6LoWPAN solution for smart metering application using the X-NUCLEO-S2868A2 expansion board and the NUCLEO-G070RB development board

Introduction

This document describes how to use the 6LoWPAN application in smart metering.

The X-CUBE-SUBG1 is the software package on top of which we built a smart metering application. This application follows the MSEDCL (Maharashtra State Electricity Distribution Co. Ltd) 6LoWPAN specification. The solution is interoperable with other competitor modules.

This document also explains how to make a device as a node and a BR using an AT command set, as well as how the node joins the mesh network by sending beacon and ping requests.

The document also covers the flash memory partition and factory default settings. The application is based on the contiki3x operating system. MAC and PHY layers are compliant with the IEEE 802.15.4g standard.

The packets are encrypted/decrypted with AES-128 bit encryption/decryption. You can periodically change them through the ROOT using a commissioning command.

All the devices can be either a node or a root. You can set it during the provisioning.

RF and network parameters are saved in the flash memory of the MCU. During the configuration, the meter is whitelisted with the root. Hence, as soon as the meter is powered on, it immediately establishes a mesh network and connects with the ROOT.

In case of power failure, the system restores the last known network parameters from the MCU flash memory and immediately reestablishes the network. The joining process is interoperable as per RPL standards.

You can configure the RF and network parameters either through the RF commissioning protocol or through AT command sets.

The application supports UDP ports for data communications.

Note:

1. *The firmware explained in this document is developed and tested using the NUCLEO-G070RB development board with the X-NUCLEO-S2868A2 expansion board.*
2. *This demo firmware can be easily ported to other STM32 Nucleo development boards or radio boards by changing the BSP layer.*
3. *Node/router/BR/root is used for the combination of the STM32 Nucleo development board and the S2-LP-based RF expansion board when used in a connected network.*

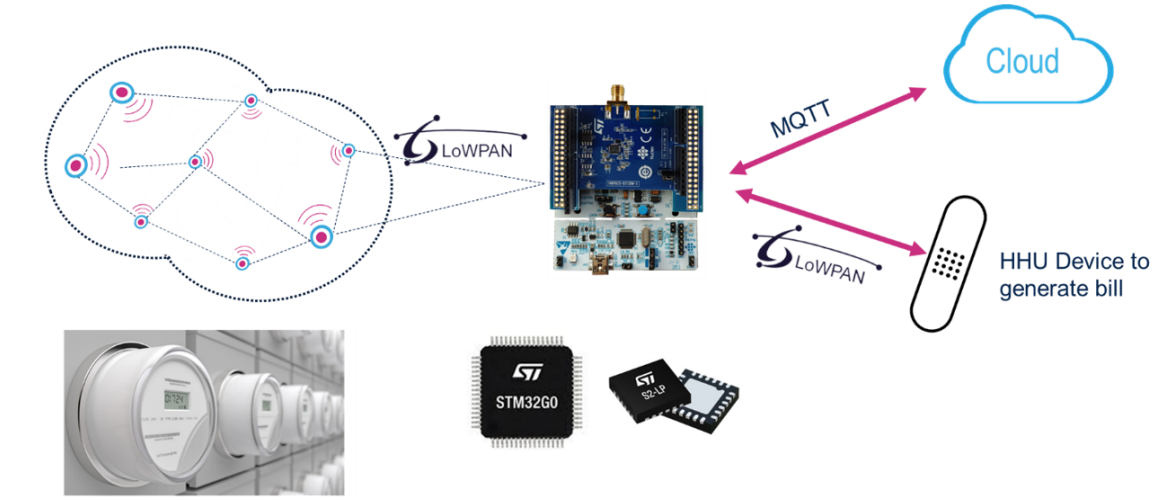
1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
BSP	Boot support package
EEPROM	Electrically erasable programmable read-only memory
GHz	Giga Hertz
HAL	Hardware abstraction layer
LED	Light emitting diode
MCU	Microcontroller unit
RF	Radio frequency communication
SPI	Serial peripheral interface
USB	Universal serial bus
BR/ROOT	Border router/PAN coordinator
6LoWPAN	IPv6 over low-power wireless personal area networks
AES	Advanced encryption standard
MAC	Medium access control
PAN	Personal area network
HHU	Hand-hold unit

2 Architecture of the smart metering application

Figure 1. Smart metering application architecture



The architecture of the smart metering application includes three main sections:

1. 6LoWPAN mesh network and meters;
2. Root device/concentrator;
3. HHU device/cloud application.

All the meters work as nodes/routers and ROOT/border routers. The border router unit has to be installed at the security gate/central location of the building. All the building meters connect to the ROOT using the Contiki-based 6LoWPAN mesh network technology.

For meter reading, there can be the following options:

1. The ROOT is connected to the cloud through the MQTT server. Hence, the electricity board can fetch all the data from the cloud and the bill can be sent via SMS/mail/online using digital technologies.
2. If there is no cloud connectivity, a person visits the DCU location, connects the hand-hold unit (HHU) device to the ROOT. Then he/she fetches and generates the data of all the meters connected to that ROOT. With this option, there is no compromise between the mesh network technology and the data security in the network as the data is encrypted with the AES CCM-32 128-bit encryption key. You can periodically change the keys through a DCU using a commissioning command.

Note: Currently, the ST application supports both metering and ROOT functionality.

3 X-CUBE-SUBG1 software package modified for the smart metering application

3.1 Overview

X-CUBE-SUBG1 is a software package that expands the functionality provided by STM32Cube. The software runs on the STM32 and includes drivers that recognize the Sub-1 GHz RF communication for the S2-LP. The expansion is built on STM32Cube software technology to ease portability across different STM32 microcontrollers.

The software comes with P2P communication protocol and 6LoWPAN applications (Udp-sender, Udp-receiver, and Sniffer) running on an X-NUCLEO-S2868A2 when connected to a NUCLEO-G070RB development board.

3.2 Architecture

This software fully complies with the architecture of the STM32Cube and expands it to enable the development of applications using the X-NUCLEO-S2868A2 that hosts the S2-LP device.

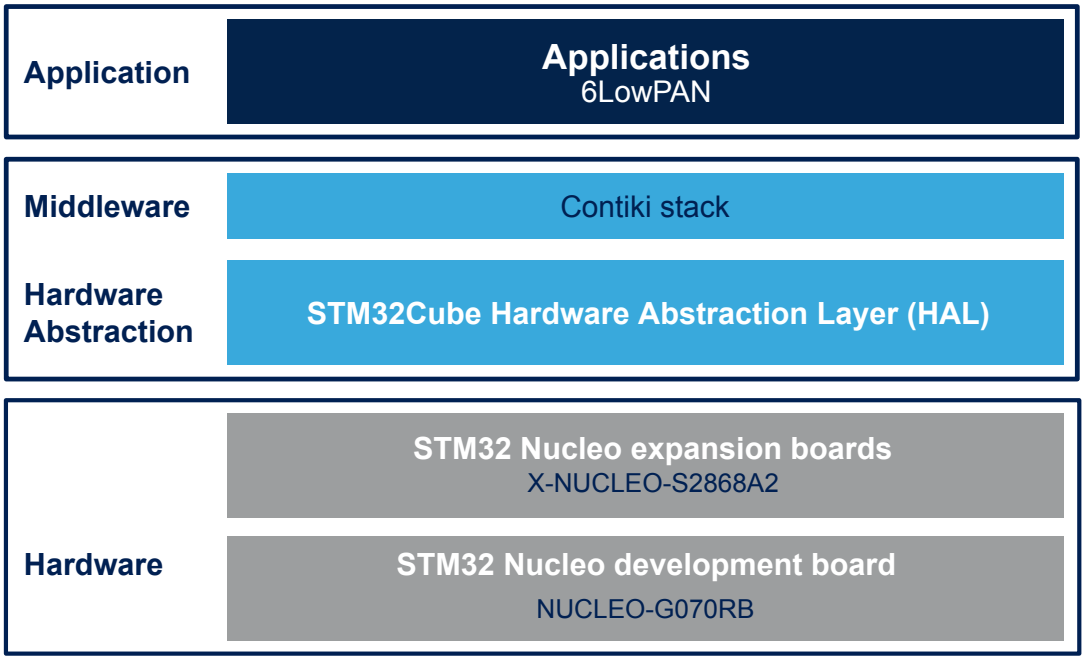
The software is based on the STM32CubeHAL, which is the hardware abstraction layer for the STM32 microcontroller. The package extends STM32Cube by providing a board support package (BSP) for the S2-LP expansion board and some example firmware for P2P and 6LoWPAN communication.

The software layers used by the application software to access and use the S2-LP expansion board are:

- STM32Cube HAL layer: provides a generic multi-instance simple set of application programming interfaces (APIs) to interact with the upper layers (application, libraries, and stacks). It consists of generic and extension APIs. It is directly built around a generic architecture and allows the layers that are built upon, such as the middleware layer, to implement their functionalities without dependencies on the specific hardware configuration for a given microcontroller unit (MCU). This structure improves the library code reusability and guarantees an easy portability across other devices.
- Board support package (BSP) layer: the software package needs to support the peripherals on the NUCLEO-G070RB apart from the MCU. A limited set of APIs provides a programming interface for certain board-specific peripherals, such as the LED, the user button, etc. The BSP firmware layer of the X-NUCLEO-S2868A2 contains a set of APIs related to the hardware components, which includes:
  - Component: the driver related to the external device on the board and not related to the STM32.

