TGR 2019 Pre-Training

Nov 2018

Pisit Rojthongkham

life.augmented

System Requirement

- Laptop
  - Administrative privileges (Admin rights) needed for driver/software install and for compiling codes.
- Windows Operating System
  - XP/Vista/Win7/Win8
- **Please bring the following:**
  - 1x USB 2.0 A - MICRO B MALE CABLE

Installations before the workshop: Download the following files

Download the following software from the links provided. Refer to the installation procedures in the next pages.

- Software Development Environment:
  - KEIL MDK-ARM for STM32F0 and STM32L0: www2.keil.com/stmicroelectronics-stm32/mdk
    - **Important:** Requires internet access to activate the free license.
  - STMicroelectronics STM32L0 Series Device Support and Examples (Device Family Pack) from www.keil.com/dd2/Pack/
- ST-Link USB driver: www.st.com/stlinkv2
- STM32 ST-Link Utility: www.st.com/stlinkv2
- Terminal emulator software: Teraterm https://ttssh2.osdn.jp/
- LoRaWAN firmware library I-CUBE-LRWAN : www.st.com/i-cube-lrwan

Software Installation Steps

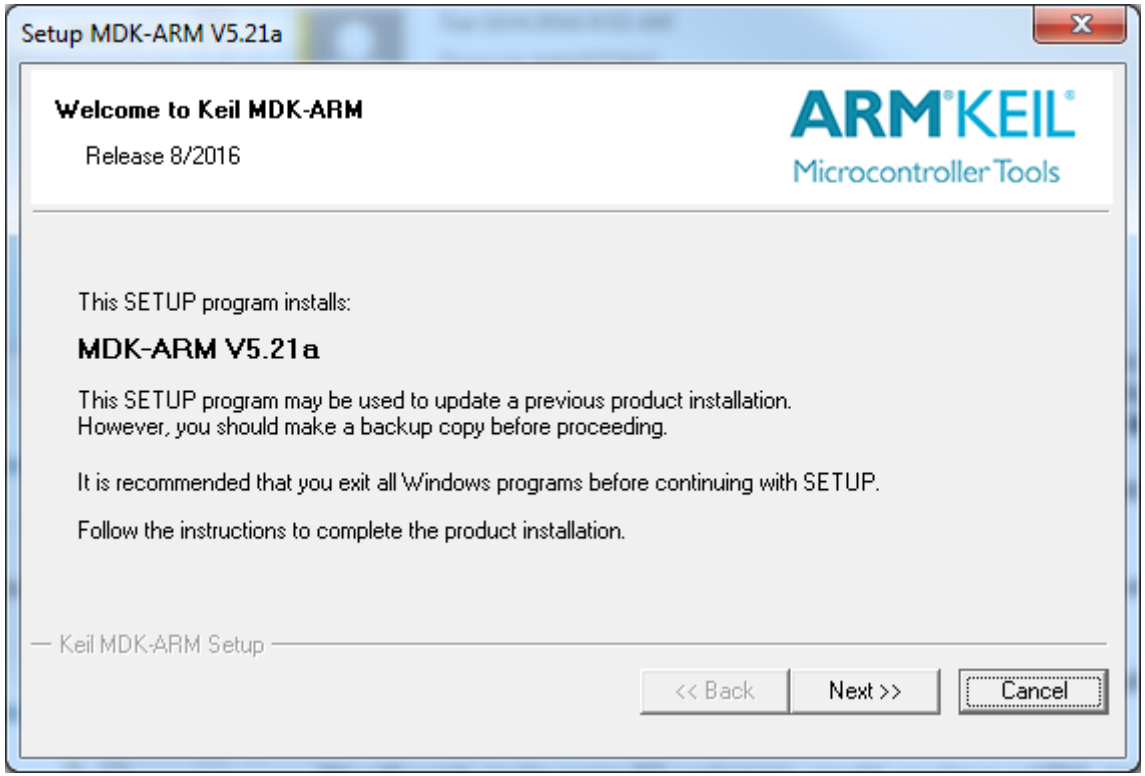
Step 1: KEIL MDK-ARM Installation

- We will be using KEIL MDK-ARM v5 in this session.
- **MDK for STMicroelectronics STM32F0 and STM32L0**
  - Keil MDK-ARM for STM32F0 and STM32L0 provides software developers working with STM32 devices based on the ARM Cortex-M0 and ARM Cortex-M0+ cores with a **FREE-TO-USE** professional tool suite.
    - Product Serial Number (PSN) to activate the MDK for STM32F0 and STM32L0:
      - **4PPFW-QBEHZ-M0D5M**
- Keil MDK is the most comprehensive software development system for ARM processor-based microcontroller applications. Keil MDK includes the ARM C/C++ Compiler, the CMSIS-RTOS RTX Kernel, and the µVision IDE/Debugger.

From www2.keil.com/stmicroelectronics-stm32/mdk

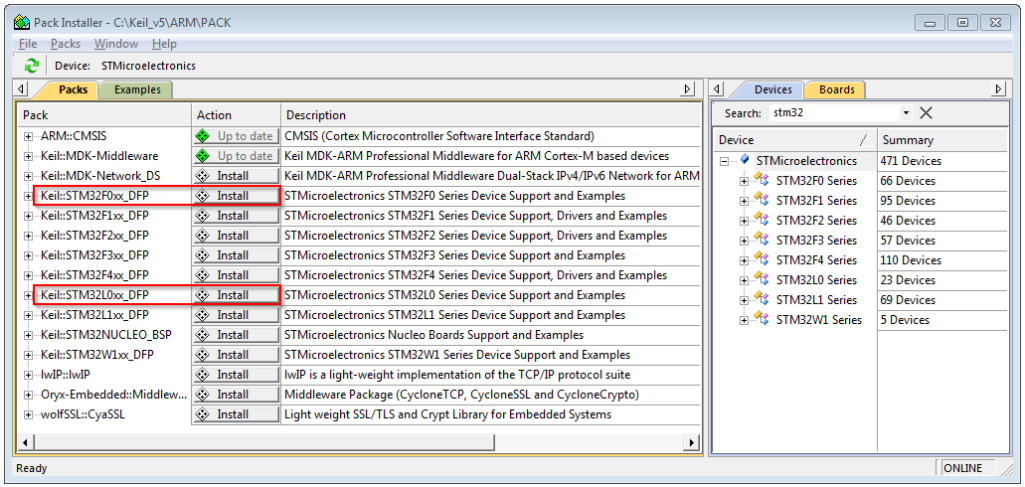
Step 1a: KEIL MDK-ARM Installation

- Download Keil MDK-ARM www2.keil.com/stmicroelectronics-stm32/mdk
  - Take note of the **Product Serial Number (PSN)** as you will use this later to activate the License.
- Double click on **MDK5xx.EXE** to begin installation.



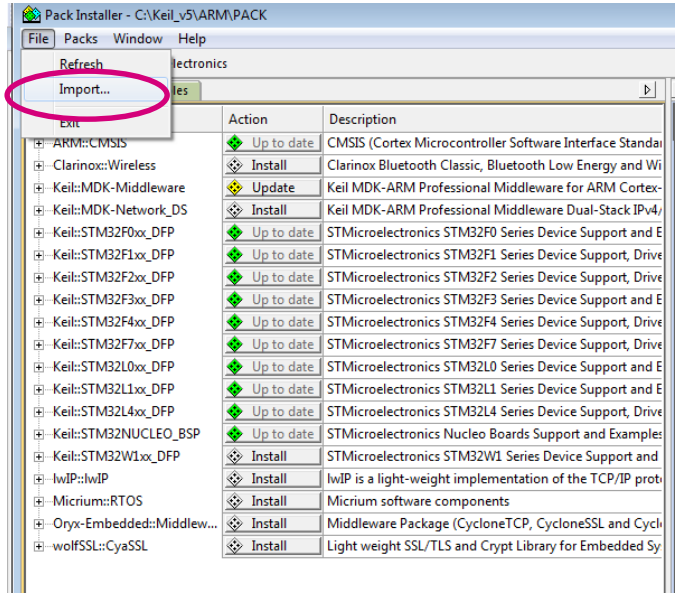
Step 1b: STM32L0 Device Family Pack (DFP) Installation

- **Device Family Pack (DFP)** contains CMSIS system/startup files, drivers, and flash algorithms for a microcontroller device family.
- The STM32L0 DFP is required to be able to compile STM32L0 Keil projects.
- After installation, KEIL Pack Installer should automatically launch. Click Install to download and install the STM32L0 Device Family Packs supplied by KEIL.
  - If not, launch it manually by clicking ***Project menu>Manage>Pack Installer***



Alternatively,

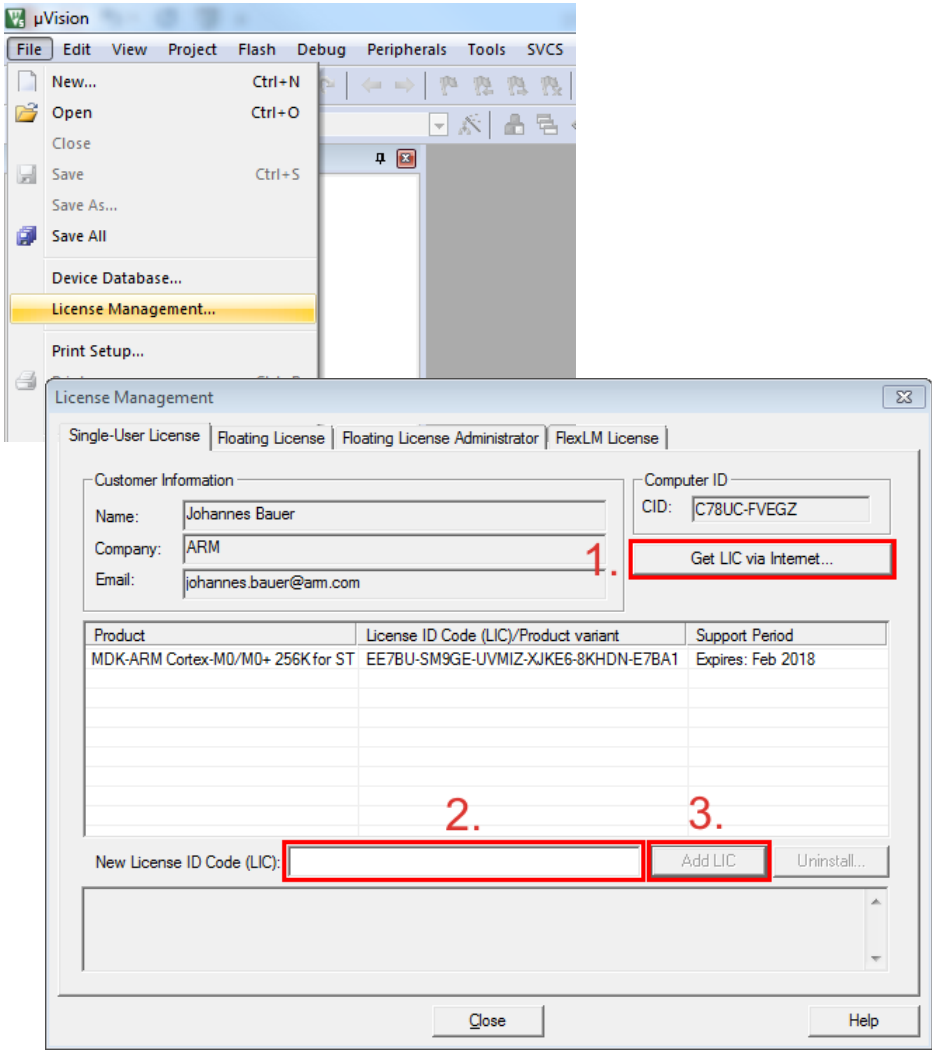
- An alternative way to install the STM32L0 DFP, is to manually import it.
- Download the STM32L0 DFP www.keil.com/dd2/Pack/
- Import the downloaded file (Keil.STM32L0xx_DFP.1.6.0.zip) through *File menu>Import*



life.augmented

Step 1c: KEIL MDK-ARM License Activation

- Login with an account that has administration rights.
- Right-click the µVision icon and select **Run as Administrator...** from the context menu.
- Open the dialog **File — License Management...** and select the **Single-User License** tab.
- Click the button **Get LIC via Internet...**, then click the button **OK** to register the product. This action opens the License Management page on the Keil web site.
- Enter the **Product Serial Number 4PPFW-QBEHZ-M0D5M** along with your contact information and click the button **Submit**. An e-mail is sent back with the **License ID Code (LIC)** within a few minutes.
- To activate the Software Product, enter the **LIC** in the field **New License ID Code (LIC)** of the dialog **License Management...** and click **Add LIC**.

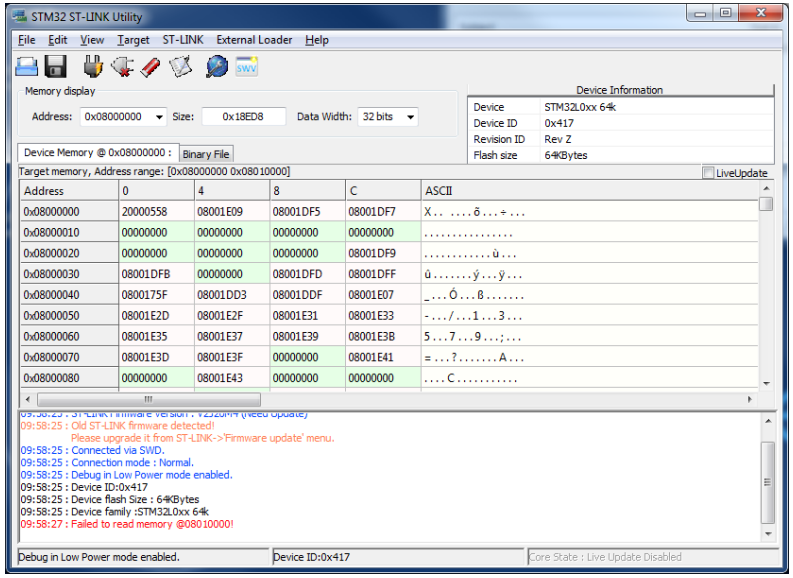


life.augmented

Step 2: ST-Link USB driver And ST-Link Utility

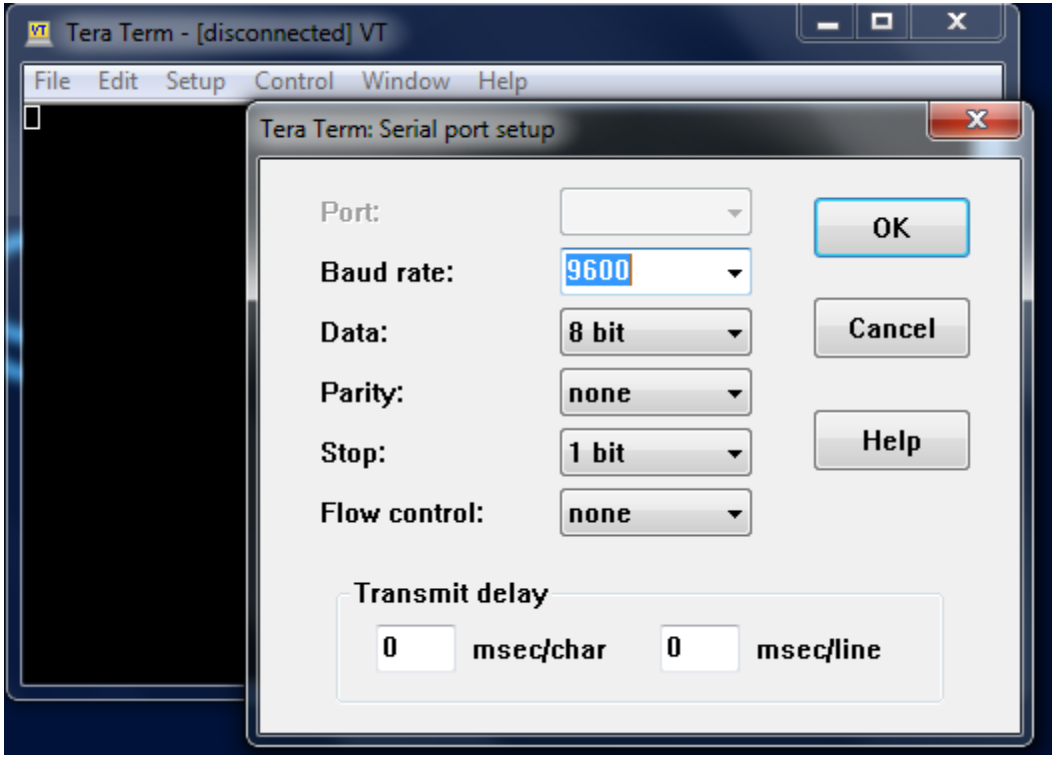
- Download the ST-Link USB drivers from www.st.com/stlinkv2 and install the drivers.

