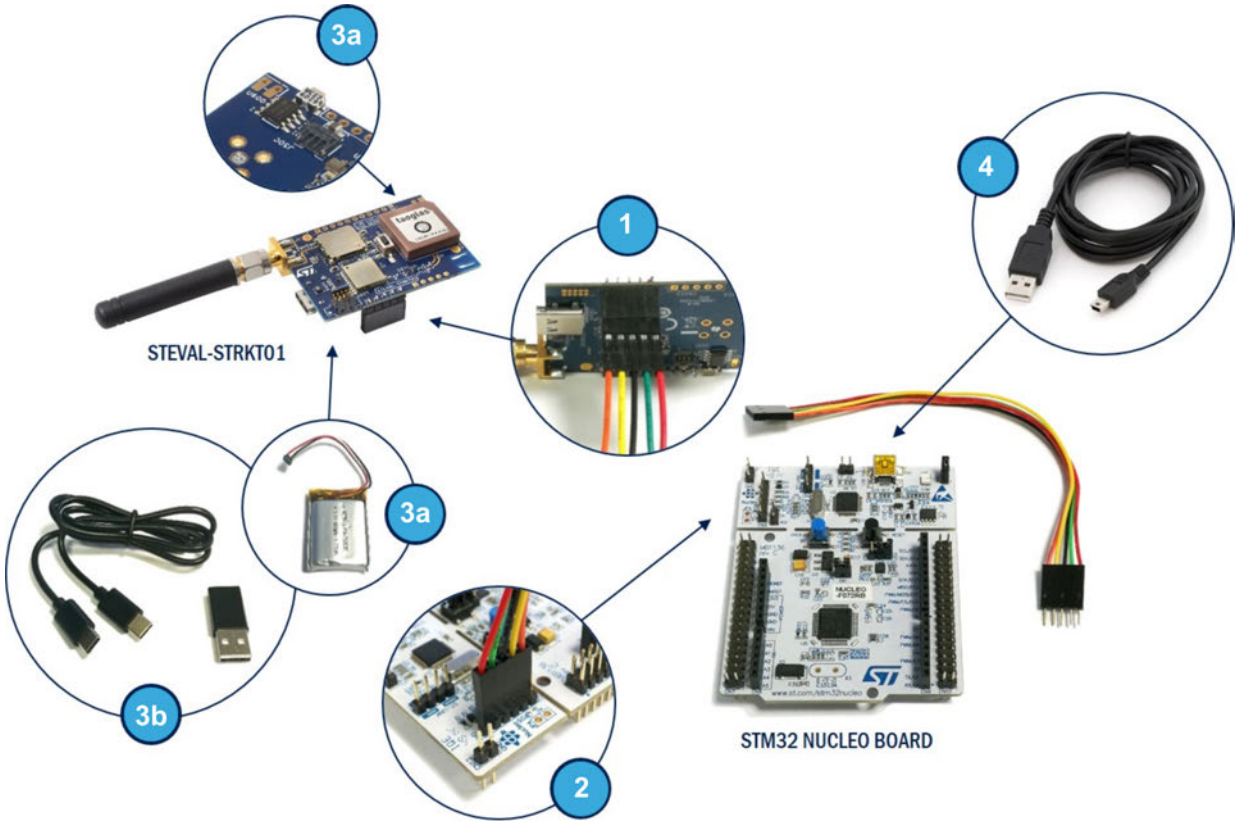
Step 3a. Connect the battery and press SW400 button to power the board on (for 1.250 s at least)

or

Step 3b. Supply the STEVAL-STRKT01 via a Type-C USB cable (and a Type-C to Type-A adapter if needed)

Step 4. Connect the STM32 Nucleo to a PC via a Type-A/mini B cable

Figure 22. **Hardware configuration for STEVAL-STRKT01 firmware update using STM32 Nucleo-64 on-board ST-LINK programmer**