    The S2-LP component driver provides specific APIs and can be ported and used on any board.
  - BSP driver: enables the component driver to be linked to a specific board and provides a set of user-friendly APIs.
- Middleware: contains the contiki3x-based open-source firmware and some files, which require to port this 6LoWPAN stack on the ST platform.
- Application layer: provides an example of the smart metering application based on the 6LoWPAN solution that uses the S2-LP sub-1GHz expansion board and the NUCLEO-G070RB development board. The application is interoperable with other competitor modules.

Figure 2. **X-CUBE-SUBG1 modified software architecture**

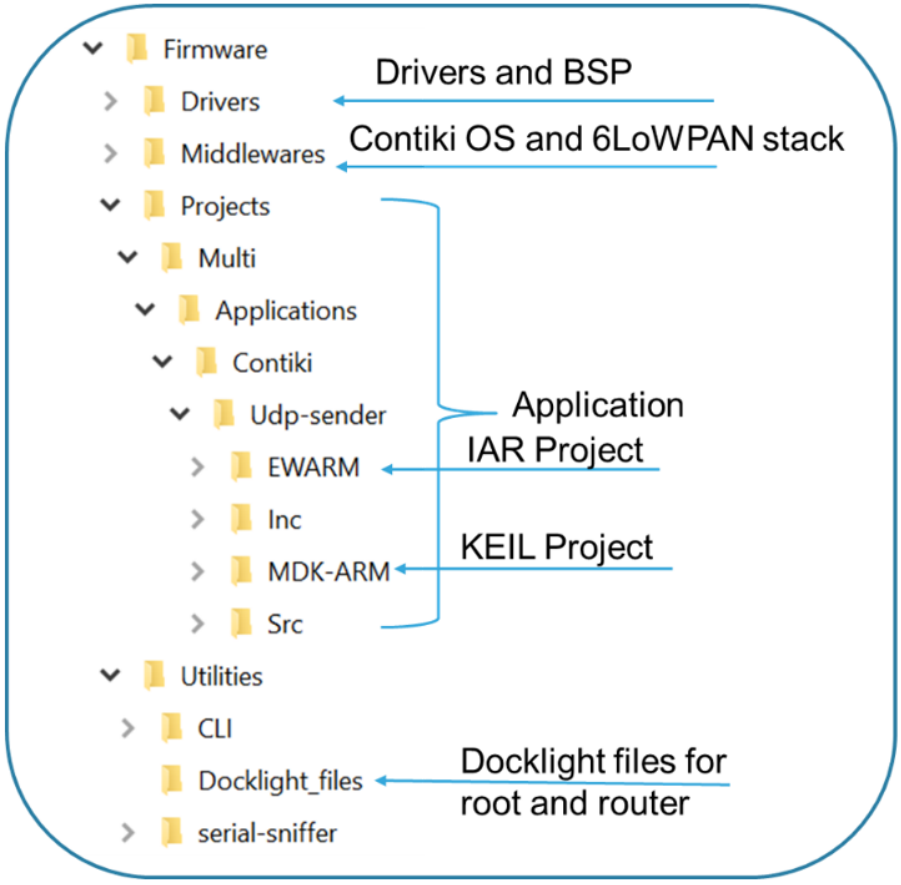


3.3 Folder structure

The software package includes the following folders:

- Drivers: contains the HAL drivers, the board-specific drivers for each supported board or hardware platform, including the on-board components ones and the CMSIS layer, which is a vendor-independent hardware abstraction layer for the Arm®Cortex®-M processor series.
- Middlewares: contains the Contiki 6LoWPAN stack and files required to port the Contiki OS on the ST platform.
- Projects: contains a 6LoWPAN application, which can be used in the e-metering application. The firmware supports two development environments, IAR Embedded Workbench for ARM and RealView Microcontroller Development Kit (MDK-ARM).
- Utilities: contains the docklight file required to configure the board as a node (router) or a BR (root).

Figure 3. **Application folder structure**



3.4 APIs

A compiled HTML file located inside the “Documentation” folder of the software package contains detailed technical information about the available APIs. The file describes all the functions and parameters.

4 System setup guide

4.1 Hardware description

This section describes the hardware components required to develop a 6LoWPAN based e-metering application.

4.1.1 STM32 Nucleo platform (NUCLEO-G070RB)

The STM32 Nucleo development boards provide an affordable and flexible way to try out new ideas and build prototypes with any STM32 microcontroller series.

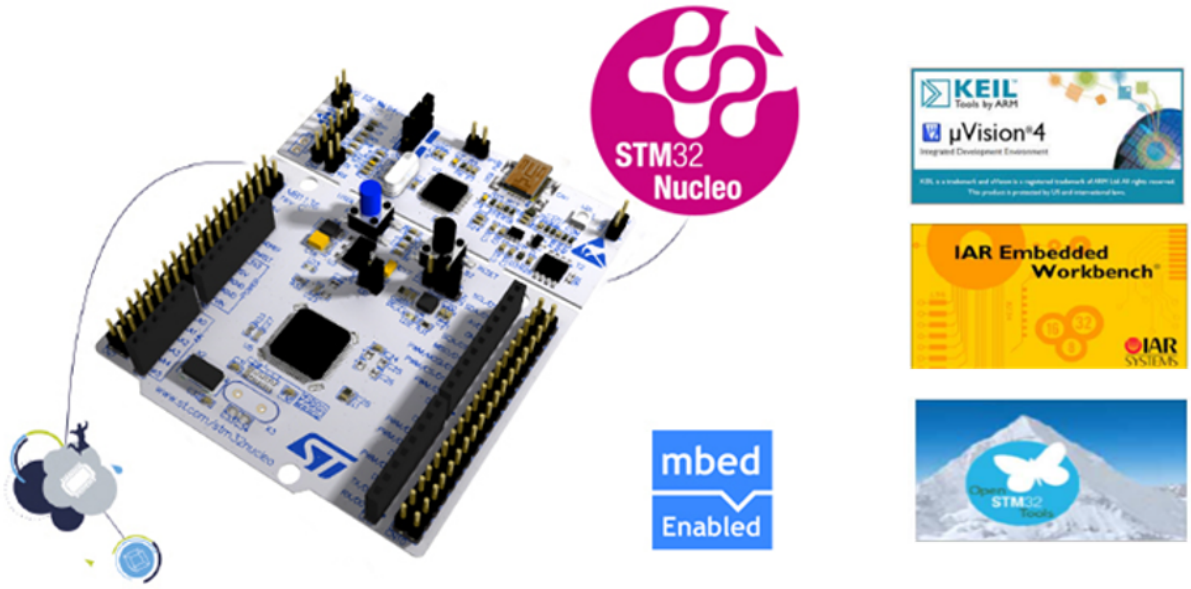
The Arduino connectivity and ST Morpho headers allow expanding the functionality of the STM32 Nucleo open development platform with a wide choice of specific expansion boards.

The STM32 Nucleo development boards do not require any separate probe as they integrate the ST-LINK/V2-1 debugger/programmer. The board comes with the STM32 comprehensive software HAL library together with several software examples.

The NUCLEO-G070RB features:

- Arm® Cortex®-M0+ core
- An up to 64 MHz frequency
- Up to 128 kbytes of flash memory
- 36 kbytes of SRAM (32 kbytes with hardware parity check)
- 1.7 to 3.6 V voltage range
- Two SPIs, four USARTs with master/slave roles, and other peripherals

Figure 4. NUCLEO-G070RB development board



4.1.2 X-NUCLEO-S2868A2 expansion board

The X-NUCLEO-S2868A2 expansion board is based on the S2-LP ultra-low power RF transceiver and operates in the 868 MHz ISM frequency band.

The X-NUCLEO-S2868A2 interfaces with the STM32 Nucleo microcontroller via SPI connections and GPIO pins. You can change some of the GPIOs by mounting or removing the resistors.

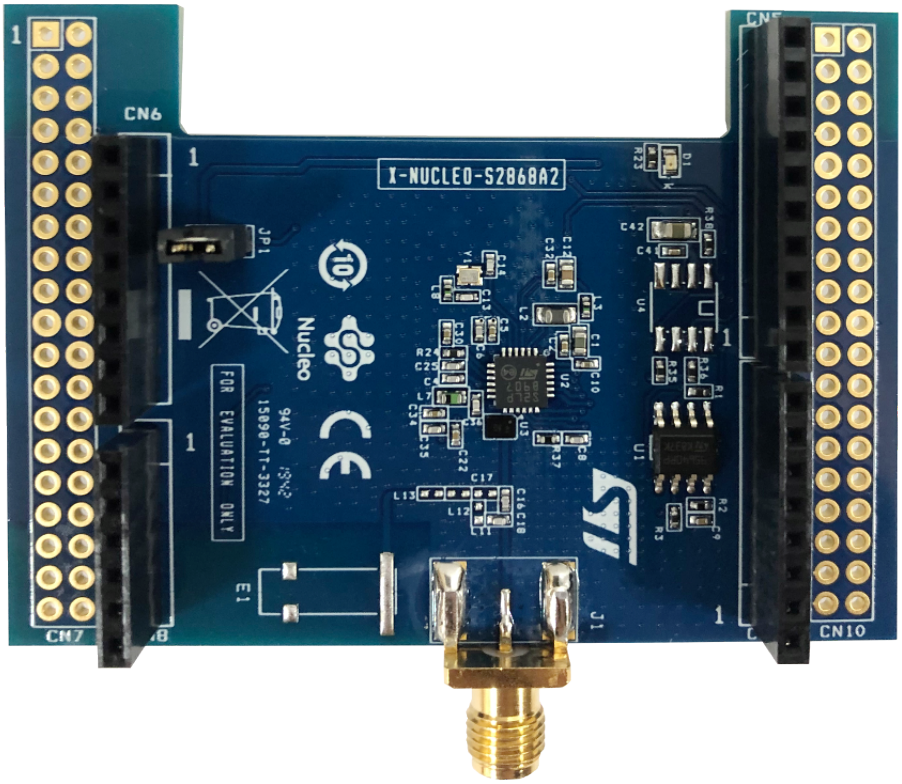
The expansion board is compatible with ST morpho and Arduino UNO R3 connectors.