- Also download the STM32 ST-Link Utility from www.st.com/stlinkv2 and install the utility.
  - The STM32 ST-Link Utility is an STM32 programming software using the ST-Link programmer/debugger tool.
  - Besides programming capability, the tool can also be used to verify the ST-Link driver installation and connection to the target MCU.



Step 3: Teraterm

- Tera Term is a free software terminal emulator (communication program) which supports multiple communication including Serial port connections which will be used during this workshop.
- Download Teraterm from https://ttssh2.osdn.jp/
- Run the teraterm-4.92.exe and follow the installation wizard.



Hands-on Outline

- STM32L0 Overview
- STM32LoRa Discovery kit Overview
- LoRa Device Firmware Library Overview
- LoRa Sensor Device Setup and Reconfiguration
- Hand-on Sensor: Temperature, Humidity, Pressure
- Hand-on Sensor: Magneto

STM32L Ultra-low-power

Ultra-low-power, market-proven solutions
Best in class with up to 100 DMIPS performance

STM32L DNA

- Ultra-low leakage technology
- Flexible LP[4] Modes
- Optimized design for ULP[4]
- Operating from 1.65 to 3.6 V
- From - 40 to 125 °C
- Reset circuitry

- Rich peripheral set
- Advanced analog features
- 16-bit, 32-bit timers
- Low power Batch acquisition mode (BAM)

- 2 watchdogs
- Temperature sensor
- Unique ID
- Cap. touch-sensing
- Single wire protocol

Product Series	System	Advanced Periph.	USB 2.0	LCD	Security
STM32L4	ART Accelerator™ Vbat New LP[4] Modes SDIO/FSMC[1]	SAI CAN Quad-SPI DFSDM[2] LP UART LP Timers	FS OTG + Xtal less	Seg. up to 8x40 + TFT	256-bit AES + TRNG
STM32L1	True EEPROM with RWW[3] + SDIO/FSMC[1]		FS	Seg. up to 8x40	128-bit AES
STM32L0	True EEPROM with RWW[3]	LP UART LP Timer	FS + Xtal less	Seg. up to 8x48	128-bit AES + TRNG

STM32 L4

- ARM Cortex-M4 + FPU at 80 MHz – 100 DMIPS
- From 128 Kbytes to 1 Mbyte of Flash memory
- Lowest power mode + RAM + RTC: 0.6 µA

ULPBENCH™ An EEMBC Benchmark 210 — COREMARK® An EEMBC Benchmark 273

STM32 L1

- ARM Cortex-M3 at 32 MHz – 33 DMIPS
- From 32 to 512 Kbytes of Flash memory
- Lowest power mode + RAM + RTC: 1.2 µA

ULPBENCH™ An EEMBC Benchmark 81 — COREMARK® An EEMBC Benchmark 93

STM32 L0

- ARM Cortex-M0+ at 32 MHz – 26 DMIPS
- From 8 to 192 Kbytes of Flash memory
- Lowest power mode + RAM + RTC: 0.8 µA

ULPBENCH™ An EEMBC Benchmark 161 — COREMARK® An EEMBC Benchmark 75

[1] FSMC : Flexible Static Memory Controller
[2] DFSDM : Digital Filters for Sigma Delta Modulation. Accepts digital microphones pdm input signal
[3] RWW : Read While Right (Dual Bank Flash and Dual Bank EEPROM)
[4] ULP / LP : Ultra-low-power / Low power

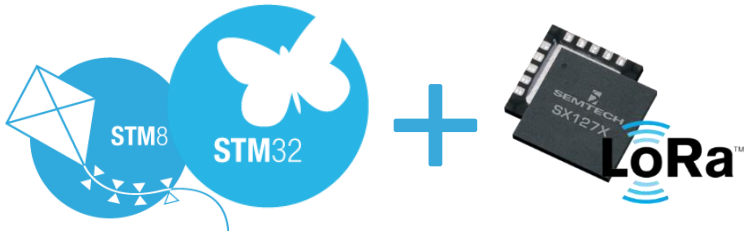
LoRa® powered by STM32™

www.st.com/stm32-lrwan

Available today

USI® Module
AT command

Cost optimized solution



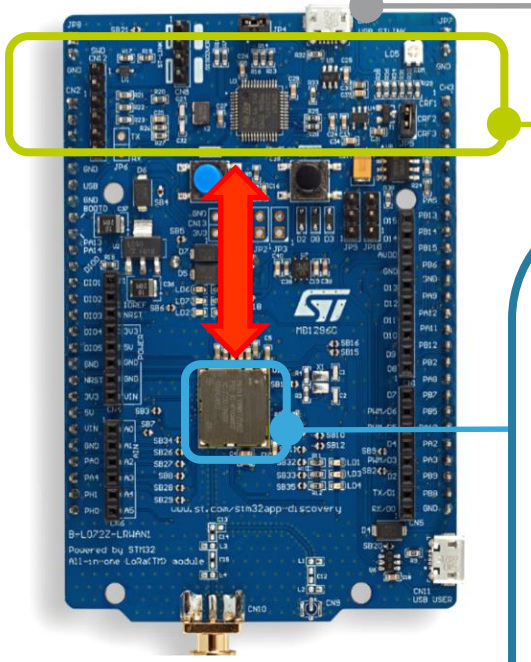
Flexible design architecture
More than **1000** P/N of **STM8/STM32**

Murata® Module
All-in-one Open

All-in-one LPWAN

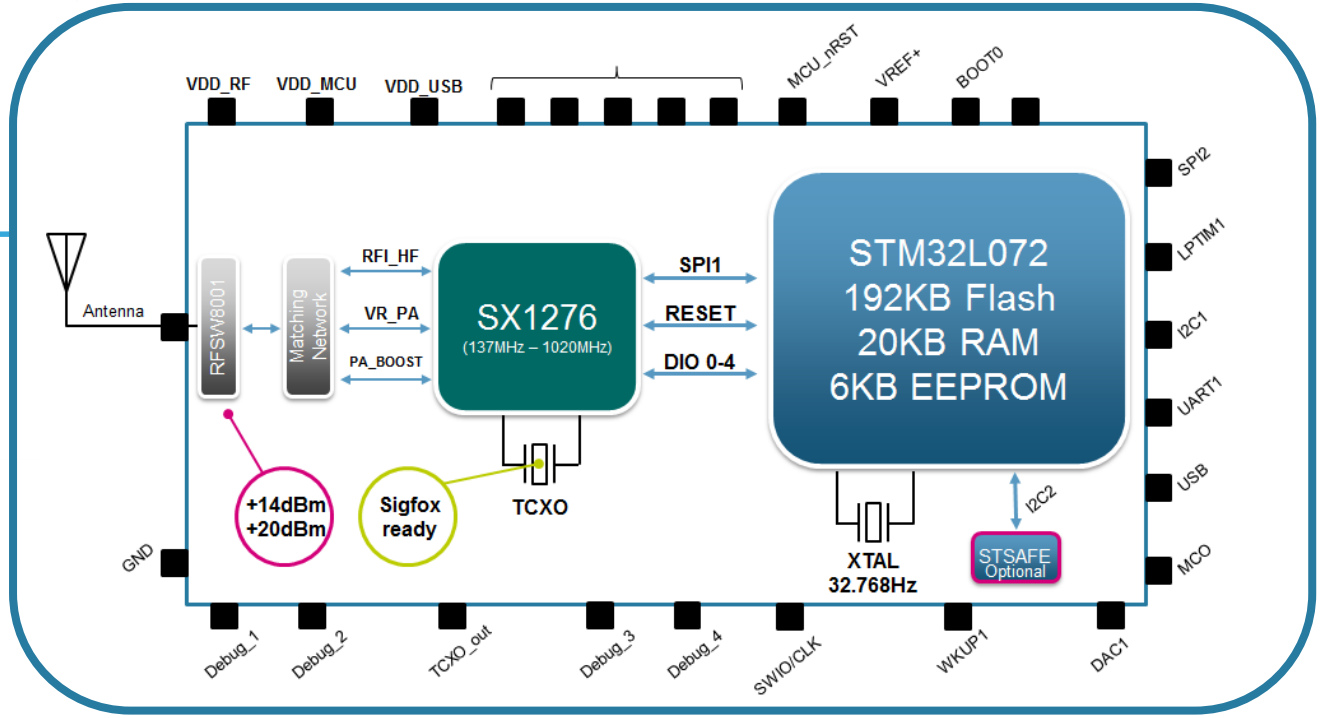
STM32LoRa Discovery kit

B-L072Z-LRWAN1: STM32 and LoRa® Discovery kit



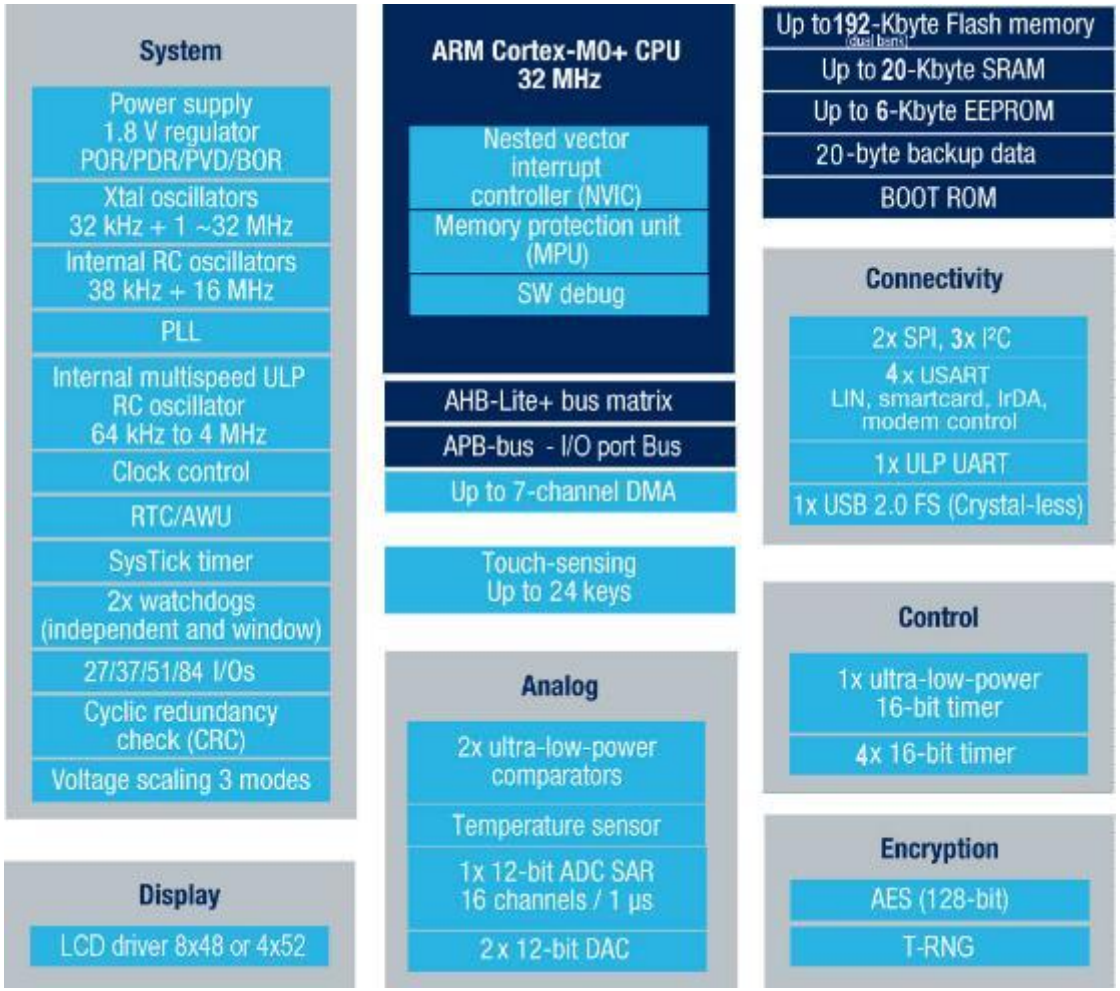
Flexible board power supply :
through USB or external source

Integrated ST-Link/V2-1:
mass storage device flash programming



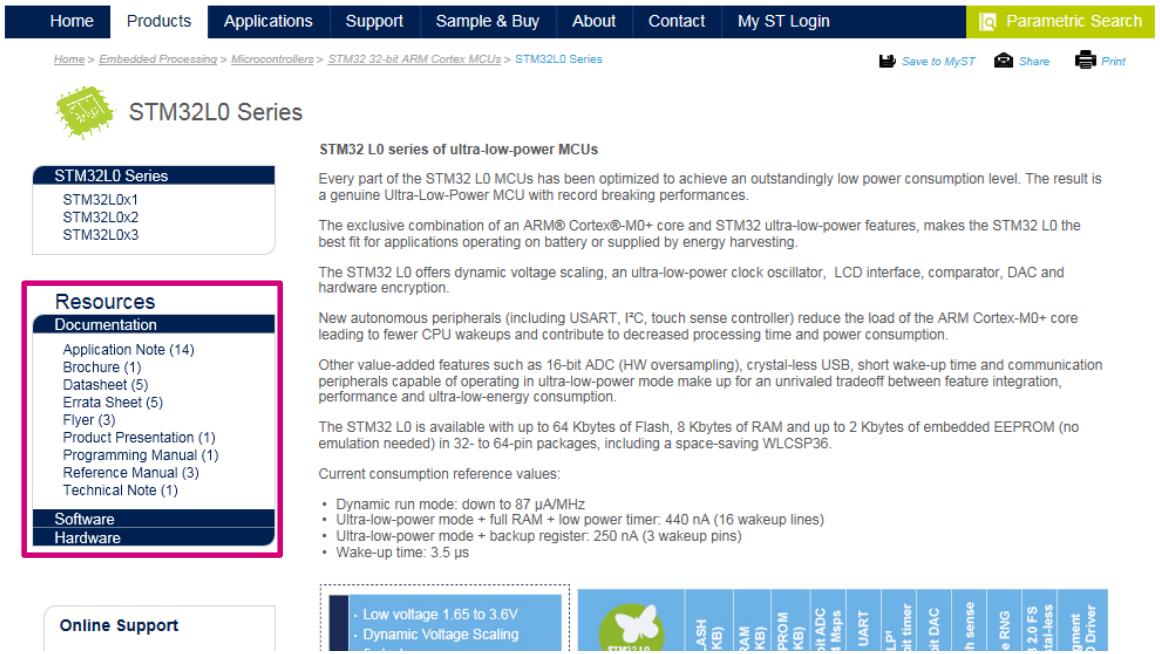
STM32L07x block diagram

- Key features
  - ARM Cortex-M0+ at 32MHz
    - Single-cycle I/O access
    - Single-cycle multiplier (MUL)
    - 0.95 DMIPS/MHz
  - 1.71V to 3.6V, 32MHz full functional
  - Digital down to 1.65V
  - -40°C to +125°C temperature range
  - ADC with build-in HW oversampling
    - Down to 1.65V
  - Flash + Ram code sector lock
  - USB 2,0 FS certified
    - Build-in 48MHz oscillator
    - Battery Charger Detection
    - Link Power Management
  - Independent clock domain
    - I2C, USART/UART
    - USB
  - **5x** timers
    - 1x 16-bit (4ch)
    - 3x 16-bit(2ch)
    - 1x 16-bit LP[1] available in stop