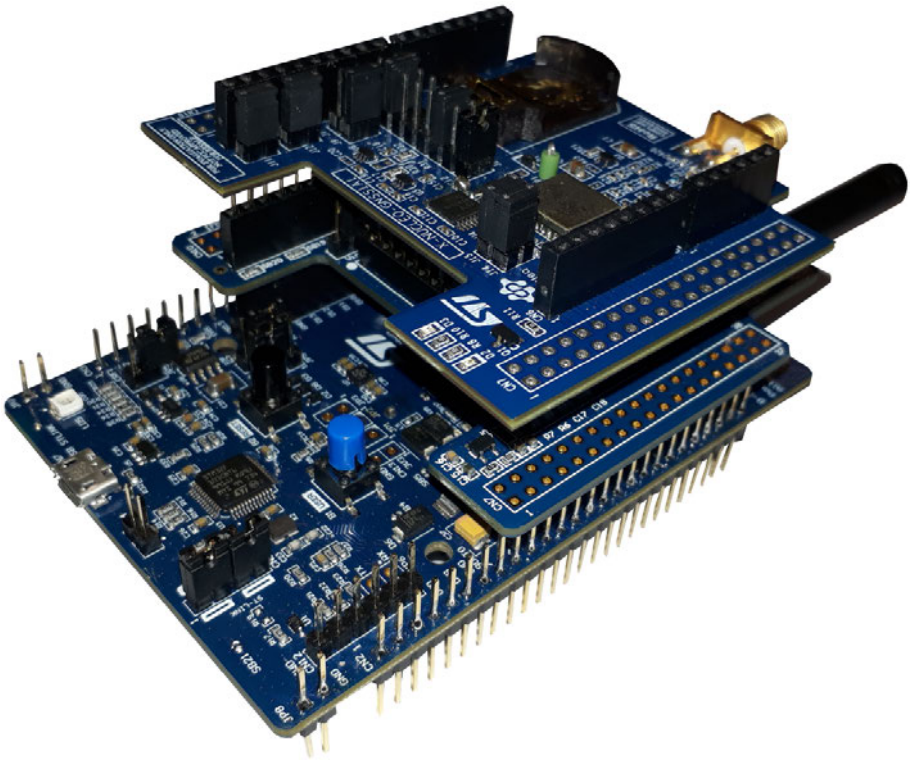
3.4.4 B-L072Z-LRWAN1, X-NUCLEO-GNSS1A1, X-NUCLEO-IKS01A2 system setup

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer.

The developer can download the ST-LINK/V2-1 USB driver by looking at the STSW-LINK009 software on www.st.com.

The X-NUCLEO-GNSS1A1 and the X-NUCLEO-IKS01A2 expansion boards can be easily connected to the STM32 Nucleo through the Arduino UNO R3 extension connector. The board interfaces with the external STM32 microcontroller on STM32 Nucleo using inter-integrated circuit (I²C) transport layer (for the X-NUCLEO-GNSS1A1) and universal asynchronous receiver-transmitter (UART) serial interface (for X-NUCLEO-GNSS1A1).

Figure 23. **B-L072Z-LRWAN1, X-NUCLEO-GNSS1A1 and X-NUCLEO-IKS01A2 stack**



Appendix A References

1. UM2115: "Discovery kit for LoRaWAN™, Sigfox™, and LPWAN protocols with STM32L0" on www.st.com
2. UM2327: "Getting started with the X-NUCLEO-GNSS1A1 expansion board based on Teseo-LIV3F tiny GNSS module for STM32 Nucleo" on www.st.com
3. UM2121: "Getting started with the X-NUCLEO-IKS01A2 motion MEMS and environmental sensor expansion board for STM32 Nucleo" on www.st.com
4. https://www.multitech.com/brands/multiconnect-conduit
5. https://www.loriot.io
6. https://cayenne.mydevices.com

Revision history

Table 6. **Document revision history**

Date	Version	Changes
08-Oct-2018	1	Initial release.
20-Feb-2019	2	Updated Section 3.1 Overview, Section 3.3 Folder structure, Section 3.5 Sample application description, Section 3.6 Configuration and registration, Section 3.6.3 Gateway setup, Section 3.6.4.1 Loriot account setup and Section 3.6.5.1 Cayenne account setup. Added Section 3.5.2 LoRa packet format, Section 3.5.3 Storing and retrieving sensor data, Section 3.6.1 B-L072Z-LRWAN1 Discovery kit plus STM32 Nucleo expansion board setup, Section 3.6.2 STEVAL-STRKT01 setup, Section 3.6.5.2 TagoIO account setup, Section 4.2.2 STEVAL-STRKT01 setup and Section 4.4.2 STEVAL-STRKT01 system setup.
04-Jun-2019	3	Updated Section 3.1 Overview, Section 3.2 Architecture, Section 3.3 Folder structure, Section 3.5 Sample application description, Section 3.5.2 Sample application state machine, Section 3.5.3 LoRa packet format, Section 3.6 Configuration and registration, Section 3.6.4 Gateway setup, Section 3.6.5.1 Loriot account setup and Section 4 System setup guide. Added Section 3.5.1 Serial port communication, Section 3.5.2.1 Enabling confirmed LoRaMAC frame, Section 3.5.2.2 Data download over the air and Section 3.6.1 LoRaWAN keys.
09-Dec-2019	4	Updated Table 2. FP-ATR-LORA1 command list, Section 2.5.2.1 Enabling confirmed LoRaMAC frame, Section 2.5.2.2 Data download over the air, Section 2.6.6.2 TagoIO account setup. Removed Section 2. What is STM32Cube? and replaced it by a link on the cover page. Added Section 3.4.1 Update via ST-LINK/V2 in-circuit debugger/programmer, Section 3.4.2 Update via ST-LINK/V3 in-circuit debugger/programmer and Section 3.4.3 Update via STM32 Nucleo-64 on-board ST-LINK programmer.
01-Mar-2021	5	Updated Section 2.6 Configuration and registration, Section 2.6.1 LoRaWAN keys, Section 2.6.4 Gateway setup, Section 2.6.5 STEVAL-STRKT01 registration on LoRaWAN network server and Section 2.6.6 Application server setup.

Contents

List of tables

List of figures