The expansion board features:

- Based on S2-LP radio
- S2-LP narrow band ultra-low power sub-1 GHz transceiver tuned for 860 - 940 MHz frequency band
- Programmable RF output power up to +16 dBm

- Modulation schemes: 2-FSK, 2-GFSK, 4-FSK, 4-GFSK, OOK and ASK
- Air data rate from 0.1 to 500 kbps
- Ultra-low power consumption: 7 mA RX and 10 mA TX at +10 dBm
- IEEE 802.15.4g hardware packet support with whitening, FEC, CRC and dual SYNC word detection
- RX and TX 128 byte FIFO buffers
- Support to wireless M-Bus
- Excellent performance of receiver sensitivity (up to -130 dBm)
- Automatic acknowledgement, retransmission and timeout protocol engine
- Compatible with STM32 Nucleo boards
- Compatible with Arduino UNO R3 connectors
- BALF-SPI2-01D3 IPD balun for matching network and harmonics filter
- Sigfox compatible
- Sample firmware for P2P communication
- 6LoWPAN compatible thanks to STM32Cube
- RoHS and WEEE compliant

Figure 5. X-NUCLEO-S2868A2 expansion board



4.1.3 GPIO configuration

4.1.3.1 SPI communication

In the firmware, the S2-LP expansion board is connected to the STM32 through the SPI interface.

Table 2. Pin configuration

SPI1	SCK	MOSI	MISO	CS	SDN	S2LP_INT (GPIO3)
MCU pin	PB3	PA7	PA6	PA1	PA8	PB8

4.1.3.2 *UART2 communication*

You can connect the UART2 interface of the STM32 to:

- the ST-LINK/V2-1 MCU;
- the ST morpho connector (CN10 pin 6 and pin 34).

Table 3. UART configuration

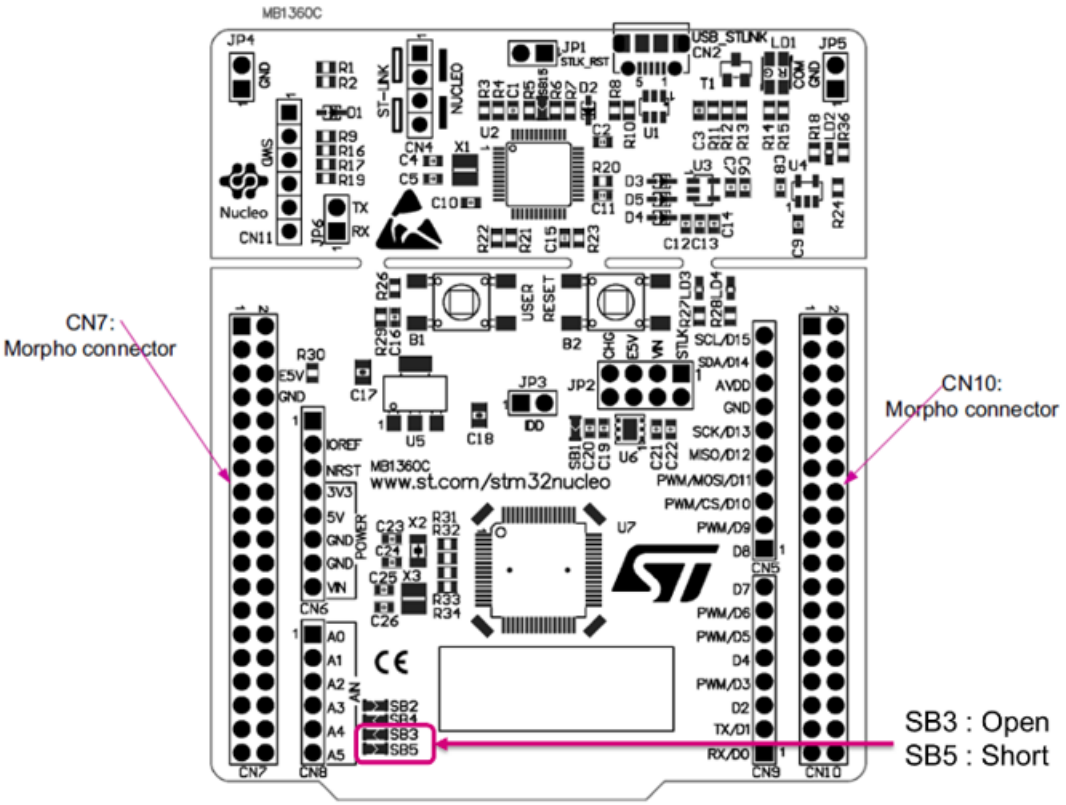
UART2	Rx	Tx
MCU pin	PA3	PA2

4.1.4 Hardware modifications

The following modifications are required in the NUCLEO-G070RB development board:

- SB3: open
- SB5: short

Figure 6. Modifications in the NUCLEO-G070RB development board



4.2 Software description

To set up a suitable development environment in order to create new applications for the NUCLEO-G070RB development board equipped with the S2-LP expansion board, you need:

- X-CUBE-SUBG1 software expansion for STM32Cube dedicated to the S2-LP application development.
- Docklight software, which is a testing, analysis, and simulation tool for serial communication protocols via VCOM.
- Wireshark software, which is a sniffer application to view the RF packet on the PC.
- Development tool-chains and compilers:
  - IAR Embedded Workbench for Arm® (EWARM) toolchain + ST-LINK or
  - RealView Microcontroller Development Kit (MDK-ARM) toolchain + ST-LINK

4.3 Network configuration

4.3.1 Hardware configuration

The following hardware components are required per node/root/router/BR:

1. One STM32 Nucleo development board (suggested order code: NUCLEO-G070RB)
2. One S2-LP expansion board (order code: X-NUCLEO-S2868A2)
3. One USB type A to Micro-B USB cable to connect the STM32 Nucleo to the PC

You need two nodes to replicate this demo. You can create a board as a sniffer to capture the RF packet.

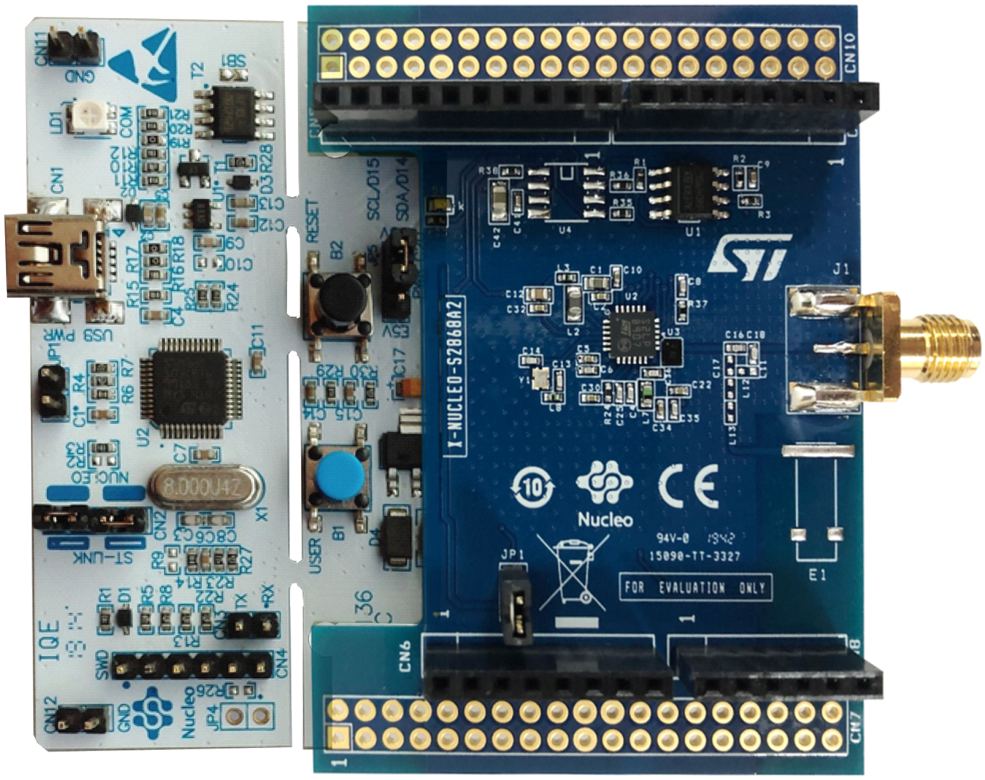
4.3.2 Software configuration

To set up the expansion board and the development board, follow the procedure below.