STM32L0 Documents

- Related documents downloadable from www.st.com/stm32l0
- STM32L0 Reference Manual
  - This reference manual targets application developers. It provides complete information on how to use the STM32L0xx microcontroller memory and peripherals.
  - Includes Peripheral Functional Description and Register Description.
- STM32L0 Series Cortex®-M0+ programming manual
  - Provides a full description of the STM32L0 Cortex-M0+ processor programming model, instruction set and core peripherals.
- Datasheet
  - Device overview
  - Functional overview
  - Pin descriptions
  - Memory mapping
  - Electrical characteristics



life.augmented

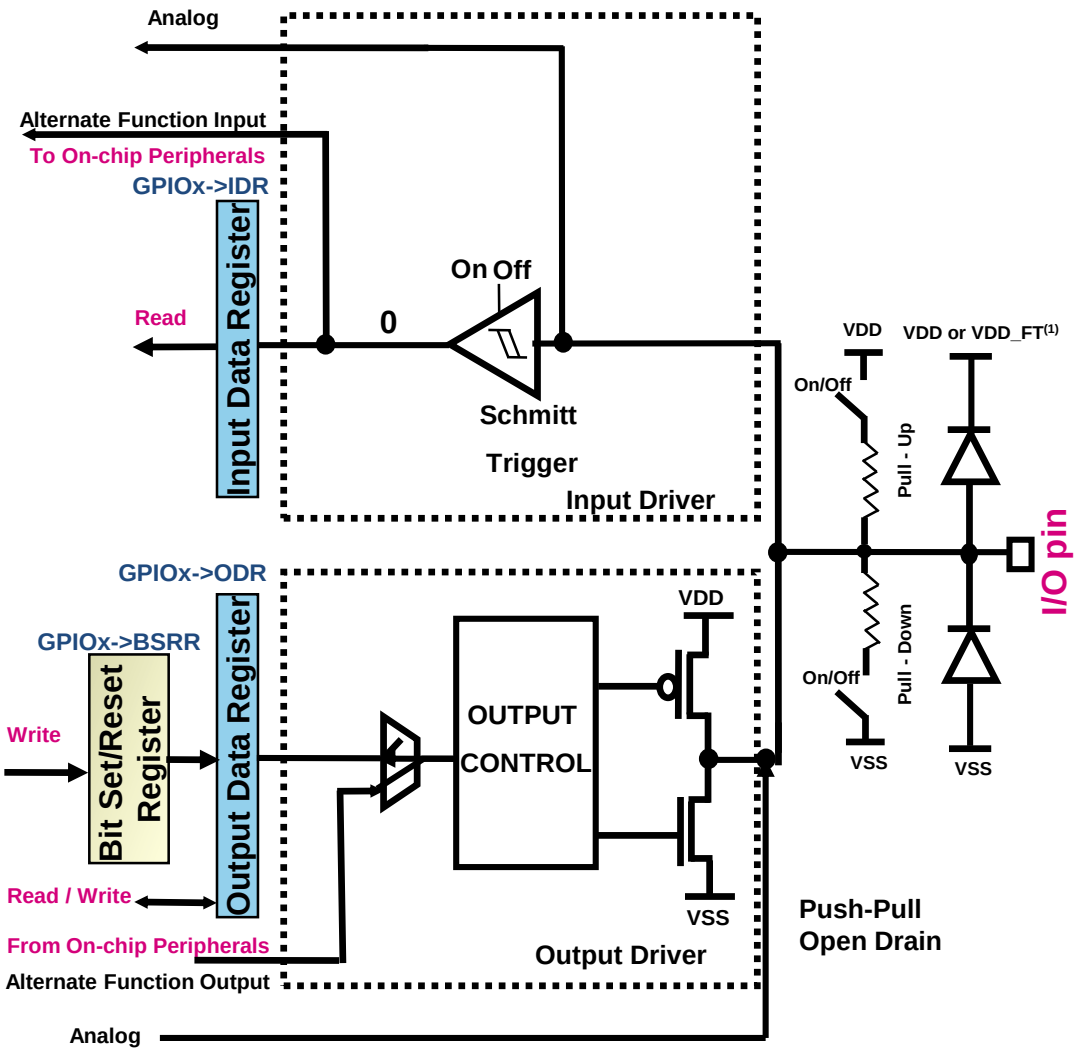
General-purpose I/Os (GPIO)

GPIO features

- Up to 51 multifunction bi-directional I/O ports available: 83% IO ratio
- Most standard I/Os are 5V tolerant*
- All Standard I/Os are shared in 6 ports (GPIOA..GPIOF)
- Atomic Bit Set and Bit Reset using BSRR register
- GPIO connected to AHB bus: max toggling frequency = $f_{AHB}/2$ = 16 MHz
- Configurable Output Speed up to 40 MHz
- Ultralow leakage per I/O: 50 nA (maximum @ max temperature)
- Up to 51 GPIOs can be set-up as external interrupt (up to 16 lines at time) able to wake-up the MCU from low power STOP mode.
- Three I/Os (PA0, PC13 and PE6) can be used as Wake-Up sources from STANDBY mode, and as Tamper Pin (PA0, PC13 and PE6) to reset back-up registers
- One I/O (PC13) can be set-up TimeStamp, RTC Alarm Output, RTC Wakeup Output or RTC Clock output

GPIO Configuration Modes

MODER(i) [1:0]	OTYPER(i) [1:0]	PUPDR(i) [1:0]	I/O configuration
01	0	0 0 0 1 1 0	Output Push Pull Output Push Pull with Pull-up Output Push Pull with Pull-down
	1	0 0 0 1 1 0	Output Open Drain Output Open Drain with Pull-up Output Open Drain with Pull-down
10	0	0 0 0 1 1 0	Alternate Function Push Pull Alternate Function PP Pull-up Alternate Function PP Pull-down
	1	0 0 0 1 1 0	Alternate Function Open Drain Alternate Function OD Pull-up Alternate Function OD Pull-down
10	x	0 0 0 1 1 0	Input floating Input with Pull-up Input with Pull-down
11	x	x	Analog mode

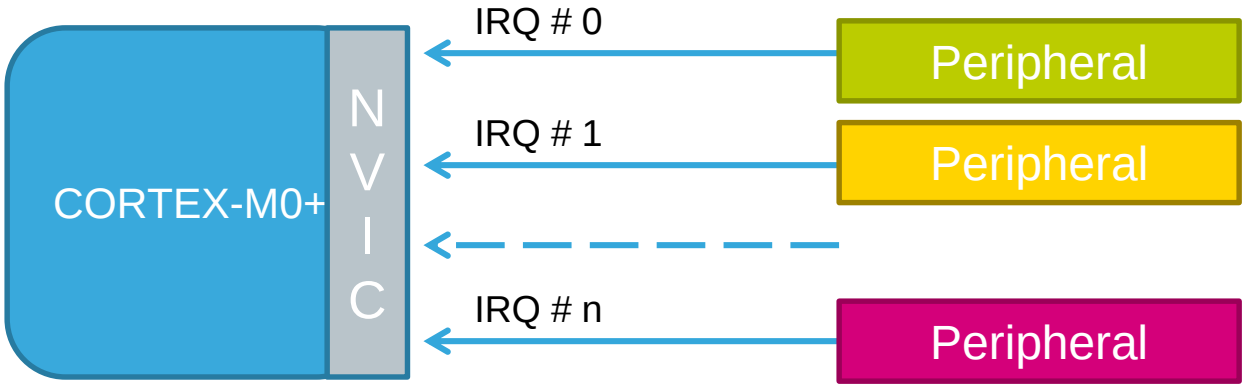


* In output mode, the I/O speed is configurable through OSPEEDR register: 400kHz, 2MHz, 10MHz or 40 MHz

(1) VDD_FT is a potential specific to five-volt tolerant I/Os and different from VDD.

life.augmented

Cortex-M0+ Interrupt processing



- Set Interrupt Priority
  - 4 levels of priority
    - Level 0 (highest) – Level 3 (lowest)
- Enable Interrupt Line at the NVIC side.
- When an interrupt occurs, the NVIC will check the priority, if highest, then the core will fetch and execute the service routine.
- If interrupts have the same software priority, the NVIC will resolve the conflict through their hardware priority as defined by their position in the Interrupt Vector Table. Refer to NVIC section in the STM32L0 Reference Manual.

- Enable Interrupt request generation at the Peripheral side.

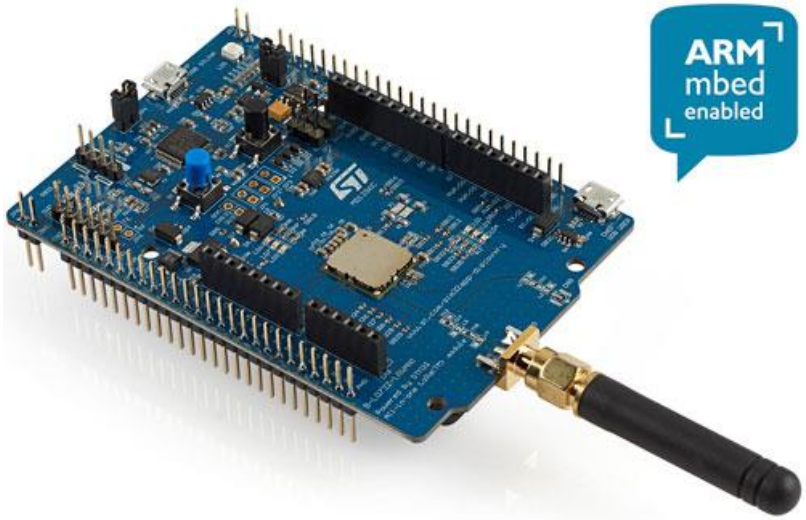
  Example:
  - GPIO EXTI (Rising\falling edge detect)
  - USART RxNE (Receive not empty)
  - USART TxE (Transmit Empty)

STM32LoRa discovery kit Overview

B-L072Z-LRWAN1 Overview

This Discovery kit features an all-in-one open module *CMWX1ZZABZ-091* (by Murata). The module is powered by an STM32L072CZ and an SX1276 transceiver. The transceiver features the LoRa®long-range modem, providing ultra-long-range spread spectrum communication and high interference immunity, minimizing current consumption. Since CMWX1ZZABZ-091 is an open module, user has access to all STM32L072 peripherals such as ADC, 16-bit timer, LP-UART, I2C, SPI and USB 2.0 FS (supporting BCD and LPM).

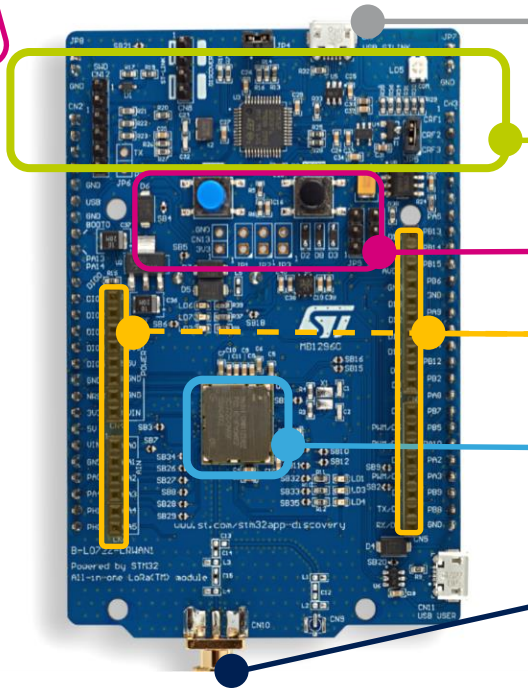
- **Key Features**
  - CMWX1ZZABZ-091 LoRa® module (Murata)
  - SMA and U.FL RF interface connectors
  - Including 50 Ohm SMA RF antenna
  - On-board ST-LINK/V2-1 supporting USB re-enumeration capability
  - USB ST-LINK functions:
  - Board power supply:
  - Through USB bus or external VIN /3.3 V supply voltage or batteries
  - 3xAAA-type-battery holder for standalone operation
  - 4 general-purpose LEDs
  - 2 push-buttons (user and reset)
  - Arduino™ Uno V3 connectors
  - ARM® mbed™ (see http://mbed.org)



New Hardware tool

B-L072Z-LRWAN1: STM32 and LoRa® Discovery kit

Available now!



Flexible board power supply :
through USB or external source

Integrated ST-Link/V2-1:
mass storage device flash programming

2 push buttons, 2 color Leds,
Jumper settings

Arduino™ extension connectors :
easy access to add-ons

Murata module

SMA Antenna connector

Arduino Header

Labels usable in code

PX_Y MCU pin without conflict

PX_Y MCU pin connected to other components
See **PeripheralPins.c** (link below) for more information

XXX Arduino connector names (A0, D1, ...)

XXX LEDs and Buttons (LED_1, USER_BUTTON, ...)

Labels not usable in code (for information only)

XXX Serial pins (USART/UART)

XXX SPI pins

XXX I2C pins

XXX PWMOut pins (TIMER n/c[N])
n = Timer number c = Channel
N = Inverted channel

XXX AnalogIn (ADC) and AnalogOut pins (DAC)

XXX CAN pins

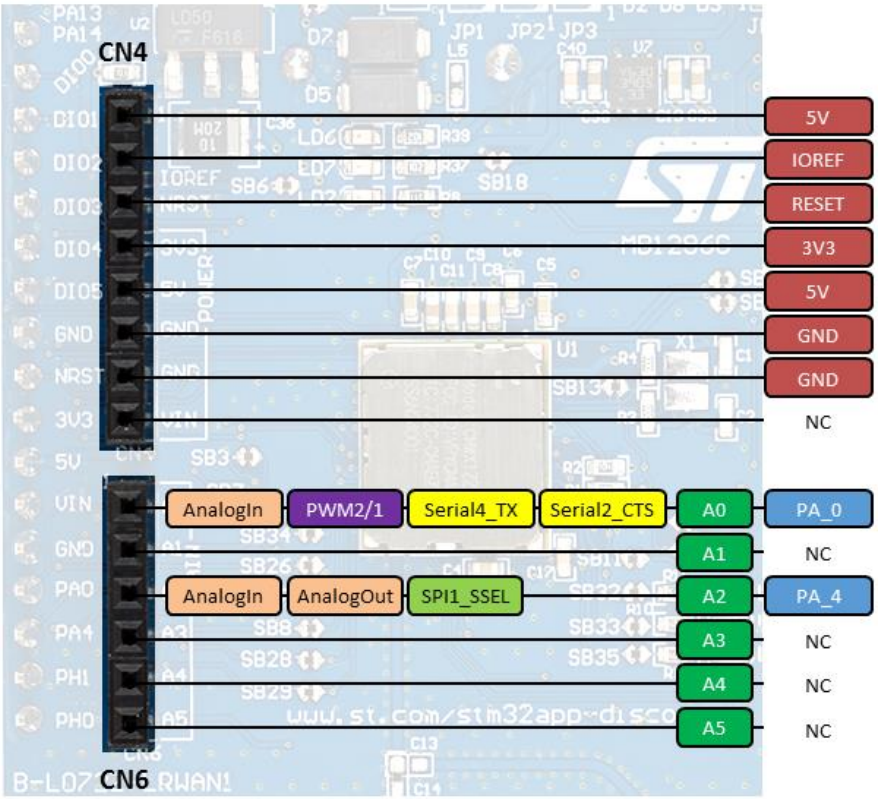
XXX Power and control pins (3V3, GND, RESET, ...)

Arduino Header

life.augmented

DISCO-L072CZ-LRWAN1

ARDUINO HEADER
(top left side)

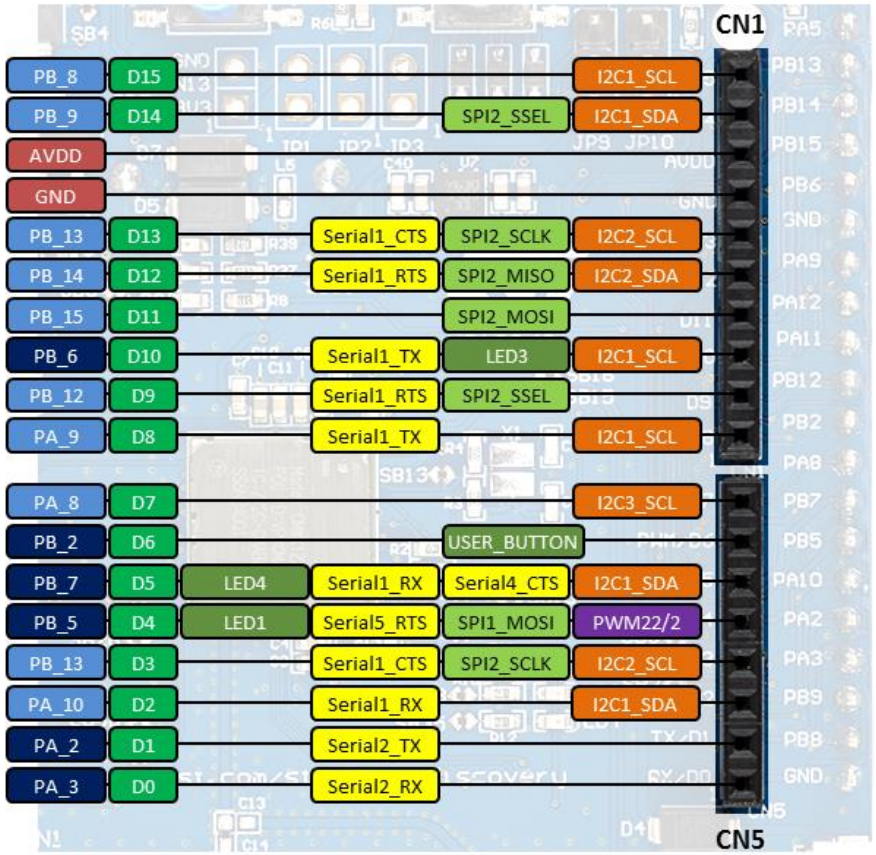


Arduino Header

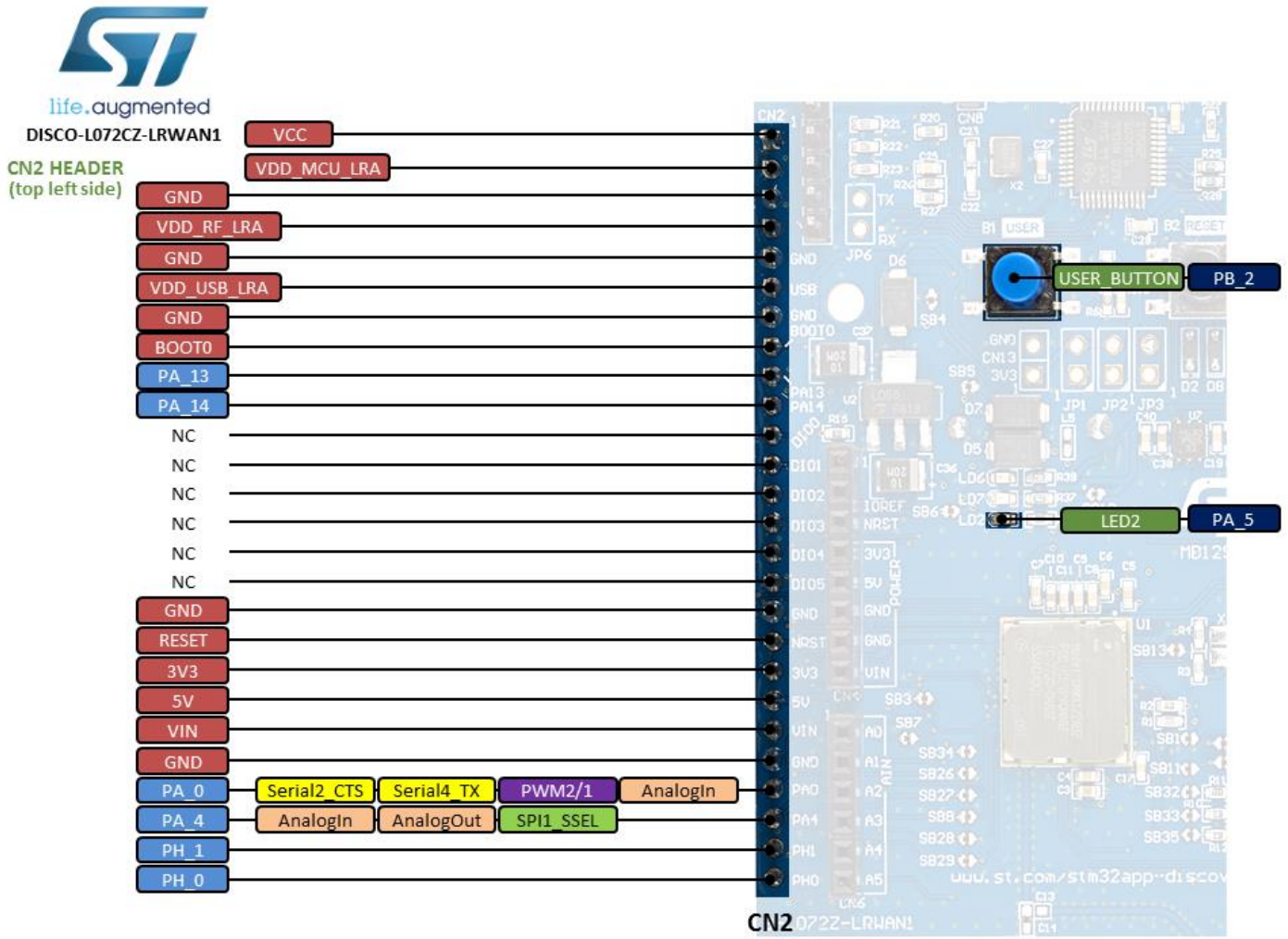
life.augmented

DISCO-L072CZ-LRWAN1

ARDUINO HEADER
(top right side)



Arduino Header

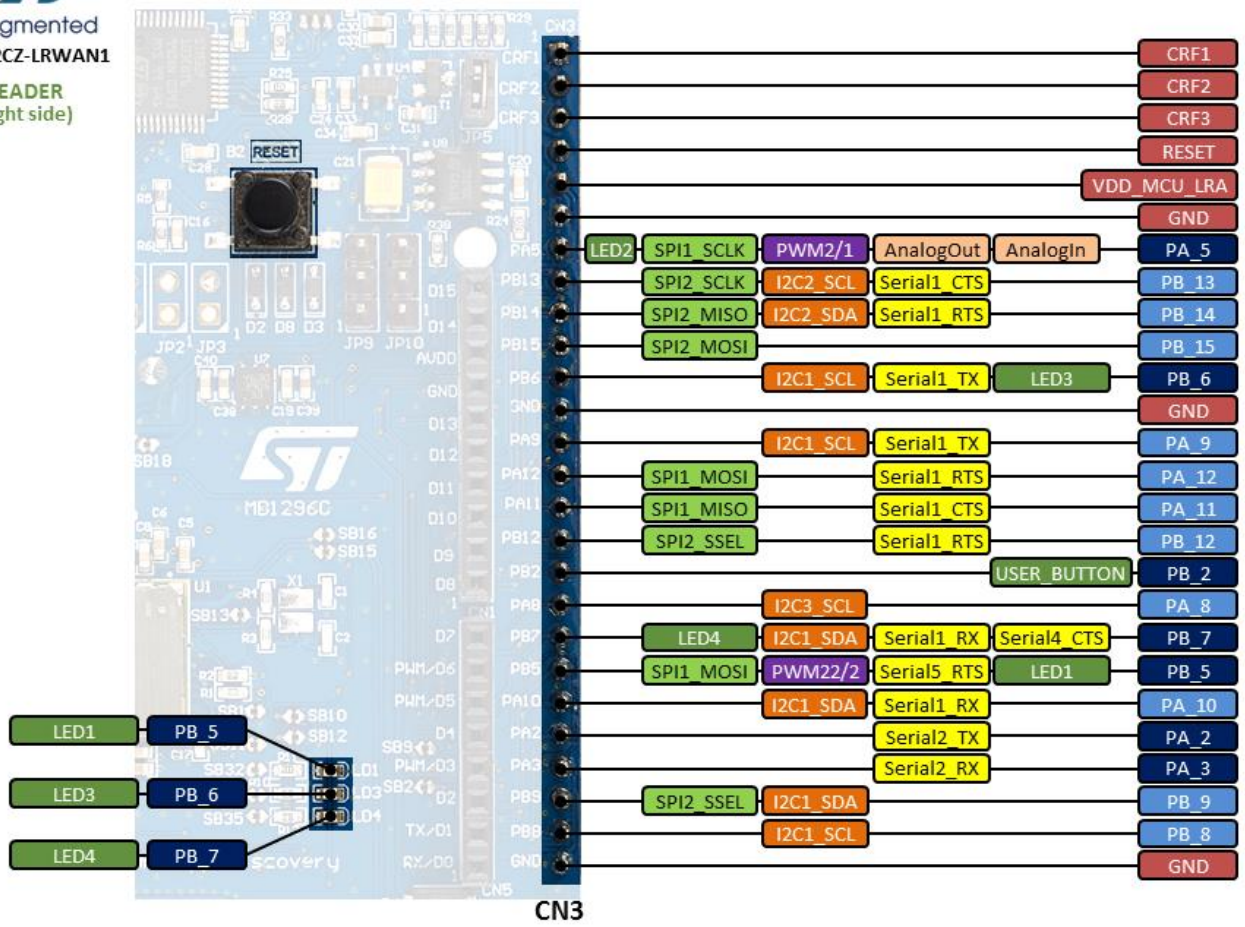


Arduino Header

life.augmented

DISCO-L072CZ-LRWAN1

CN3 HEADER
(top right side)



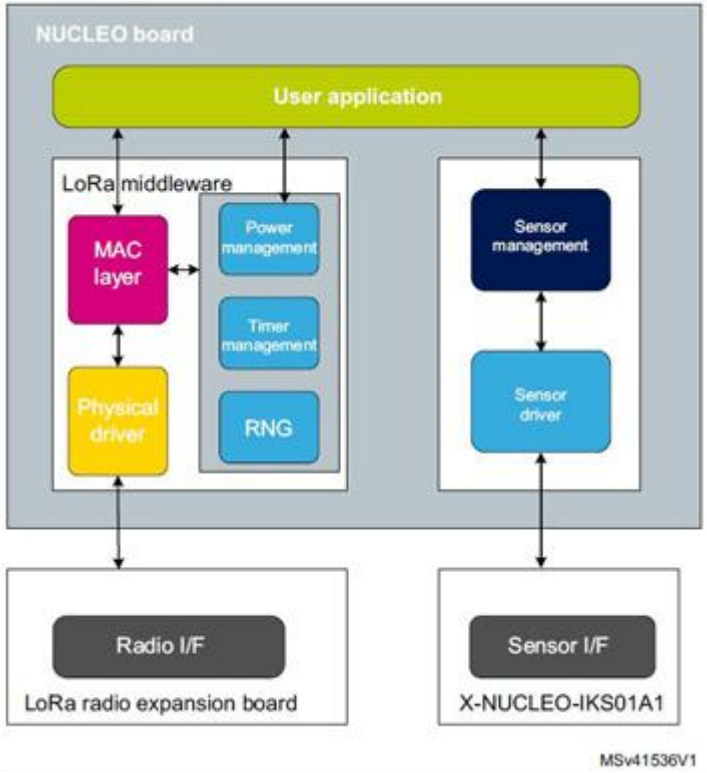
LoRa Device Firmware Library Overview

I-CUBE-LRWAN: LoRaWAN software expansion for STM32Cube

The I-CUBE-LRWAN software expansion package consists of a set of libraries and application examples for STM32L0, STM32L1 and STM32L4 devices acting as end devices.

Downloadable from www.st.com/i-cube-lrwan

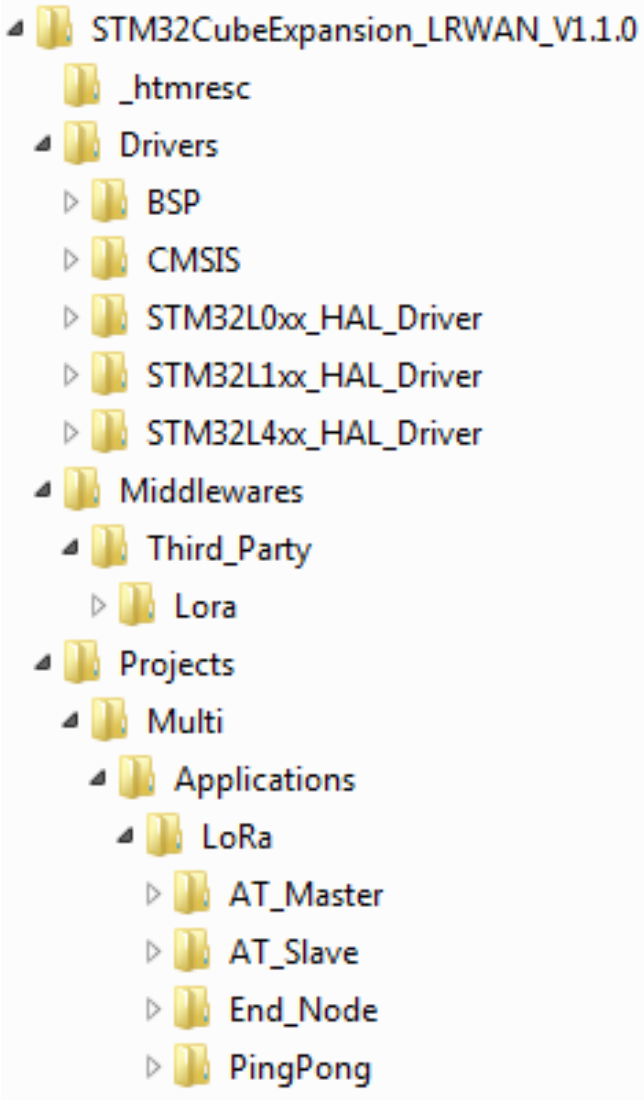
- **Key Features**
  - Compliant with the LoRa™ Alliance specification protocol named LoRaWAN™ version V1.0.2
  - Bidirectional end-devices with class A and class C protocol support
  - Class A certified (V1.0) and class C functional
  - EU 868 MHz ISM band ETSI (European telecommunications standards institute) compliant
  - EU 433 MHz ISM band ETSI compliant
  - US 915 MHz ISM band FCC (federal communications commission) compliant
  - End-device activation either via OTAA (over-the-air activation) or via ABP (activation-by-personalization)
  - Adaptive data rate support
  - LoRaWAN™ test application for certification tests included
  - Low-power optimized
  - Full STM32 portfolio compatible



- Supported LoRa radio expansion boards by Semtech
  - SX1276MB1MAS, SX1276MB1LAS and SX1272MB2DAS
- Supported ST boards
  - NUCLEO-L053R8, NUCLEO-L152RE, NUCLEO-L476RG, and B-L072Z-LRWAN1.
  - X-NUCLEO-IKS01A1 (sensor expansion board)