Step 1. Check whether a jumper is connected on the JP1 connector.
This jumper provides the required voltage to the devices on the board.

Step 2. Stack the X-NUCLEO-S2868A2 with the NUCLEO-G070RB as shown in the figure below.

Figure 7. X-NUCLEO-S2868A2 stacked on NUCLEO-G070RB



Step 3. Power the NUCLEO-G070RB using the Micro-B USB cable.

Step 4. Program the firmware in the STM32 on the NUCLEO-G070RB using the firmware example provided under “Projects\Multi\Applications\Contiki\Udp-sender”.

Step 5. Reset the board MCU using the [**Reset**] button on the NUCLEO-G070RB.

Step 6. Flash the same firmware on the two boards to make one as a ROOT/BR and the other as a node/router.

Step 7. Make a device as a node or a BR following these steps.

Step 7a. Configure the device as a node (router):

- Connect the board to the ST-LINK and full erase the flash memory (first time only).
- Flash the firmware or the binary file.
- Open the docklight file, select the correct COM port number, and set the baud rate (9600).
- Configure the channel number from 0 to 9 (default is 2).
- Set eight bytes of the device MAC address (0x3C, 0xC1, 0xF6, 0x01, 0x01, 0x5F, 0x22, 0x10).
- Select [**Save**] and [**Exit**].

Step 7b. Configure the device as a BR (ROOT):

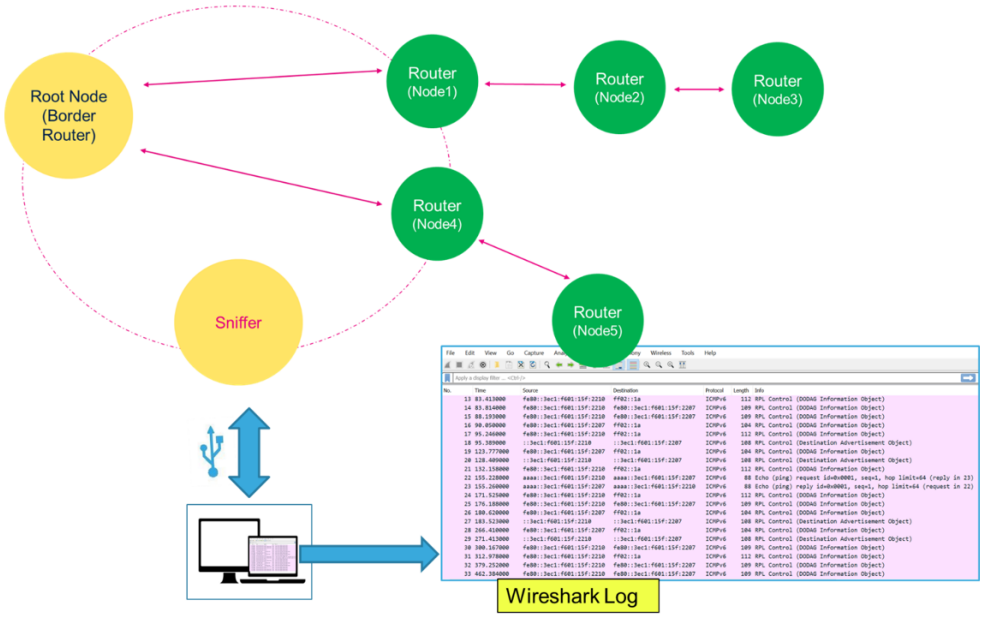
- Connect the board to the ST-LINK and full erase the flash memory (first time only).
- Flash the firmware or the binary file.
- Open the docklight file, select the correct COM port number, and set the baud rate (9600).
- Configure the channel number from 0 to 9 (default is 2).
- Set eight bytes of the device MAC address (0x3C, 0xC1, 0xF6, 0x01, 0x01, 0x5F, 0x22, 0x07).
- Set PAN ID (any 2 bytes, default is 0xFFFF).
- Select [**Device Type ROOT**] and make a device as BR/ROOT.
- Whitelist the node/router address.
- Select [**Save**] and [**Exit**].

*Note: Fully erase of the memory is required only the first time. If you want to use factory default settings, select [**Restore**]>[**Save**]>[**Exit**]. Then, the system reboots with the default configuration. You have to configure the channel number, MAC ID, and device type to make the device a node/router or BR/ROOT. After changing the configuration parameters, press [**Save**] and [**Exit**] to make the changed parameters effective.*

The evaluation kit is ready to use.

To view the 6LoWPAN packet, make a board as a sniffer and capture all the packets on the wireshark software. Ensure the sniffer, node, and root are all on the same channel number.

Figure 8. Wireshark and sniffer



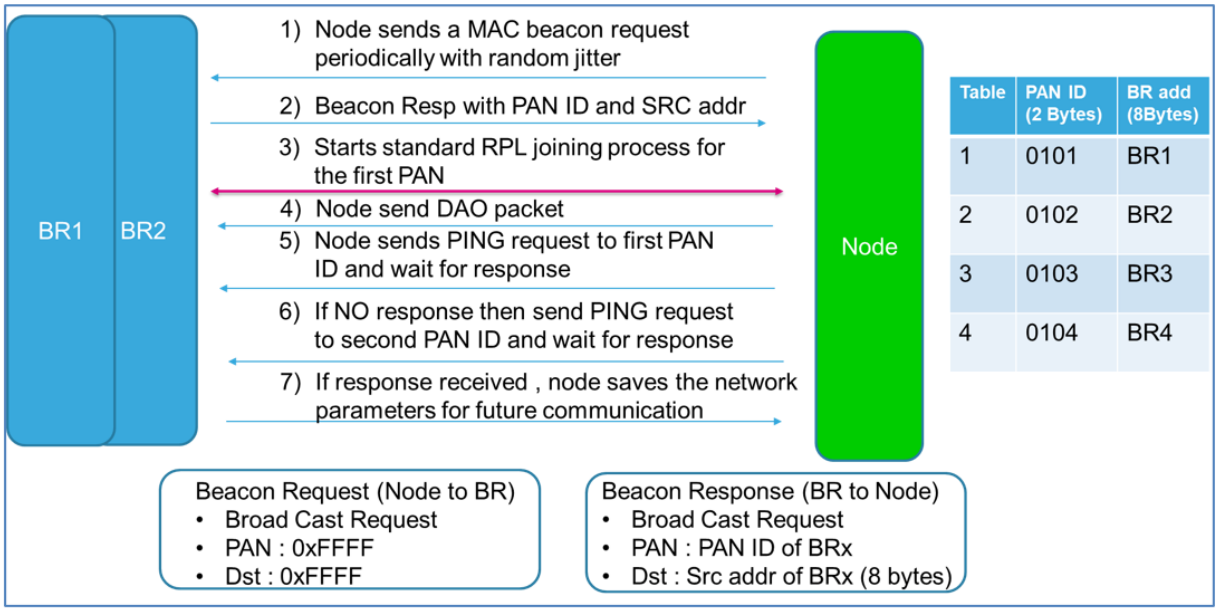
4.3.3 Node joining process

At power on, a node can be in commissioned state or in a non-commissioned state. If the node is not in a commissioned state, the device is in the default node (router) state with the factory default network parameters. Refer to Section 6.3 for the network default parameters.

4.3.3.1 Non-commission node joining process

1. After power on, if the device is in a non-commissioned state, it first sends IEEE 802.15.4G MAC beacon request packet.
2. The node periodically sends this beacon packet with some random jitter until the device has not received the beacon response packet.
3. The beacon request is a broadcast request. With this request, the node uses the default PAN ID (0xFFFF) and the default destination address (0xFFFF).
4. If there are multiple border router/ROOT devices, each BR/ROOT device, which is in the direct vicinity of the node, sends the beacon response packet.
5. BR/ROOT device sends the last eight bytes of the MAC address and PAN ID in the beacon response packet.
6. On receiving this response packet, the node stores the PAN ID and destination address of BR/ROOT device in its data structure used in the ping request.
7. The standard RPL joining process starts with the first PAN ID.
8. After sending the DAO packet, the node sends a ping request to a BR/ROOT node.
9. If a node address is whitelisted in the BR/ROOT, the BR/ROOT node sends the ping reply. Otherwise no response arrives from the BR/ROOT device.
10. If the node does not receive any response from the BR/ROOT device, it periodically sends again the ping request with some random jitter. The node sends this ping request as predefined by the user (default is 10).
11. If the node does not receive any ping response during this count, it detaches itself from the parent and restarts the joining process with the second discovered PAN ID (second BR/ROOT device).
12. If the node receives the ping response from the BR/ROOT, it saves the network parameters (PAN ID, parent address) in the flash memory for future communication and marks itself as a commissioned device.

Figure 9. Node joining process



4.3.3.2 Commission node joining process

1. After power on, if the device is in a commissioned state, it periodically sends the DIS packet with random jitter.
2. The device starts the RPL joining process.

5 6LoWPAN application

5.1 Application overview

The application we developed is a smart metering application based on a 6LoWPAN solution that uses the X-NUCLEO-S2868A2 expansion board and the NUCLEO-G070RB development board.

This application can be used in smart metering, smart lighting, etc. It follows the MSEDCL 6LoWPAN specification. The 6LoWPAN solution is interoperable with other competitor modules.

Considering the today's world of smart cities, smart lighting, and smart home, we designed this application to meet the metering concept requirements. Before, electricity metering was based on the actual manual reading, which required man power and wasted time.

In this smart metering application, we have used a Contiki-based 6LoWPAN mesh network technology. We have used the S2-LP RF device, which can be integrated in a smart meter. All the meters work as nodes in the 6LoWPAN mesh network.

The meter reader uses a small HHU/BR/ROOT device to fetch the meter data from all the nodes in the mesh network.

Our firmware supports both node/router and HHU/BR/ROOT functionality.

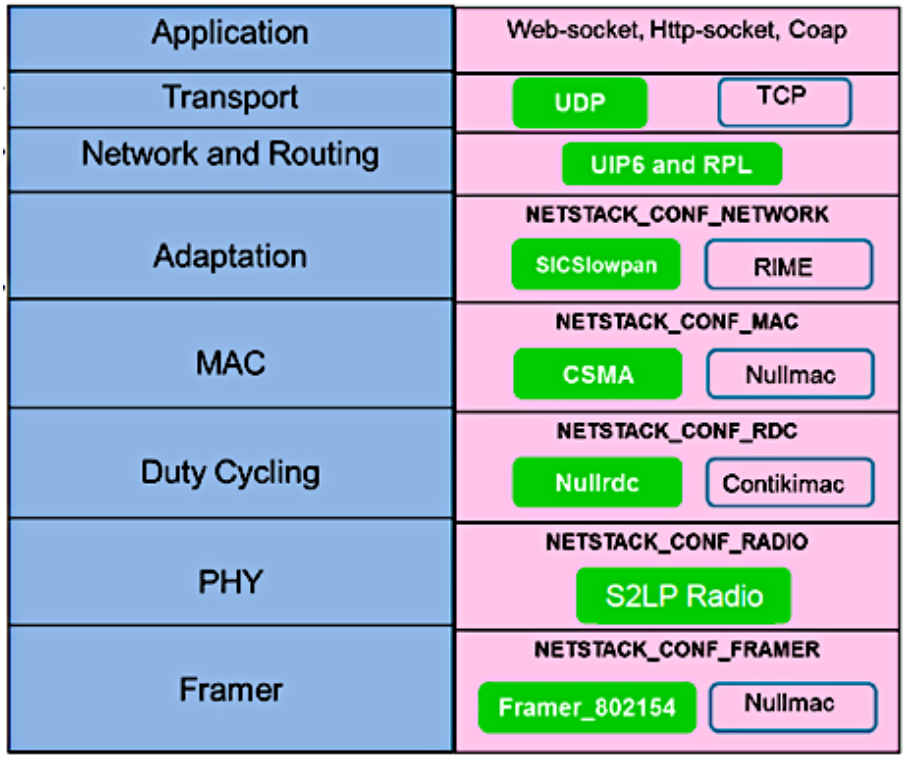
5.2 Key features

- Middleware library with Contiki OS and Contiki 6LoWPAN protocol stack 3.x
- Support for mesh networking technology through the standard RPL protocol
- Built-in support for NUCLEO-G070RB and X-NUCLEO-S2868A2
- Interoperable with other competitor modules
- Support for the 802.15.4g packet in the 6LoWPAN application
- AES CCM-32 128-bit encryption and decryption library at network layer
- AT command sets to configure device and radio parameters
- Support for beacon request, ping request, and response packet
- UART and RF commissioning protocol
- Multiple UDP ports and Hop-by-Hop protocol support for data communication
- Free and user-friendly license terms

5.3 Protocol overview

The diagram below shows the protocol stack layers. The green color identifies the blocks that are active/enabled in this application.

Figure 10. **Protocol stack**



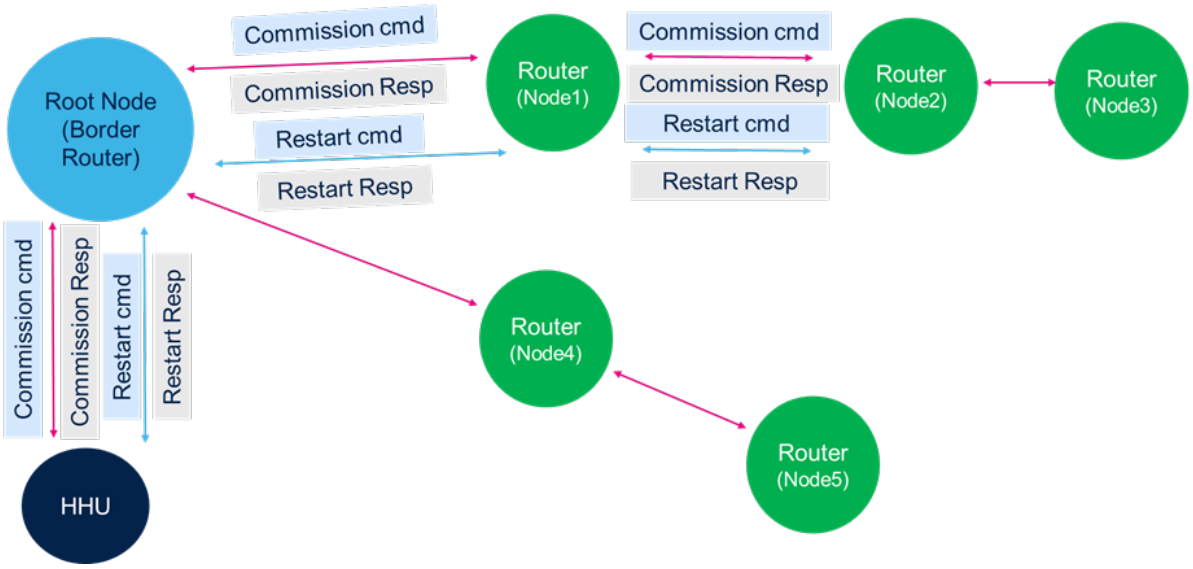
- **Application layer**: follows the MSEDCL specification. The application supports the AT command set to configure the device and radio settings.
- **Transport layer**: supports the UDP protocol. The application configures three UDP ports (61616, 61617, 61618) and a callback function that corresponds to each port for the data communication. The Hop-by-Hop option is enabled in all UDP packets.
- **Network and routing layer**: supports the IPv6 and ICMPv6 (Internet control message protocol for the Internet protocol version 6) protocols. It supports echo reply and echo request messages. RPL protocol is used for data routing. This protocol follows the RFC 6550.Transit information, including the parent option, is included in the DAO messages.

  IPv6 addresses are used in the protocol (16x8 bit). The first eight bytes represent the IPv6 prefix address. There can be a local or a global address (local address: FE 80 00 00 00 00 00 00). The next eight bytes represent the MAC address. It consists of the five most significant bytes that contain the organizationally unique identifier (OUI) purchased from the IEEE and the last three significant bytes mapped to the serial number of the meter.
- **Adaptation layer**: is the SICSlowpan layer. This is an adaptation of the IPv6 packets onto the underlying lossy low-power network (LLN). This layer takes care of the header compression, fragmentation, and reassembly.
- **MAC layer**: follows the IEEE 802.15.4G specification. It supports the carrier sense multiple access (CSMA) functionality to avoid collision.
- **Radio duty cycling**: the radio consumes more power than other components. Therefore, this layer turns the radio on/off radio to save power (SPLEEPY MESH).
- **PHY**: supports the S2-LP driver. It supports the frequency bands from 430-470 MHz, 860-940 MHz, and different modulation schemes (2-FSK, 4-FSK, 2-GFSK, 4-GFSK, OOK, and ASK).

5.4 Commissioning protocol

With the commissioning protocol, you can change the RF and network parameters of the existing network, like PAN ID, RF channel, device prefix, AES key, and device ID/MAC ID through RF. In this feature, the HHU device sends a commissioning command to the destination node and changes the required RF and network parameters. It then waits for the ACK (commission response). After receiving the ACK from the DST node, the HHU sends a restart command so that the updated parameters can be saved in the flash memory. At the next reboot of the device, the updated parameters appear.

Figure 11. Commissioning protocol



The diagram below shows the packet structure of the request to write the commissioning command attributes. You can update single or multiple configuration parameters through a single commission command.