I-CUBE-LRWAN firmware package

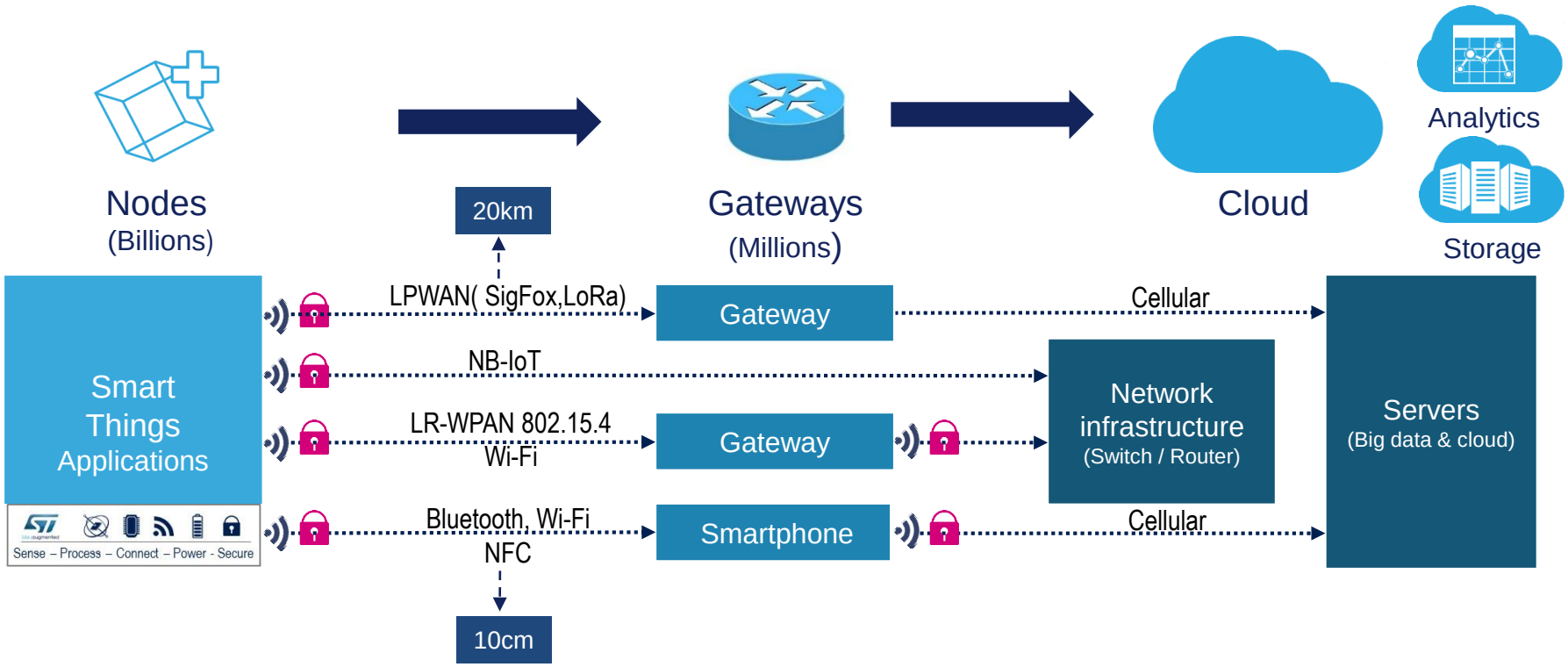
- LoRaWan 1.0.2 compliant with lots of new regions supported!. To choose your region, select one of the supported compile switch:
  - REGION_AS923
    - Brunei, Cambodia, Hong Kong, Indonesia, Japan, Laos, New Zealand, Singapore, Taiwan, Thailand, Vietnam
  - REGION_AU915
    - Australia
  - REGION_CN470\REGION_CN779
    - China
  - REGION_EU433\REGION_EU868
    - Europe
  - REGION_KR920
    - Korea
  - REGION_US915\REGION_US915_HYBRID
    - US
- Includes libraries for the MEMS and Enironmental Sensors expansion board (X-NUCLEO-IKS01A1)
- Application examples available for
  - NUCLEO-L053R8, NUCLEO-L152RE and NUCLEOL476RG using Semtech expansion boards
  - P-NUCLEO-LRWAN1, STM32 Nucleo pack for LoRa® technology
  - B-L072Z-LRWAN1, STM32 Discovery kit embedding the CMWX1ZZABZ-091 LoRa® module (Murata)
  - I-NUCLEO-LRWAN1, LoRa® expansion board for STM32 Nucleo, based on the WM-SG-SM-42 LPWAN module (USI®).



LoRaWAN™ Network Overview

The IoT Trend

Any system able to leverage the Internet and its ecosystem



LoRa® technology powered by STM32

The widest ecosystem-ever now available !

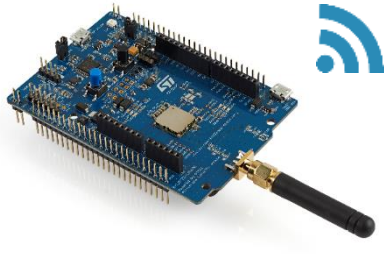
Best-in-class in ultra-low-power and Long Range

Widest HW and SW ecosystem

Easy to use

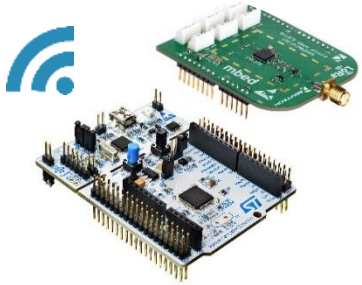
LoRa® Gateway STM32F7 based

Available Demo



B-L072Z-LRWAN1
LPWAN Discovery kit

I-NUCLEO-LRWAN1
LoRa® + Mems Shield

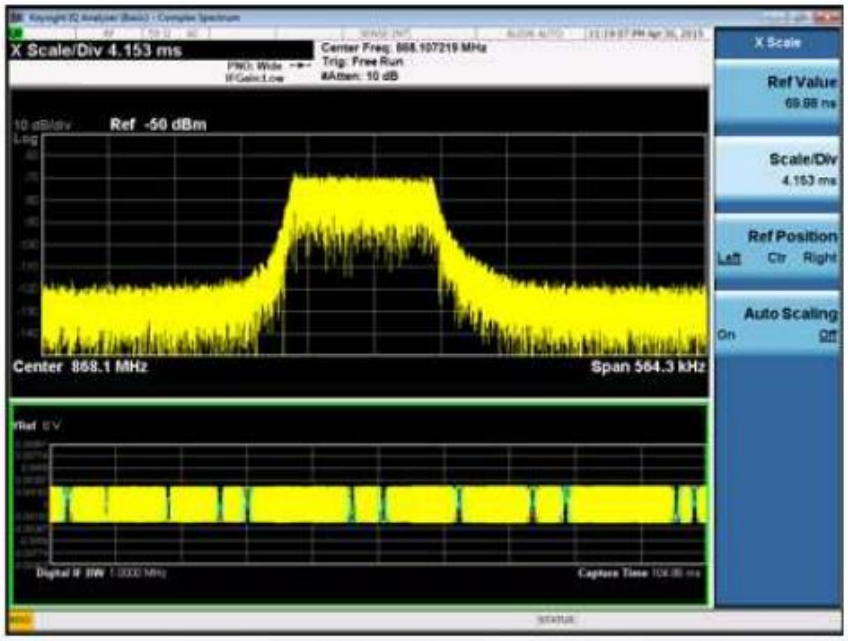


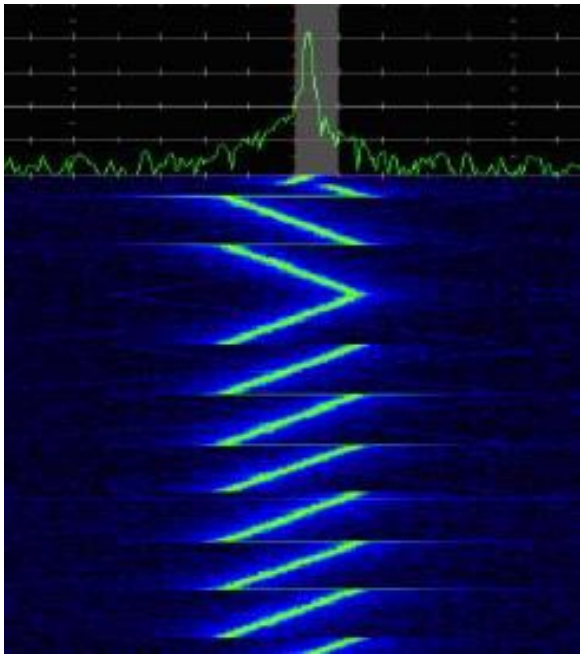
P-NUCLEO-LRWAN1
LoRa® Nucleo Pack

life.augmented

LoRa™ Technology Modulation

- LoRa™ technology is based on the Spread Spectrum Technology
- It is a Chirped Frequency Modulation





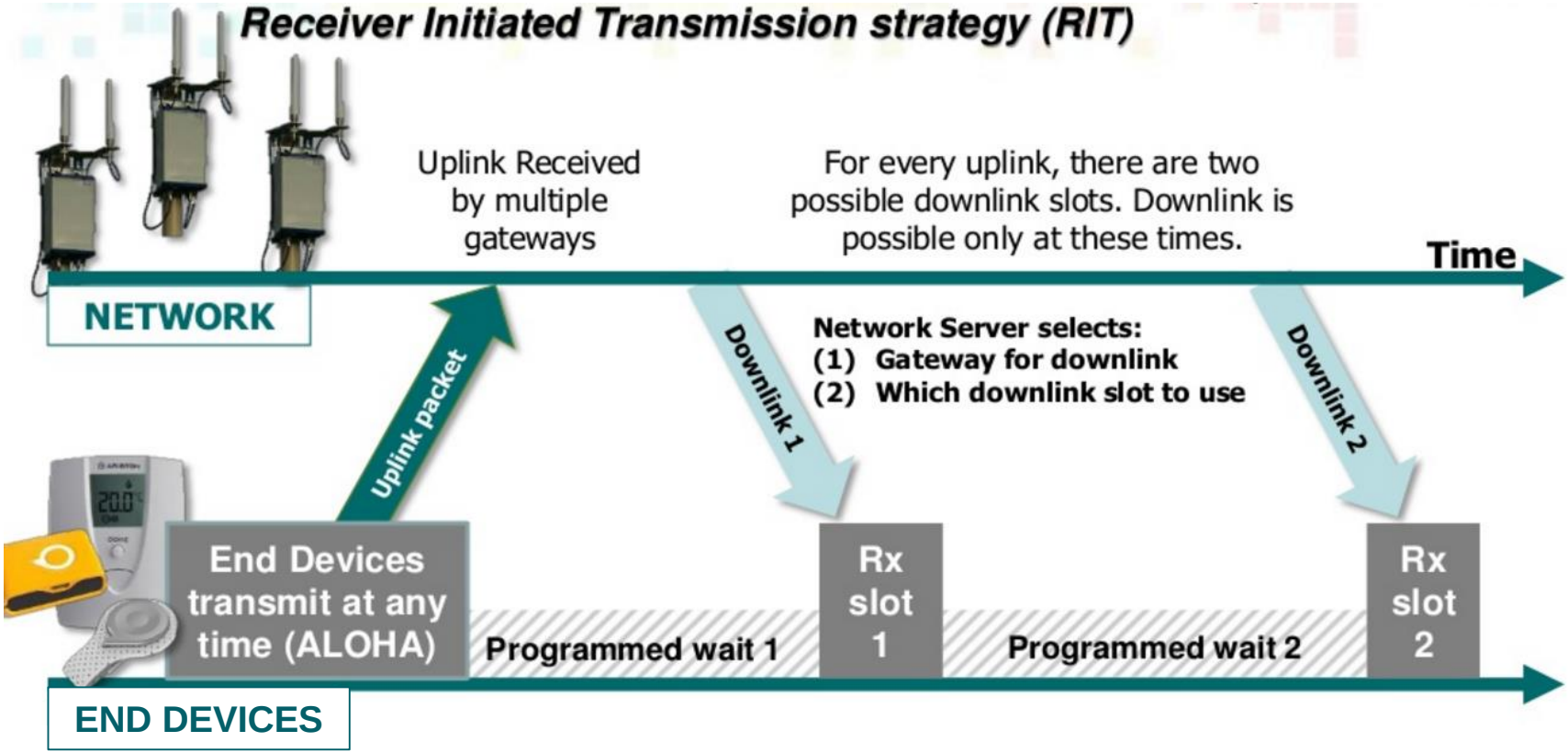
LoRaWAN™ Devices Classes

3 classes to cover all the use cases

Class name	Intended Usage	
A « All »	**Battery powered sensors**, or actuators with no latency constraint. Most energy efficient communication class. Must be supported by all devices	**Mainly uplink with two potential downlink slots after each uplink**
B « Beacon »	**Battery powered actuators** Energy efficient communication class for latency controlled downlink. Based on slotted communication synchronized with a network beacon	**Programmed downlink slots to allow control within certain latency limits**
C « Continuous »	**Mains powered actuators** Devices which can afford to listen continuously. No latency for downlink communication.	**Lowest latency command and control for less power critical device**

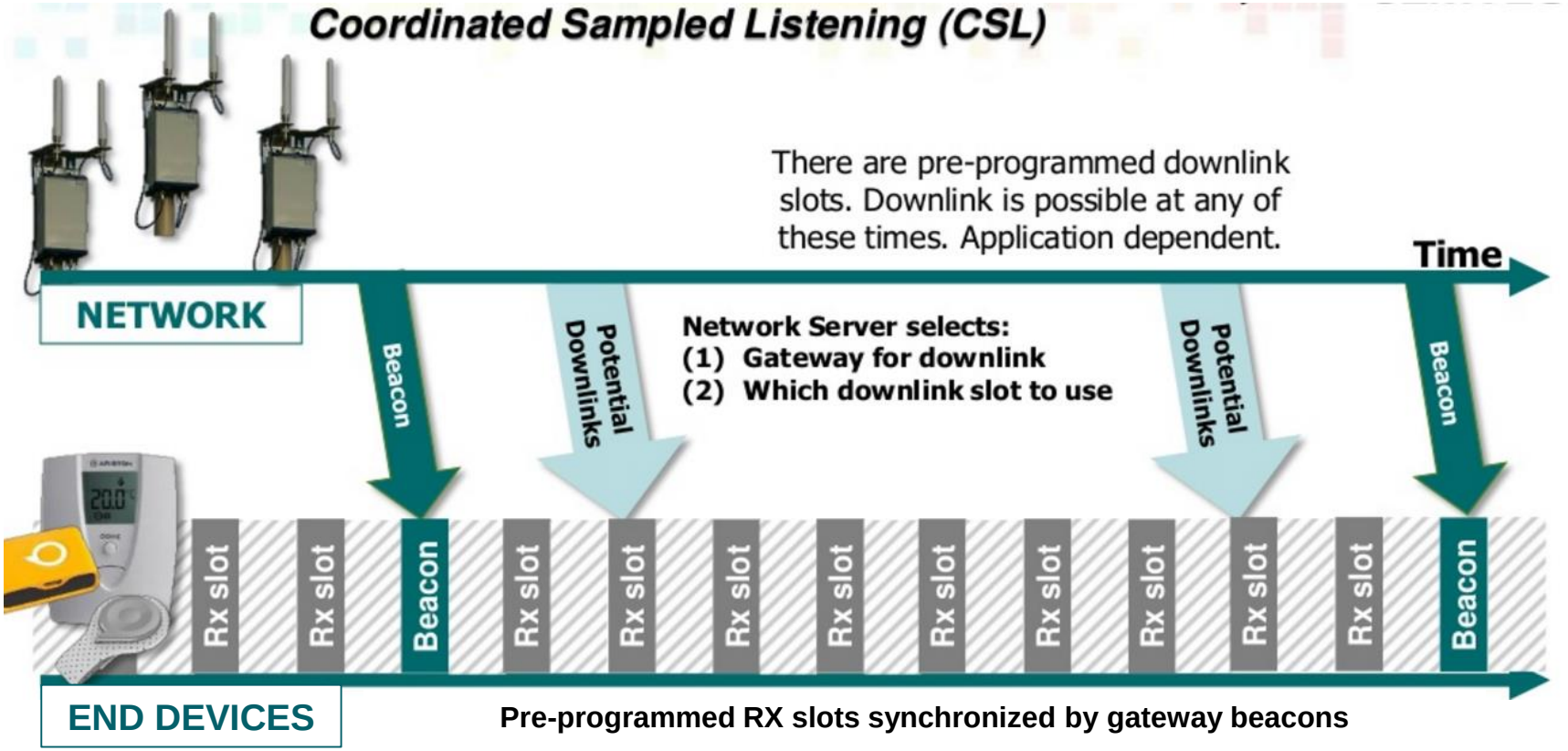
LoRaWAN™ Devices Classes

Class A – Bidirectional Communication



LoRaWAN™ Devices Classes

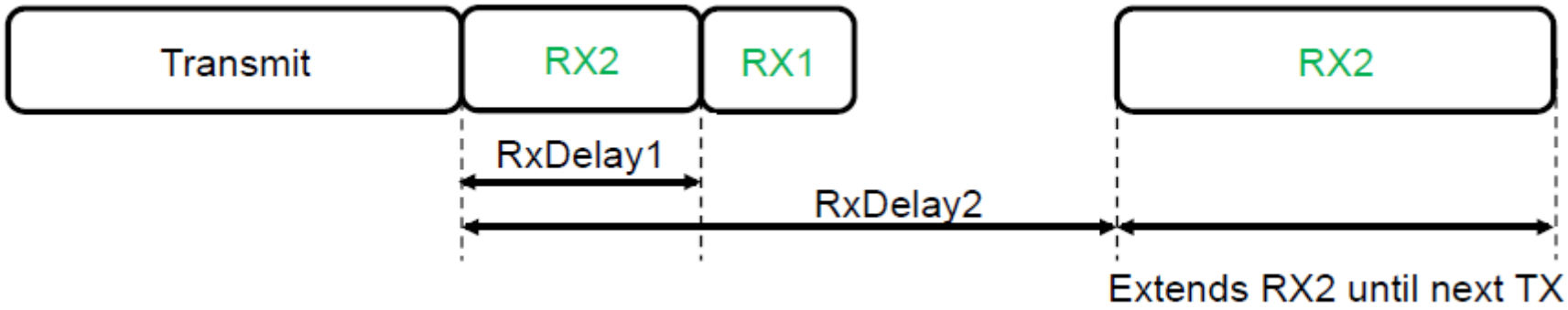
Class B – Bidirectional Communication



LoRaWAN™ Devices Classes

Class C – Bidirectional Communication

- Bidirectional communication
- Unicast and Multicast messages
- Server can initiate transmission at any time
- End-device is constantly receiving
- Pros : Lowest downlink latency
- Cons : highest power consumption (expect end-device to be mains powered)

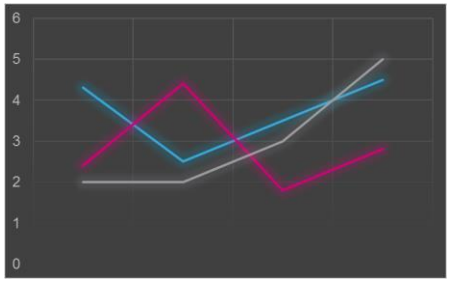


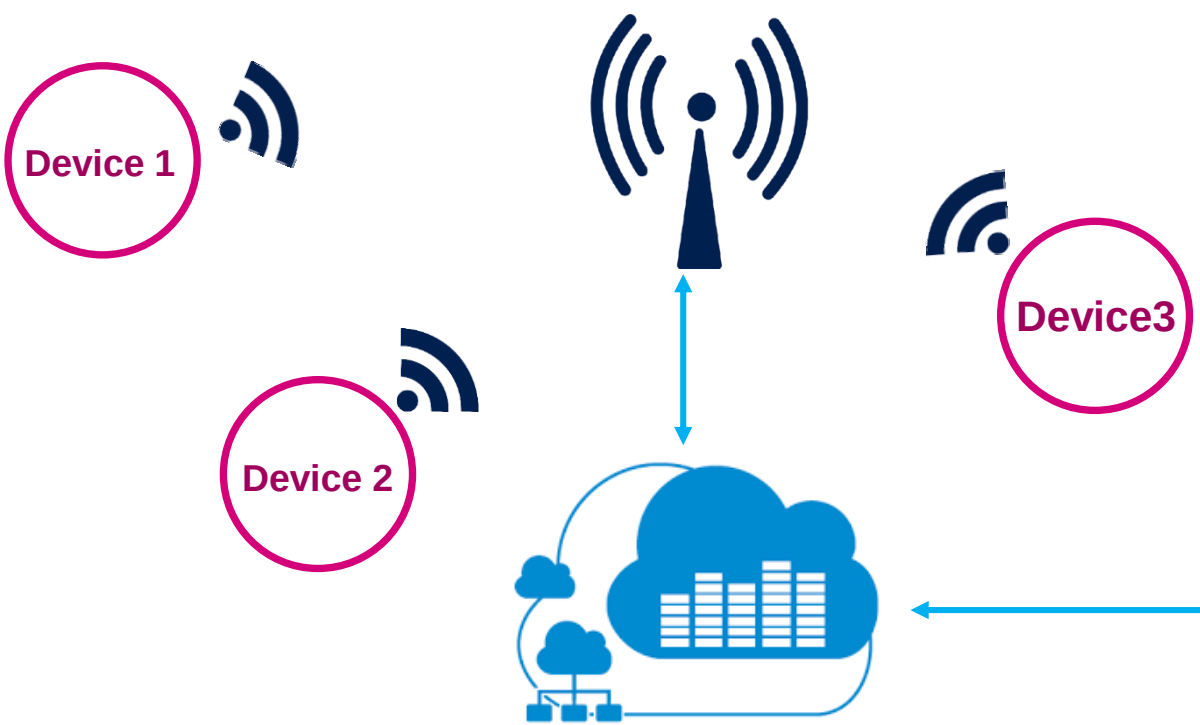
LoRaWAN™ Network

Get connected to the cloud !

Application Dashboard:

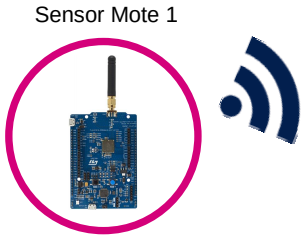
- Seamless provisioning
- Scheduling
- Rules Engine
- Visualization
- Threshold Alerts
- Remote Monitoring & Control
- Asset Tracking

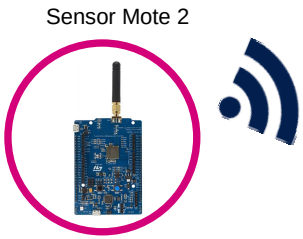




LoRaWAN™ Network Protocol

Network Topology Overview





Sub-GHz

3G / 4G / ***Ethernet*** (IP)

Ethernet (IP)

Devices

The aim is to sense, track, record, monitor any activity. They could be connected to several Gateway and many devices could be connected to the same Gateway. Devices are base on SX1276(100MHz – 1020MHz) or SX1272 (800MHz – 1020MHz) radio.

- **Each device has its own unique device address (DevAddr)**

Gateways

Is ‘’just’’ a bridge between the devices and the Network Server. The Gateway(Base station) are Telco properties (Verizon, Orange, Bouygue-T …), The Semtech radio chipset is the SX1301.

Network Servers (NS)

Will decode and authenticates data from the device. If data is addressed to the NS then it will be forwarded to the right Application Server. Other players on top of Telco may offer Network Server services.

- **Each N.S has its own and unique Network Session Key (NwkSKey)**

Application Servers (AS)

Runs the application by decrypting the packets coming fro the NS. The App could run on a cloud server, PC, Phone … . Then the data are sent back to the right device via the N.S and the Gateway. Many AS can coexist in the same Network.

- **Each A.S has its own and unique Application Session Key (AppSKey)**

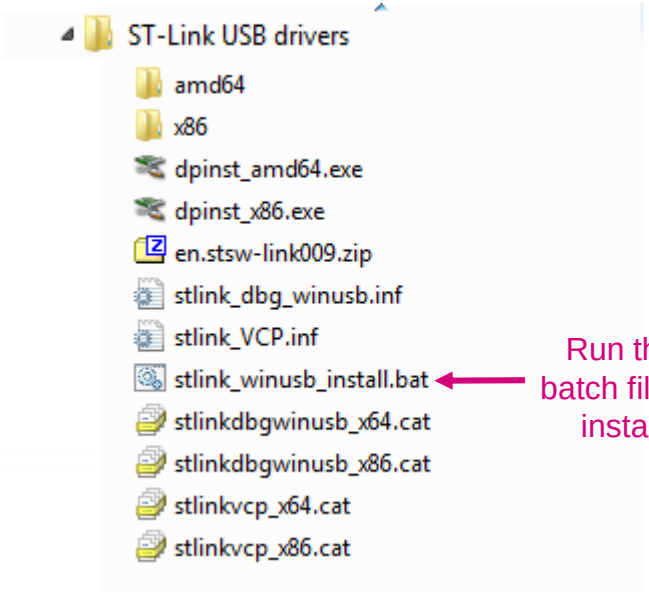
life.augmented

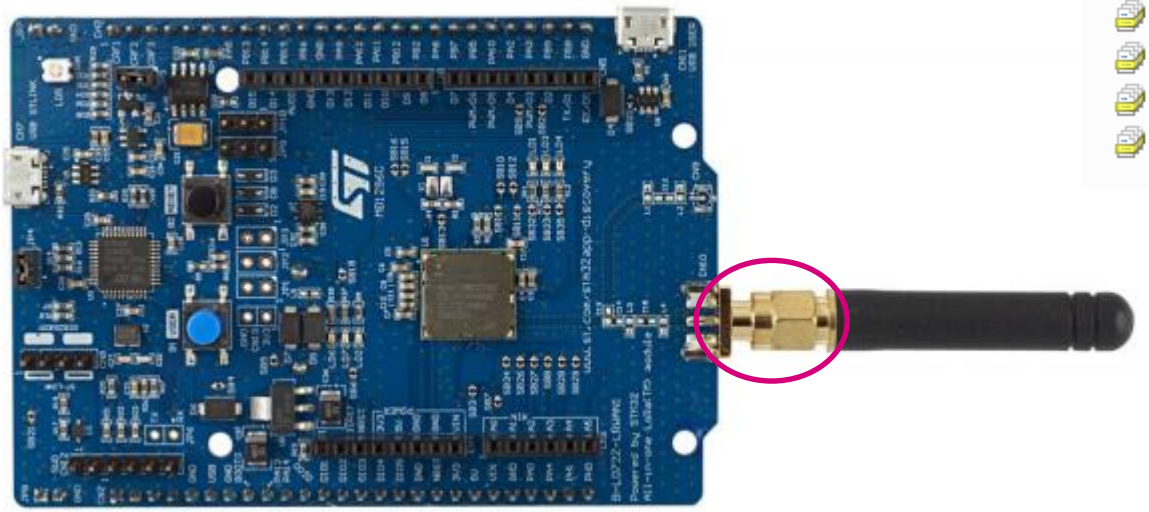
LoRa Sensor Device Setup and Reconfiguration

Sensor device power up (1/3)

LoRa sensor device setup

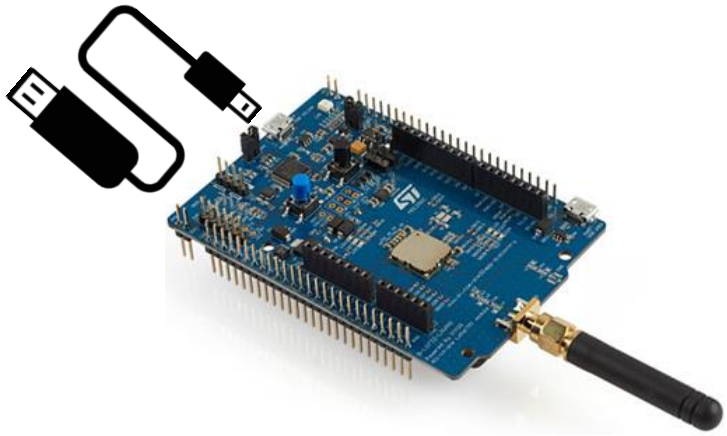
1. Make sure that the *USB drivers are installed*. (drivers downloadable from www.st.com/stlinkv2 or get from files shared to you)
2. On the B-L072Z-LRWAN1, connect the antenna to the SMA connector.



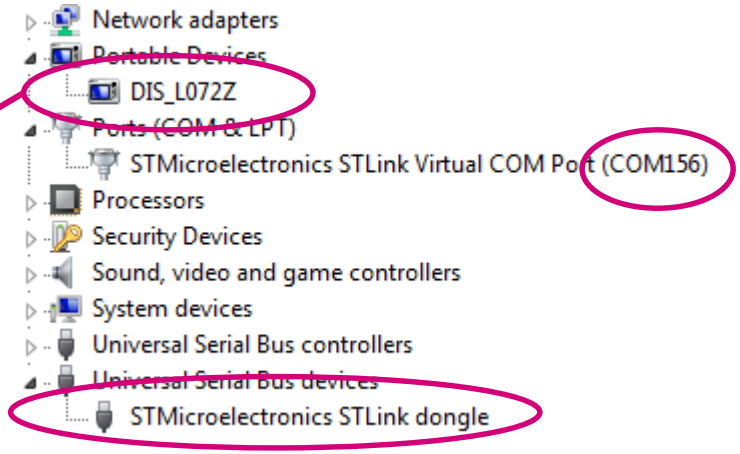


Sensor device power up (2/3)

3. Connect the Nucleo board to a PC with a USB cable type A to mini-B through USB connector CN1 to the power the board. Then green LED LD3 (PWR) and LD1 (COM) light up.
4. Allow the PC to enumerate and install the USB drivers. *Take note of the virtual COM port number assigned to the board*.

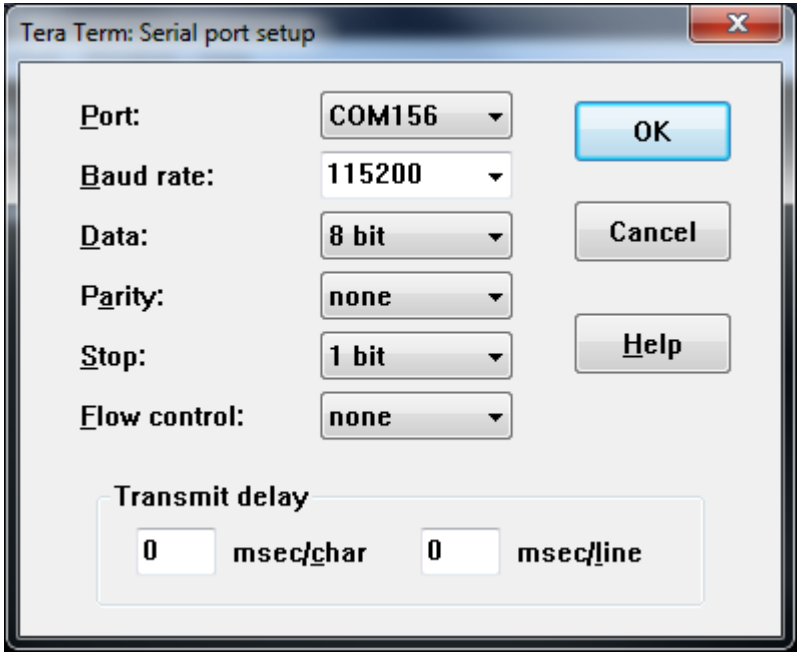


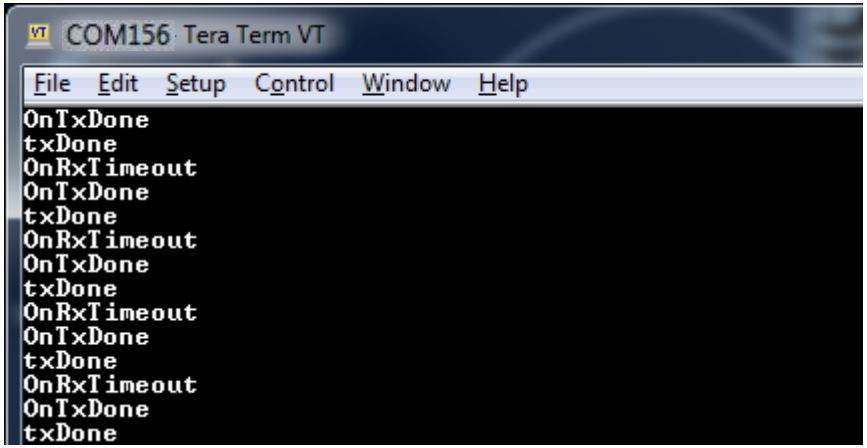
Note: The Nucleo board is also enumerated as an mbed removable storage device.