Figure 12. Commissioning protocol packet

Frame Type	Seq No	CMD ID	No of Attributes	Attribute ID 1	Attribute ID1 Value	Attribute ID n	Attribute IDn Value
1B	1B	1B	1B	1B	optional	1B	optional

0x00 : R/W Commission cmd
0x01 : Restart cmd
0x02 : R/W Commission Resp
0x03 : Restart Resp

0x00 : Commission Write cmd
0x01 : Commission Write cmd Resp
0x02 : Commission Read cmd
0x03 : Commission Read cmd Resp

Parameters	Length (Bytes)	Default Value	Comments/limitation
RF Channel Number	1	0x02	Base freq is 865.125e6 Channel number: 0 to 9
Device Type	1	Node (Router)	Support both Node (Router) and BR (ROOT)
Device State	1	Not joined / Not Commission	Not joined = 0 Joined = 1
PAN ID	2	0xFFFF	
AES security key	16	<Customer Specific>	16 Bytes any key
IPv6 prefix	8	0xFFFF::	0xFFFF000000000000
MAC address	8	<Customer Specific>	User can SET 8 bytes device address
UART Baud rate	1	9600	Support:2400,4800,19200, 38400,57600,115200
Max PING count	1	10	

5.5 Node whitelisting

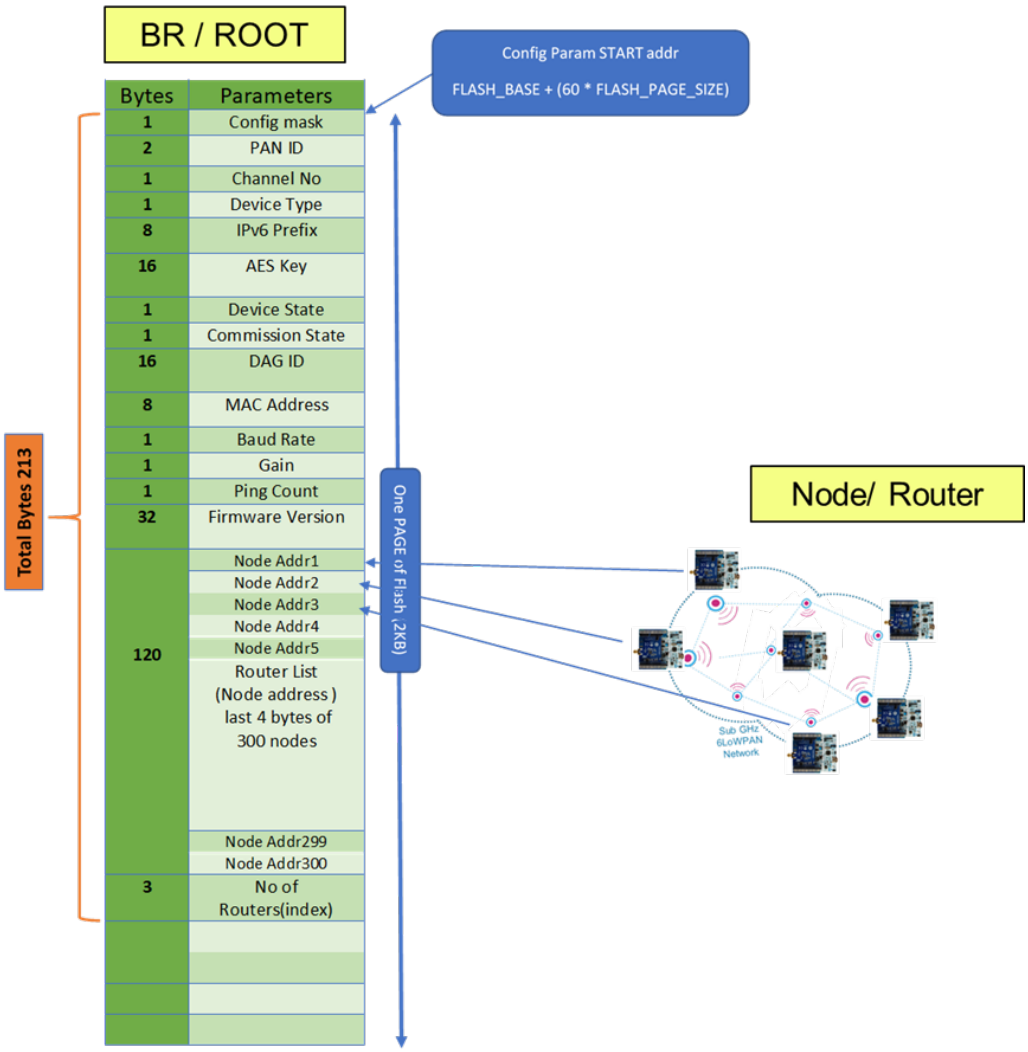
In this feature, when installing the new meter, you have to whitelist the new meter with the ROOT device.

In this ROOT device, store the last four bytes of the meter address. As soon as the meter is powered on, it immediately establishes a mesh network and connects with the ROOT. After power-on, the new meter sends the beacon and then the ping request. Then, the ROOT checks the meter address in the flash memory: if the address exists or the node is whitelisted, the ROOT device immediately sends the ping response. The meter joins and connects to the existing network.

5.6 Flash memory loader

The NUCLEO-G070RB supports 128 KB of flash memory. In this flash memory, each page is 2 KB. Therefore, a total of 64 pages is available for the user application as shown in the figure below.

Figure 13. Flash memory configuration



Currently, the device and RF configurations parameters are stored on page 60. We are using approximately 213 bytes of this page. Page 61 is reserved for future use.

If the configuration parameter size increases more the 2 KB, you can use page 61 for the extra configuration parameters.

Page 62 is used for the bootloader. So, you can flash the bootloader application on page 62.

Page 63 and page 64 are currently free. You can develop the application and archive the copy of the configuration parameters or bootloader in this location as a backup.

In the configuration parameters (page 60), the first byte is used as a configuration mask byte (0x55). This indicates that the configuration parameters are stored in this block. When the firmware boots for the first time on a fresh device or a fully erased device, it flashes the factory default parameters on this block and starts the joining process. After successfully joining the network, all the configuration parameters are stored in this block for future communication.

In case the device reboots due to power failure or some watchdog expiration, the system restores the last known network parameters and immediately joins the network.

In the configuration parameters, there is a router list, which stores the address of the nodes of the network. This is called whitelisting.

5.7 Device capability

All the devices in the network can be a 6LoWPAN node (router) or a BR (ROOT) device.

In the factory default settings, the device works as a simple node (router) device with the default configuration parameters listed below.

Table 4. Factory default parameters

S/N	Parameters	Length (bytes)	Default value	Comments/limitations
1	RF channel number	1	0x02	The base frequency is 865.125e6 and the channel space is 200e3. The channel number is from 0 to 9.
2	Device type	1	Node (router)	It supports both node (router) and BR (ROOT).
3	Device state	1	Not joined/not commissioned	Not joined = 0. Joined = 1.
4	PAN ID	2	0xFFFF	
5	AES security key	16	<Customer specific>	16 bytes for any key.
6	IPv6 prefix	8	0xFFFF:	0xFFFF000000000000
7	MAC address	8	<Customer specific>	You can set an eight-byte device address.
8	UART baud rate	1	9600	2400, 4800, 19200, 38400, 57600, 115200
9	Max. ping count	1	10	

Factory default parameters can be changed using AT commands.

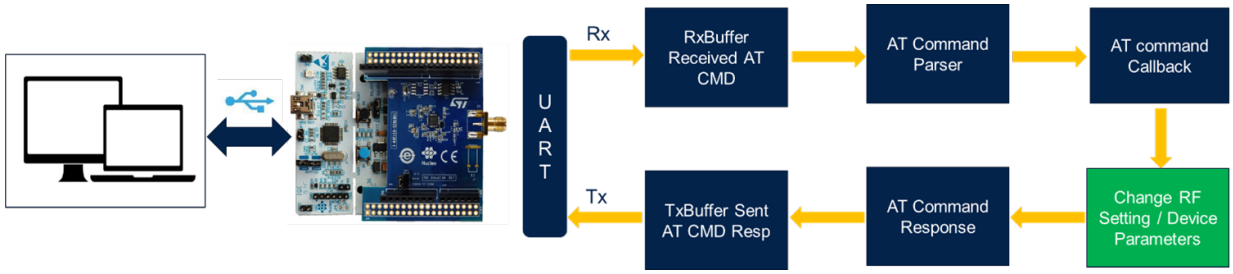
5.8 AT commands

The AT command sets make the solution more robust and flexible. You can configure the RF and network parameters either through the RF commissioning protocol or through the AT command sets.

AT commands work on the UART terminal. Therefore, after opening the docklight file, you have to select the COM port number and set the baud rate (9600). The docklight file(.ptp file) provides some sets of AT commands.

Using the docklight file, you can also check the configuration parameters by executing the read command. After the read/write AT command, you receive the command response on the terminal as shown in the diagram below.

Figure 14. AT command diagram



If you receive no response on the terminal, check that the COM port connection and settings are correct.
After changing the configuration parameters, press [**Save**] and [**Exit**] to make the changed parameters effective.
You have to install the docklight software to execute the AT command set.
The PTP file that corresponds to the node and border router is available in the utility folder.

5.8.1 AT command list

The table below lists the AT commands supported by the smart metering application.
The AT command codes consist of two bytes. The description column defines the range of the configuration parameters.

Table 5. AT command list

S/N	AT command name	Command code	Description
Generic command			
1	Restore	0x0000	It restores the factory default settings.
2	Exit	0x0001	Device restart (soft restart)
3	Save	0x0002	It saves the updated parameters in the flash memory.
RF/device configuration command			
4	R/W baud rate	0x0005	1: 2400 2: 4800 3: 9600 4: 19200 5: 38400 6: 57600 7: 115200
5	R/W power output (S2-LP gain)	0x0006	0: 0 dBm 70: 10 dBm 7F: max. power (approximately 14 dBm)
6	R/W frequency channel	0x0008	0 to 9
7	R/W PAN ID	0x0009	0 to 0xFFFF
8	R/W serial number	0x000A	Eight-byte device ID
9	R/W AES key (link key)	0x000B	16 bytes (user-defined)
10	Read hardware version	0x000E	It displays the hardware version.
11	Read firmware version	0x000F	It displays the released firmware version.
12	R/W ping count	0x0013	It defines the number of the max. ping count.
13	R/W device type	0x0016	1: ROOT 2: router
14	R/W IPv6 prefix	0x0035	As per IPv6 specifications
15	R/W parent info	0x0037	It reads the parent ID.
16	Read device state	0x0063	The device is commissioned or not.
17	Check flash status	0x00AB	It returns the flash memory read status.
18	Config carrier wave signal	0x00AC	It configures the carrier wave signal.
19	Read DODAG ID	0x00AD	It reads the DODAG ID.
29	R/W S2-LP calibration offset	0x00AE	It configures the S2-LP calibration RF offset.