Sensor device power up (3/3)

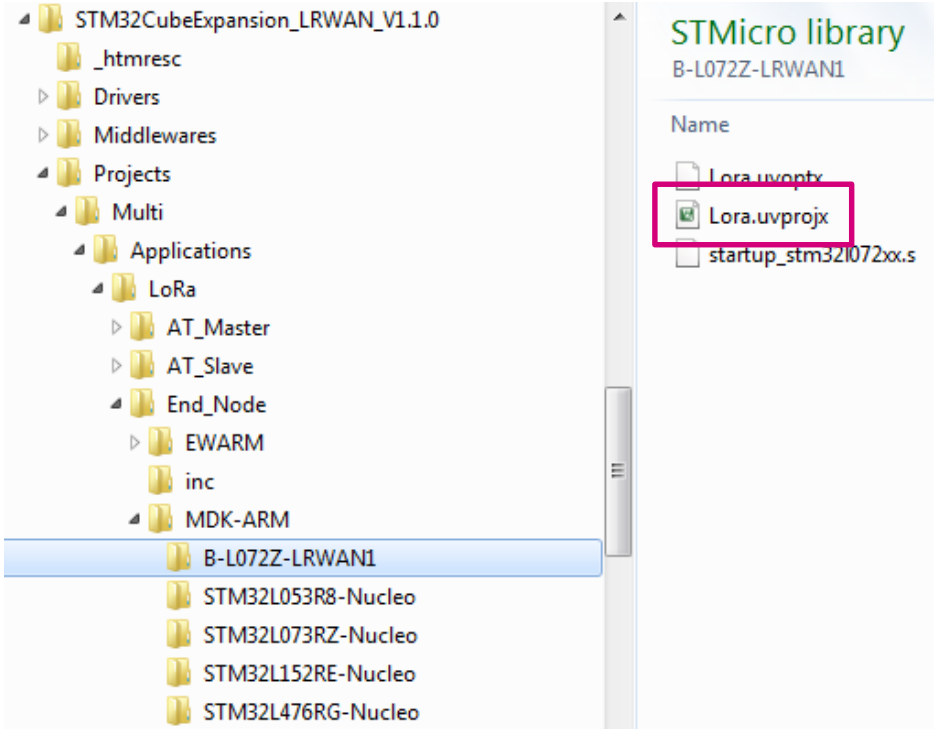
5. Open a terminal emulation software (e.g. Tera Term) and configure it with the following settings:
   - Port: (from step 4)
   - Baud rate: 115200
   - Data: 8 bit
   - Parity: none
   - Stop bit: 1 bit
6. Out of the box default demo:
   - Ping Pong demo





End Node Sample Project (1/4)

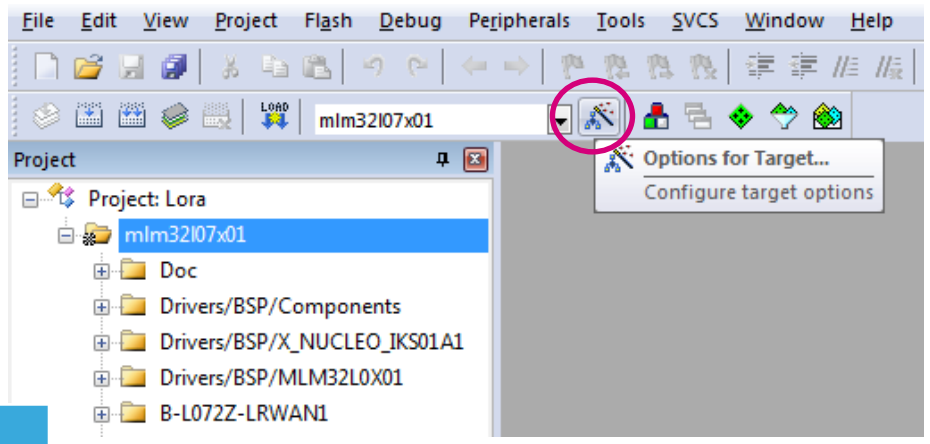
1. Download the firmware package from www.st.com/i-cube-lrwan and extract the files.
2. Go to \STM32CubeExpansion_LRWAN_V1.1.5\Projects\Multi\Applications\LoRa\End_Node
3. Open the Keil project in …\MDK-ARM\B-L072Z-LRWAN1 folder.



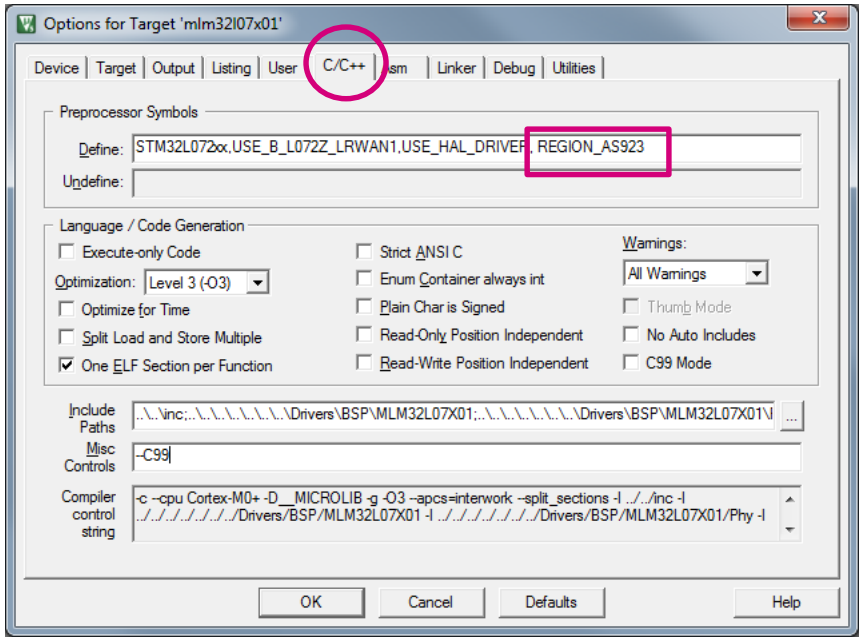
End Node Sample Project (2/4)

4. Got to Project options.
5. Go to the preprocessor settings and set to the desired frequency band. Use only one setting.

   For this workshop, use the define for REGION_AS923.



Define	Region
REGION_EU433 REGION_EU868	Europe
REGION_US915 REGION_US915_HYBRID	US
REGION_KR920	Korea
REGION_AU915	Australia
REGION_CN470 REGION_CN779	China
REGION_AS923	Brunei, Cambodia, Hong Kong, Indonesia, Japan, Laos, New Zealand, Singapore, Taiwan, Thailand, Vietnam



End Node Sample Project (3/4)

6. Edit the ***comissioning.h*** file in End_Node/inc to set the end device comissioning parameters.
   - For this workshop, set the following parameters.

     Use Activation by Personalization (ABP)

     #define OVER_THE_AIR_ACTIVATION **0**

     Set the NWKSKEY and APPSKEY (***Follow CAT's network***)

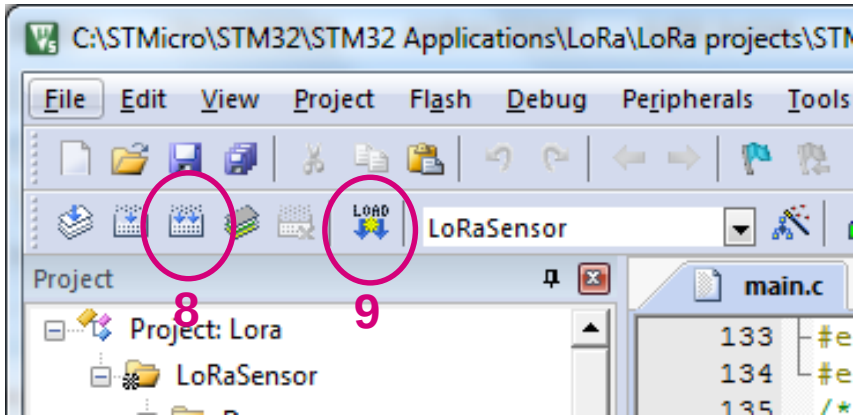
     #define LORAWAN_NWKSKEY { 0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88, 0x09, 0xCF, 0x4F, 0x3C }

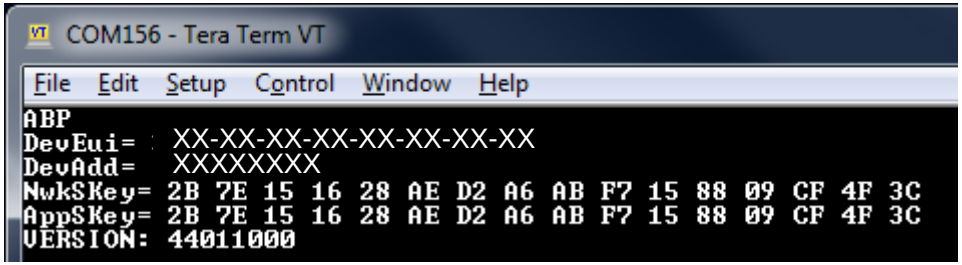
     #define LORAWAN_APPSKEY { 0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88, 0x09, 0xCF, 0x4F, 0x3C }

7. Edit the hw_conf.h file in End_Node/inc to set additional options
   - Debug switches – to show debug printf messages
   - Low power mode – enable/disable low power mode
   - Sensor enable switch – enable it if X-NUCLEO-IKS01A1/A2 (sensor expansion board) is attached

End Node Sample Project (4/4)

8. Rebuild all files and load your image into target memory
9. “Load” the firmware to the device.
5. Press the reset button B2 (black button) to view the device activation parameters which are shown only once after reset: ***DevEUI, DevAdd, NwkSKey, AppSKey***. These parameters will be used to enroll the device to the application server. See Network Setup.





Debug mode

- Debug, it can be disable and enable by comment/uncomment the define function in file “**hw_conf.h**”.

- Enable :

```
127 /* --------Preprocessor compile swicth------------ */
128 /* debug swicthes in debug.h */
129 #define DEBUG
130 //#define TRACE
```

- Disable:

```
127 /* --------Preprocessor compile swicth------------ */
128 /* debug swicthes in debug.h */
129 //#define DEBUG
130 //#define TRACE
```

- If the application need to lowest power consumption, we recommended to disable debugging mode.

Low Power mode

- Low power mode, the default source code is enable this feature in order to enter to low power mode when MCU finish send the data. But when MCU enter to low power mode the MCU will not debug and monitor the register, variable and memory.

- Enable :

```
132 /* uncomment below line to never enter lowpower modes in main.c*/
133 //#define LOW_POWER_DISABLE
```

- Disable:

```
132 /* uncomment below line to never enter lowpower modes in main.c*/
133 #define LOW_POWER_DISABLE
```

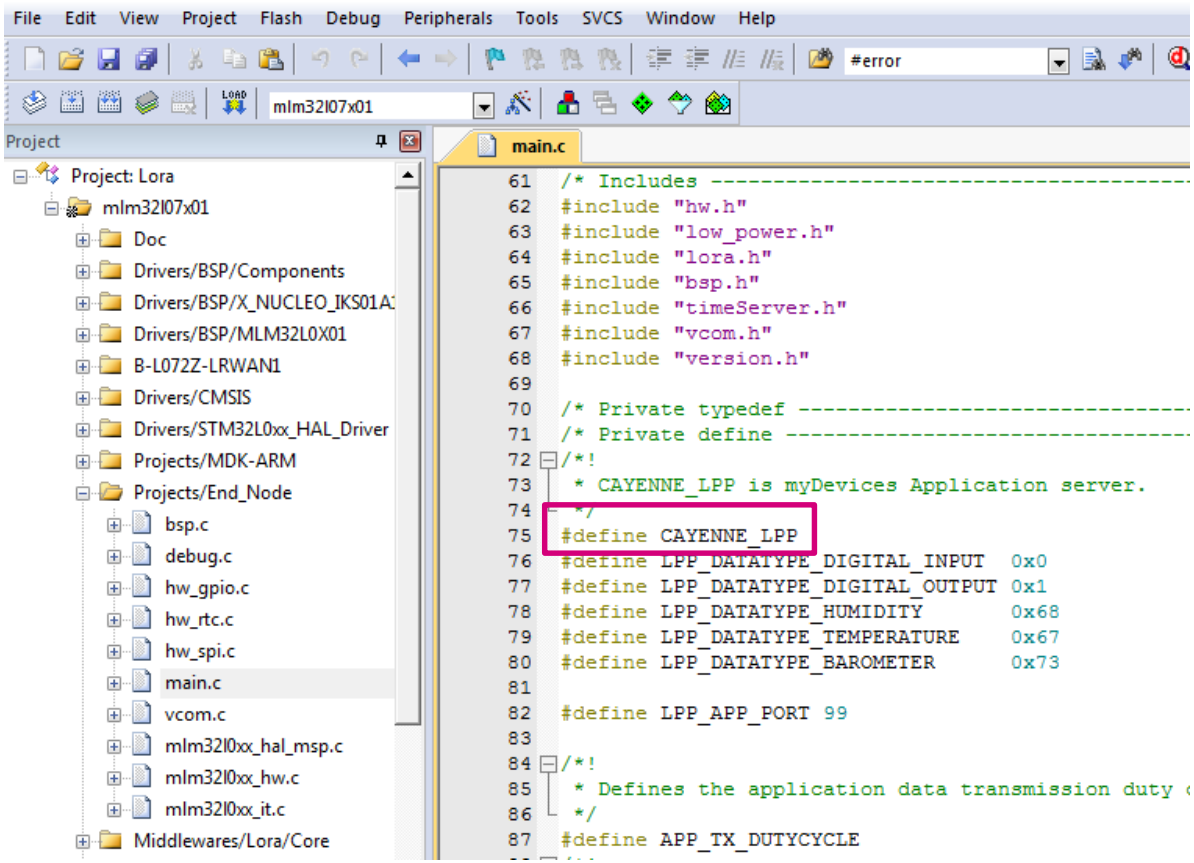
Sensor Enable

- In order to enable sensor, it has to uncomment SENSOR_ENABLE in hw_conf.h , when enable sensor the include files will be added into project

```
135 /* debug swicthes in bsp.c */
136 #define SENSOR_ENABLED
```

Cayenne LLP payload format

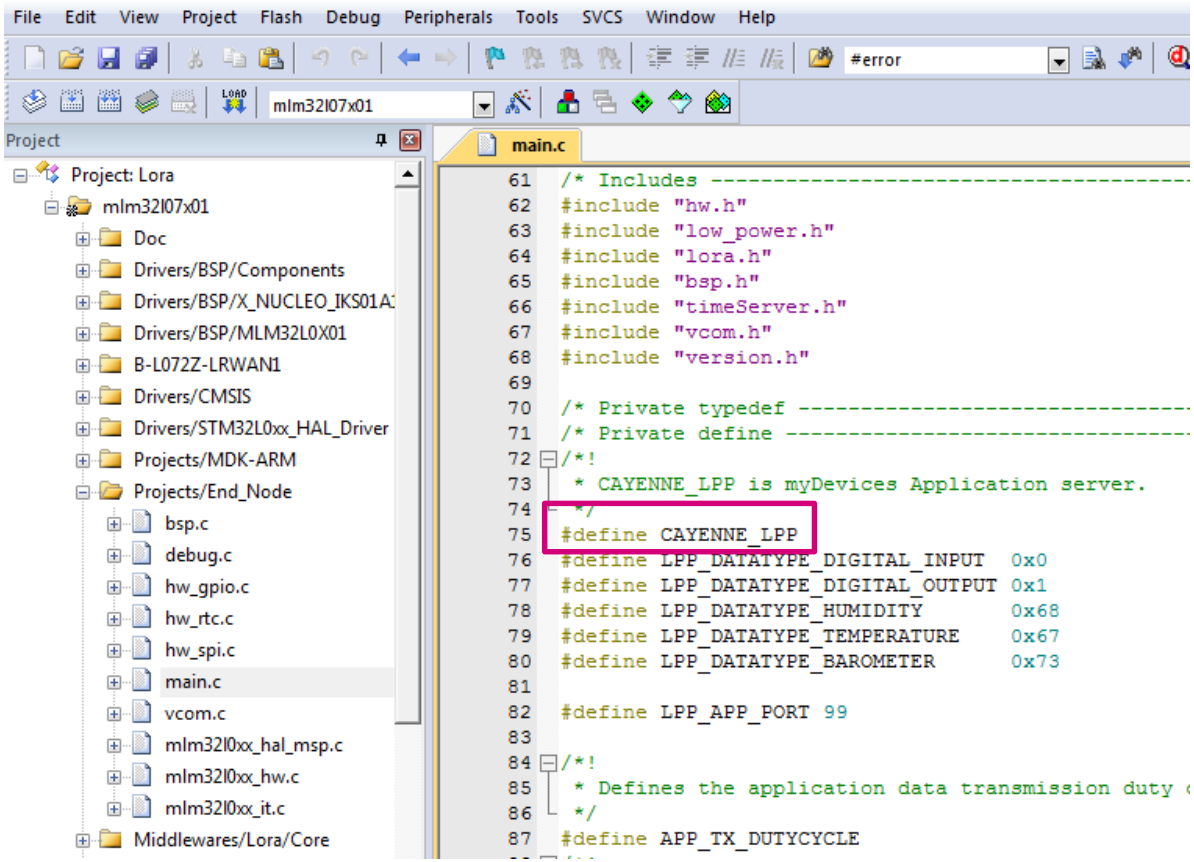
- By default, the Cayenne LLP payload format is enabled. This is to be compatible with the myDevices Cayenne Dashboard application which will be used later for the visualization of the data. (https://mydevices.com/)
- Note that for US915\US915 HYBRID, the payload is limited to 11 bytes. You may need to remove some of the Cayenne LLP payload data to comply.