5.8.2 AT command and response format

The tables below describe the format of the read/write AT command and its response.

Table 6. AT command format

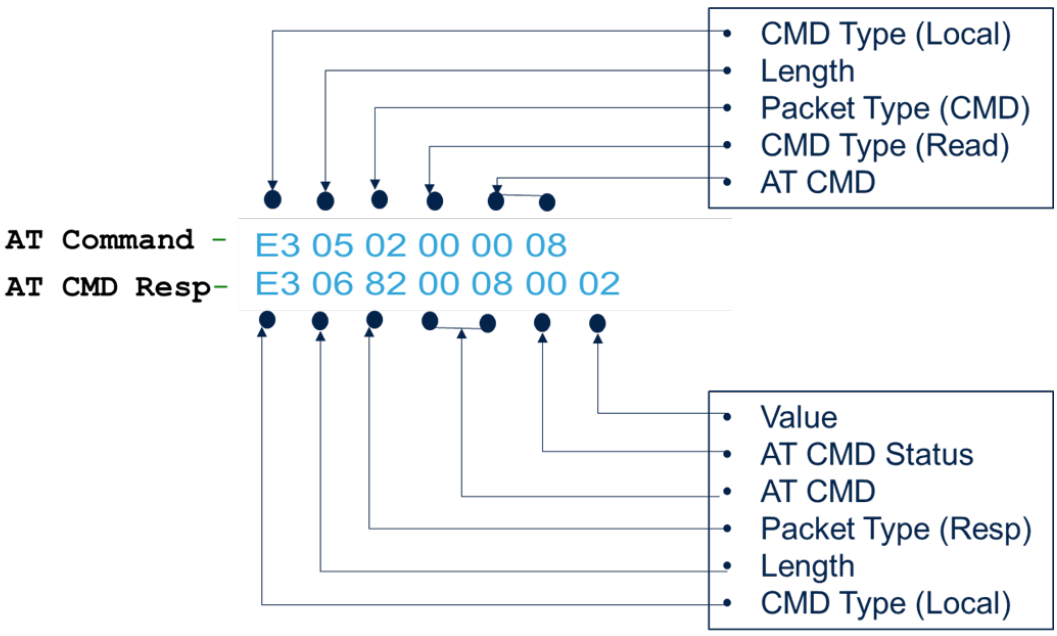
CMD type (1 byte)	Length (1 byte)	Packet type (1 byte)	CMD type R/W (1 byte)	AT command (2 bytes)	Value
0xE3: local CMD	Length of the packet	0x02: CMD	0x00: read CMD 0x01: write CMD	See Table 5. AT command list.	Optional

Table 7. AT command response

CMD type (1 byte)	Length (1 byte)	Packet type (1 byte)	CMD type R/W (1 byte)	CMD status (1 byte)	Value
0xE3: local CMD	Length of the packet	0x82: response	See Table 5. AT command list.	0x00: success 0x01: invalid parameter 0x02: AT CMD error	Optional

- **CMD type**: specifies that the AT command is a local command (0xE3) or is an RF command (0xE4) as a data downloading/commissioning command.
- **Length**: length of the packet.
- **Packet type**: specifies whether it is a command packet or a response packet.
- **CMD type R/W**: specifies whether it is a read or a write command.
- **AT command**: specifies the AT command code (as mentioned in the table).
- **Value**: value of the RF/network parameters.
- **CMD status**: used in the response packet to specify the status of the AT command.

Figure 15. Example of AT command: “RF Channel Read”



5.8.3 AT command for data download

The HHU executes the data downloading command in the destination node of the mesh network. With this command, you can specify the destination meter/node address and payload as specified in the table below. The payload is a meter protocol-specific command.

Table 8. **Command for data download**

CMD type (1 byte)	Length (1 byte)	Packet type (1byte)	Destination address (8 bytes)	Frame type over RF (3 bytes)	Payload to be sent over UDP
0xE3	0x28	0x01	3C C1 F6 01 01 5F 22 10	00 01 00	40 08 00 FC 0D BF 08 03 01 02 00 00 00 01 00 02 00 03 00 04 00 05 00 06 00 07 00

5.8.4 AT command for commission protocol

The figure below shows the AT command for the commissioning request. You can configure the RF and network configuration parameters using this AT command. You can also modify a single or multiple parameters anytime.

Figure 16. **Commission commands**

CMD Type	Len	Packet Type	Dst Addr (8B)	Frame Type (1B)	Seq No (1B)	CMD ID (1B)	Attribute Count (1B)	Attribute ID1 (1B)	Attribute ID1 value	Attribute ID2 (1B)	Attribute ID2 value	Attribute ID3 (1B)	Attribute ID3 value	Attribute ID4 (1B)	Attribute ID4 value	Attribute ID5 (1B)	Attribute ID5 value
E3	2F	05	00 80 E1 0B 01 5F 22 B1	00	02	00	05	00	01D0	01	08	02	01	03	AA AA 00 00 00 00 00 00	04	00 00 00 00 00 00 00 00 00 00 00 00 4D 53 45 44 43 4C

- E3: CMD type
- 2F: length
- 05: packet type
- 00 80 E1 0B 01 5F 22 B1: destination address
- 00: frame type commissioning for read/write
- 02: sequence number
- 00: command ID (commissioning write request)
- 05: attribute count
- 00: attribute ID 1 (short PAN ID)
- 01D0: short PAN ID value
- 01: attribute ID 2 (channel)
- 08: channel value
- 02: attribute ID 3 (device type)
- 01: device type value
- 03: attribute ID 4 (IPv6 prefix)
- AA AA 00 00 00 00 00 00: IPv6 prefix value
- 04: attribute ID 4 (AES key)
- 00 00 00 00 00 00 00 00 00 00 4D 53 45 44 43 4C: 16-byte AES key

5.9 Contiki process

All the Contiki programs are processes. A process is a piece of code that the Contiki system executes regularly. The processes typically start when the system boots, or when a module that contains a process is loaded into the system. The processes run when something happens (for example, a timer firing or an external event occurrence). All the node initialization requests (like beacon and ping) are executed in the process.

A process is first declared at the top of a source file. The `PROCESS()` macro takes two arguments: one is the variable with which we identify the process and the other is the name of the process. On the second line, we signal to the Contiki system that this process should be automatically started after the system boot.

You can specify multiple processes by separating them with commas. If the `AUTOSTART_PROCESSES()` line does not include an existing process, you have to start manually that process by using the `process_start()` function.

Figure 17. **Contiki process**

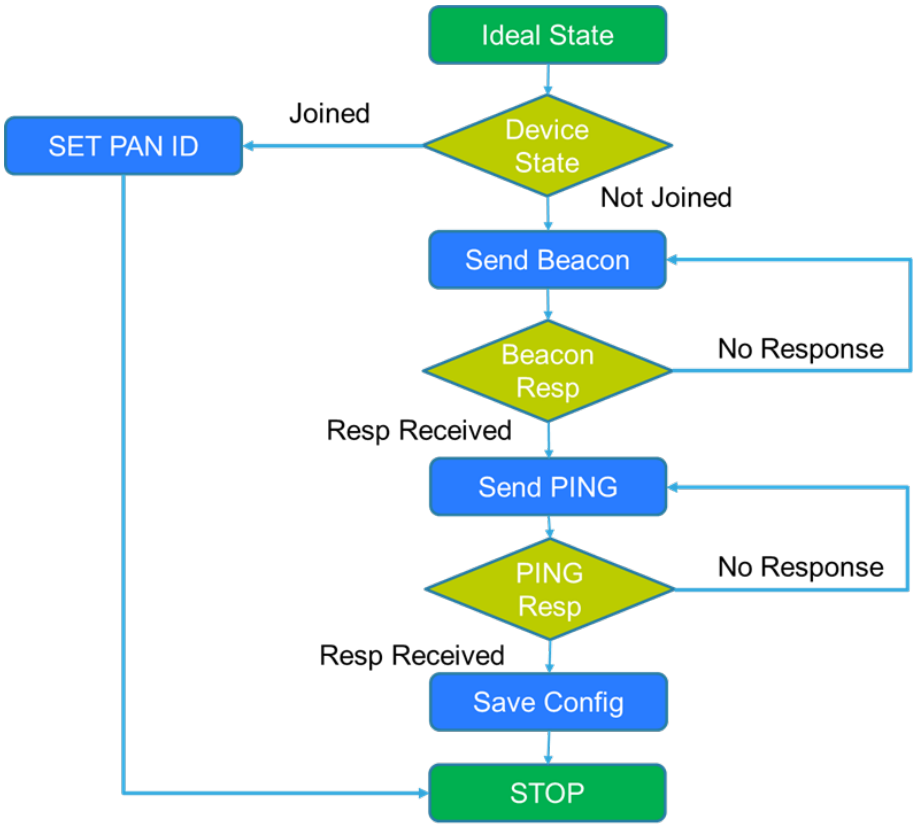
```
PROCESS(Node_Init_Process, "Node Init Process");
AUTOSTART_PROCESSES(&unicast_sender_process,&Node_Init_Process,&border_router_process);
```

We use the following three processes in the application:

1. **Unicast sender process**: registers all the UDP ports and the callback functions corresponding to the port numbers.
2. **Border router process**: runs only when the device is configured in ROOT/BR mode. With this process, we configure the IPv6 prefix address and the PAN ID.
3. **Node initialization process**: runs only when the device is in node/router mode. The node starts the initialization process by sending the beacon and ping requests as shown in the flowchart below.

Note: You can create/add processes according to your needs.

Figure 18. **State machine of the node initialization**



5.10 UDP communication

The application provides three UDP ports and a callback function corresponding to each port for the data communication:

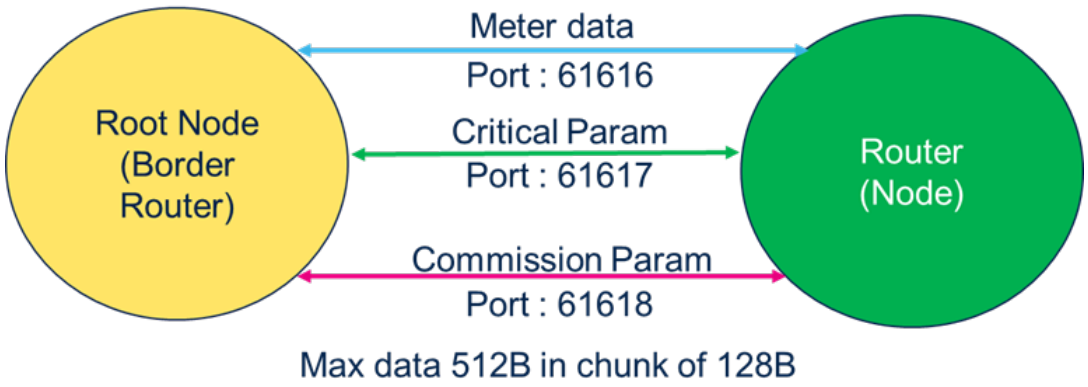
- Port no. 61616: used to read metering data.
- Port no. 61617: used to set critical parameters.
- Port no. 61618: used to set the network parameters and for the commissioning protocol.

After the network joining process, you can transmit up to 512 bytes of raw data via the AT command. The data are encrypted and fragmented into chunks of 128 bytes (S2-LP FIFO size is 128 bytes).

The device receives all the fragmented packets and reassembles them. After that, it decrypts the packets using the AES key. Then, it provides the data to the user through the user callback function, registered during the initialization. The user does not receive any UDP callback in case of error.

Using the commission protocol, you can also change the network parameters like the PAN ID, the channel number, the device type (ROOT or router), the IPv6 prefix, and the AES key. The commission protocol sends then its response to the same UDP port.

Figure 19. State machine of the node initialization



6 Firmware description

6.1 Setting the radio configuration parameters

You can set the radio parameters in the following section of the user-configurable file (radio-driver.h). However, it is recommended not to change these settings.

Figure 20. Radio configuration setting

```
#define BASE_FREQUENCY                  865.125e6 //channel 0
#define CHANNEL_NUMBER                  2
#define CHANNEL_SPACE                   200e3
#define DATARATE                        50000
#define MODULATION_SELECT               MOD_2GFSK_BT1
#define FREQ_DEVIATION                  25e3
#define SYNC_WORD                       0x55904E00
#define SYNC_LENGTH                     SYNC_BYTE(3)
#define PREAMBLE_LENGTH                 PREAMBLE_BYTE(3)
#define BANDWIDTH                       160e3
#define RSSI_RX_THRESHOLD               -105.0
#define RSSI_TX_THRESHOLD               -80.0
#define CRC_MODE                       PKT_CRC_MODE_16BITS_2
```

6.2 Setting the packet configuration parameters

You can set the packet configuration in the following part of the code (file name: radio-driver.h). However, it is recommended not to change these settings.

Figure 21. Packet configuration setting

```
#define USE_BASIC_PROTOCOL
#define EN_WHITENING                    S_DISABLE
#define PREAMBLE_LENGTH                 PREAMBLE_BYTE(8)
#define VARIABLE_LENGTH                 S_ENABLE
#define EXTENDED_LENGTH_FIELD           S_DISABLE
#define EN_ADDRESS                      S_DISABLE
#define EN_FEC                          S_DISABLE
#define PREAMBLE_BYTE(v)                (4*v)
#define SYNC_BYTE(v)                    (8*v)
#define RADIO_MAX_FIFO_LEN              128
#define XTAL_FREQUENCY                  50000000    /* Hz */
```

6.3 Device default parameters

The define for the device default configuration parameters is shown below. It is in the radio-driver.h file.

Figure 22. **Device default parameters**

```
#define MAX_NO_OF_BR            10
#define DEFAULT_DEVICE_TYPE     Device_Type_ROUTER
#define DEFAULT_DEVICE_STATE    Device_Not_Joined
#define DEFAULT_PAN_ID          0xFFFF
#define DEFAULT_BAUD_RATE       3  // 3: 9600, 7 : 115200
#define DEFAULT_PING_COUNT      10
```

7 Tests and results

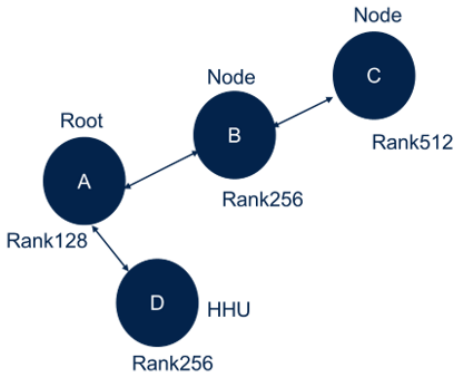
7.1 Data downloading performance

The HHU device sends the data downloading command to all the meters connected in the mesh network. The device then downloads their data.

We have measured the performance of data download up to three hops, including the time to send the command and receive the data back.

The figure below shows the results of this test case.

Figure 23. Data downloading performance



RF performance of commands and data

Data Downloading command (85 bytes including payload)	D to A	A to B	B to C
Approximate Time Taken on RF	30ms	31ms	30ms

Response Data (2 segments, 167 bytes, including payload)	C to B	B to A	A to D
Approximate Time Taken on RF	71ms	56ms	57ms

Data downloading performance

7.2 Mesh stability

We have validated the mesh stability up to six hops by executing the data downloading command in a four-second interval.

Figure 24. Mesh network

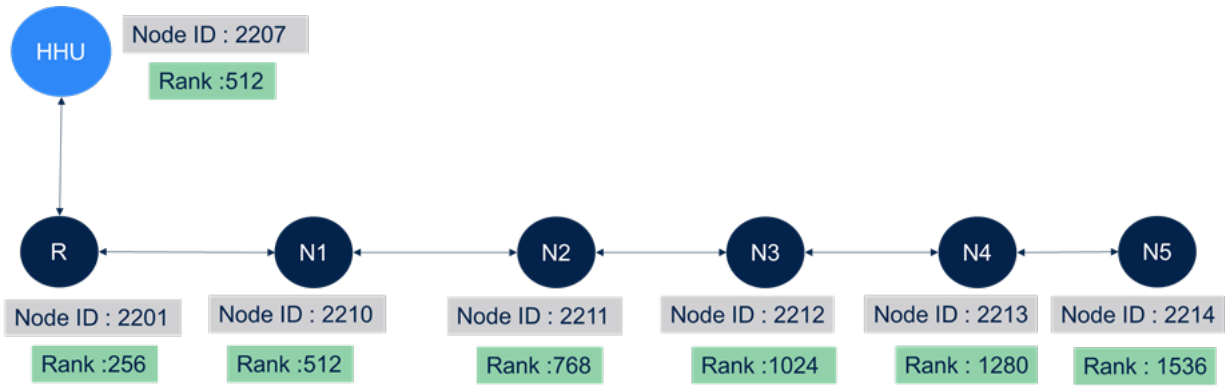


Table 9. Test results

Number of hops	Data download (four-second interval)
Third hop	Two days
Fourth hop	One day (25 hours)
Fifth hop	One day (26 hours)
Sixth hop	Three hours

Appendix A References

1. For the NUCLEO-G070RB development board, see:
   - Datasheet
   - UM1727: "Getting started with STM32 Nucleo board software development tools"
   - Schematic diagrams
2. For the X-NUCLEO-S2868A2 expansion board, see:
   - S2-LP datasheet
   - UM2638: "Getting started with the X-NUCLEO-S2868A2 Sub-1 GHz 868 MHz RF expansion board based on S2-LP radio for STM32 Nucleo"
   - Schematic diagrams
3. For MSEDCL specification, see:
   "Tech_Spec_5_30A_RF_Meter_with_ENCLOSURE_LPRF_6LoWPAN_Revised_25.10.2018.pdf" at https://www.mahadiscom.in/supplier/material-technical-specifications/
4. Contiki GitHub

Revision history

Table 10. **Document revision history**

Date	Revision	Changes
25-Jan-2022	1	Initial release.

Contents

List of tables

List of figures