Follow the instructions in myDevices Cayenne

- MyDevices Cayenne and Loriot setup
  - https://mydevices.com/cayenne/docs/#lora-loriot-network

- Make sure that the Cayenne LLP payload format is enabled during the compilation of the End_Node code for the sensor device. (Enabled by default)



life.augmented

Hand-on Sensor

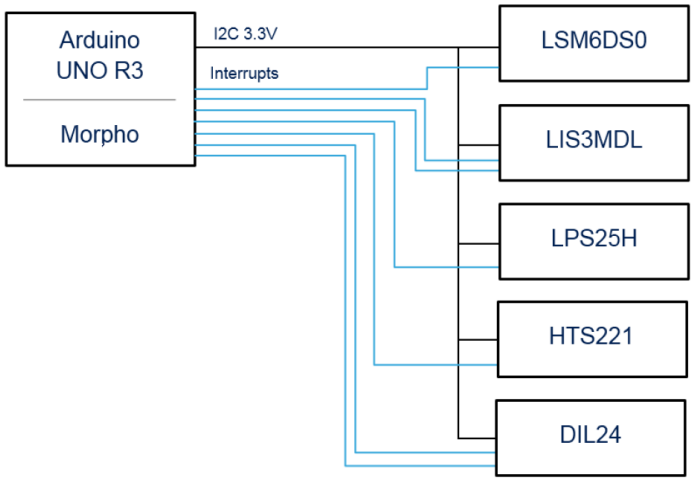
X-NUCLEO-IKS01A1/A2

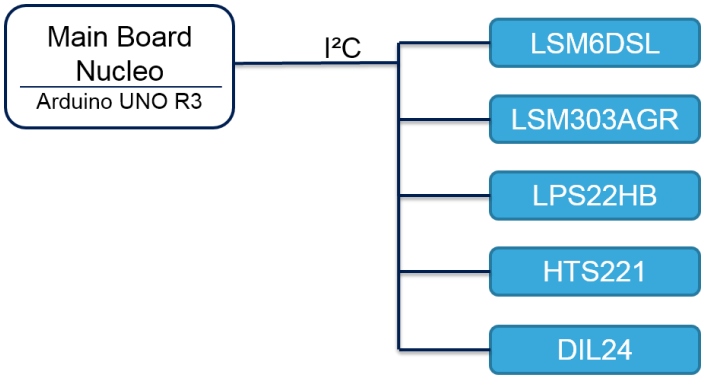
Supported Hardware

X-NUCLEO-IKS01A1

onboard

- LIS3MDL
- LSM6DS0
- LPS25HB
- HTS221





X-NUCLEO-IKS01A2

onboard

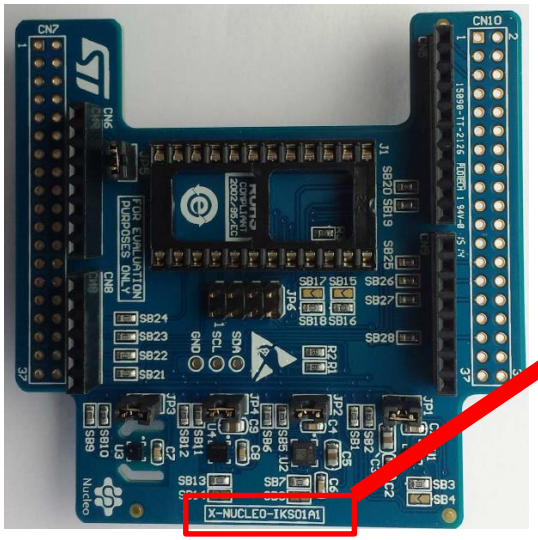
- LSM303AGR
- LSM6DSL
- LPS22HB
- HTS221

Hand on Sensor

- Enable sensor at “hw_conf.h” by uncomment line 136

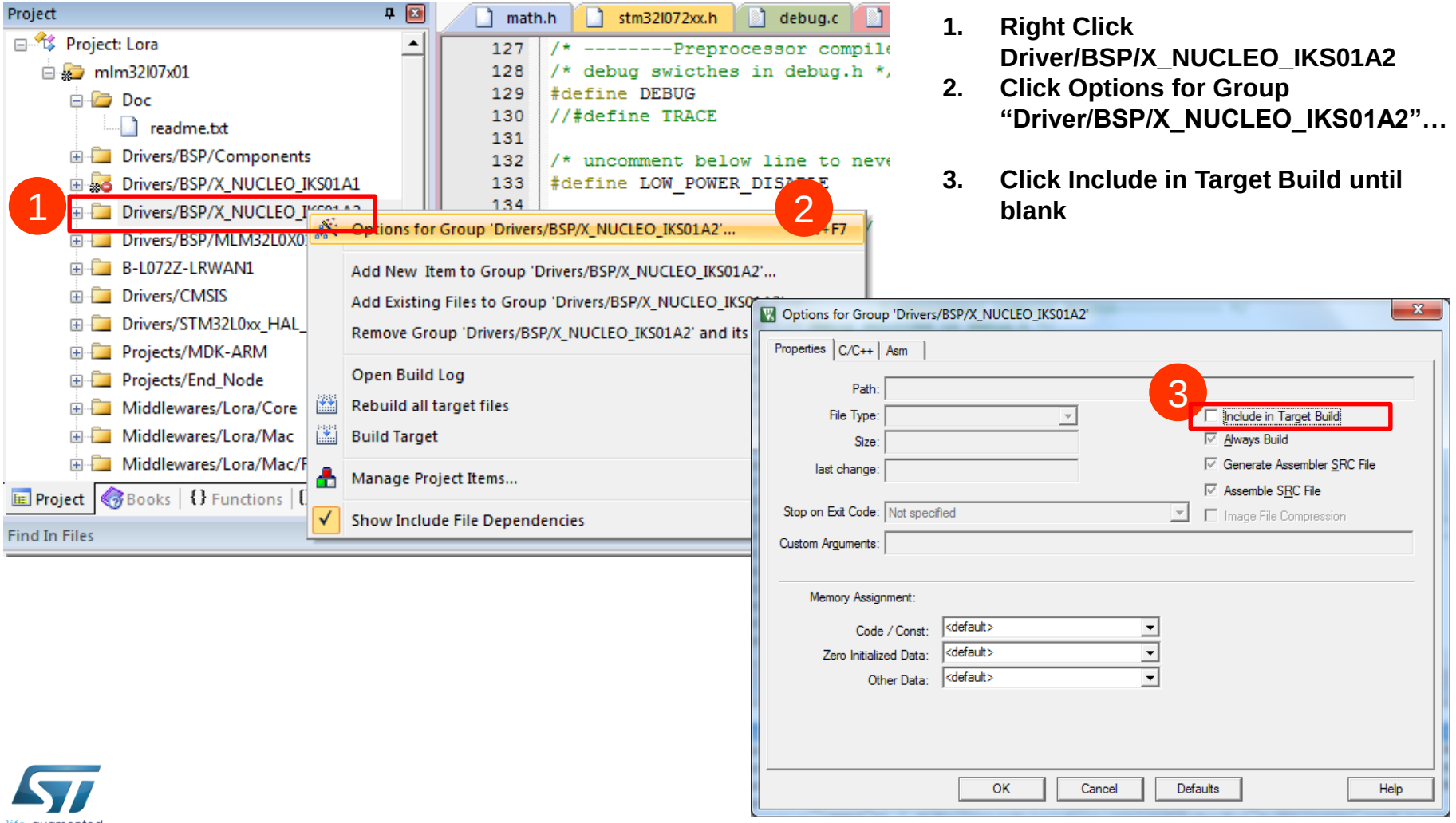
```
127 /* --------Preprocessor compile swicth------------ */
128 /* debug swicthes in debug.h */
129 #define DEBUG
130 //#define TRACE
131
132 /* uncomment below line to never enter lowpower modes in main.c*/
133 #define LOW_POWER_DISABLE
134
135 /* debug swicthes in bsp.c */
136 #define SENSOR_ENABLED
```

- Check X-NUCLEO. If X-NUCELO is IKS01A1, please follow next slide.



X-Nucleo IKS01A1

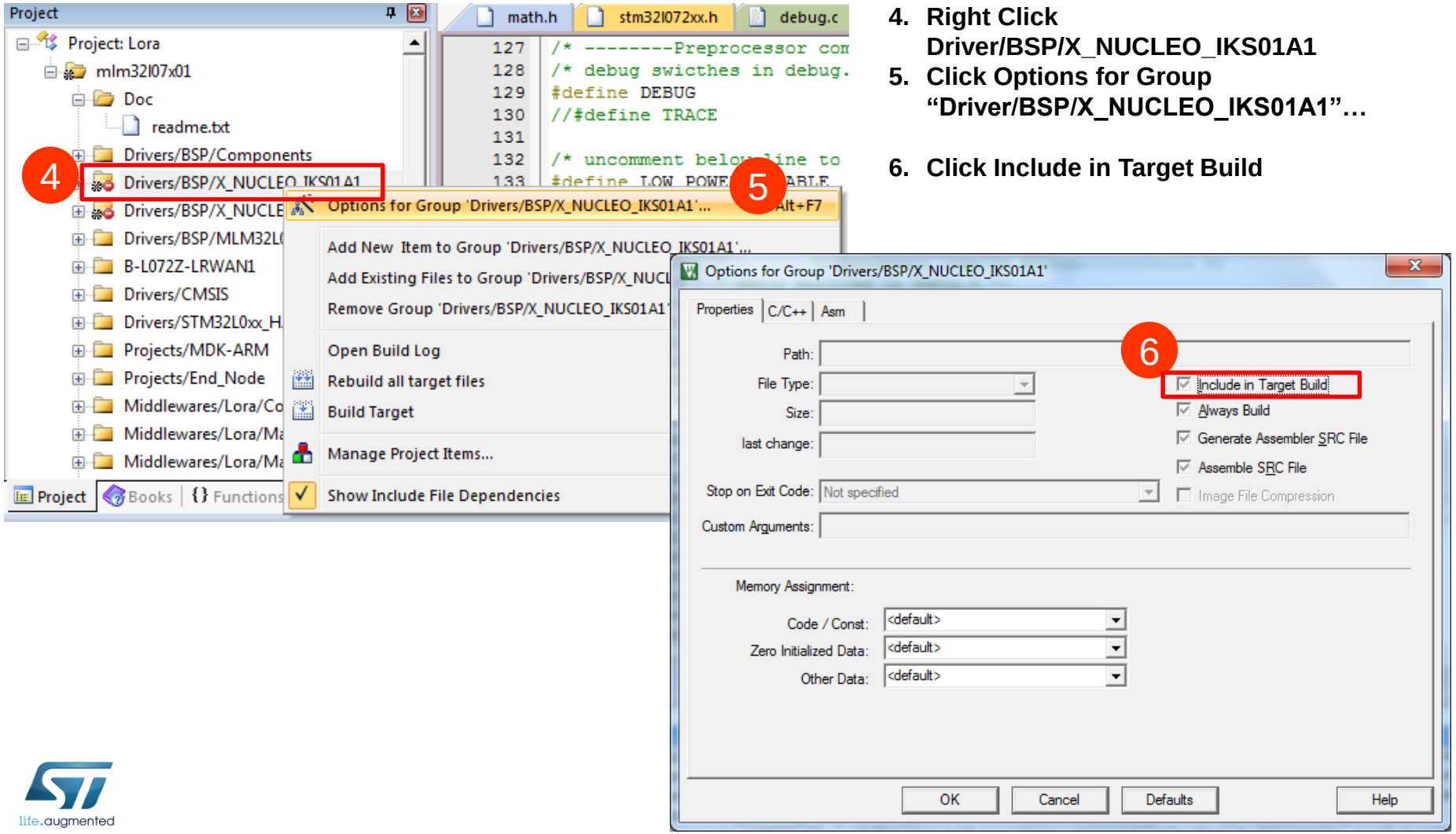
- If environment sensor board is IKS01A1, please follow



1. **Right Click Driver/BSP/X_NUCLEO_IKS01A2**
2. **Click Options for Group "Driver/BSP/X_NUCLEO_IKS01A2"…**
3. **Click Include in Target Build until blank**

X-Nucleo IKS01A1

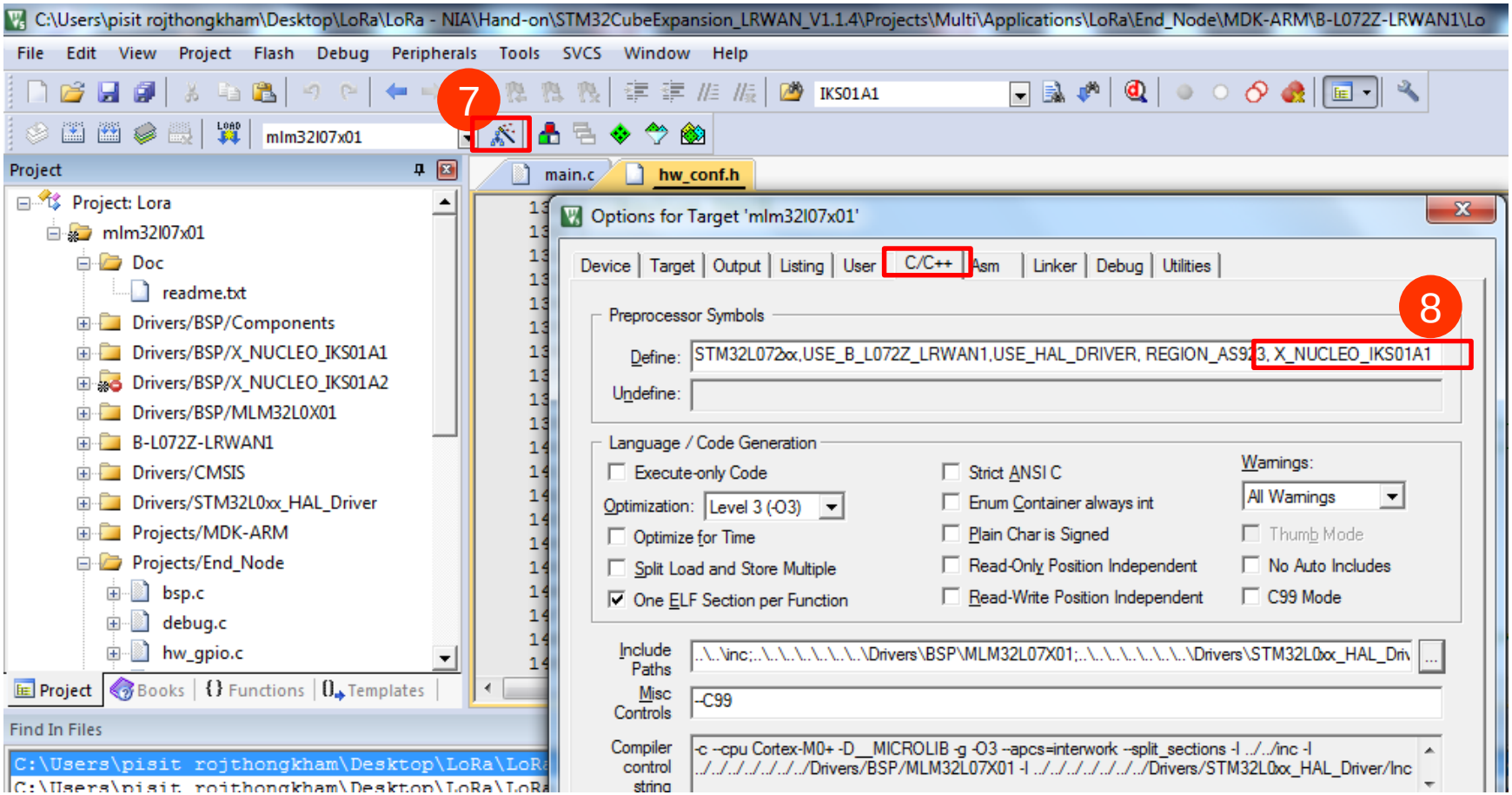
- Continue (2/3)



4. **Right Click Driver/BSP/X_NUCLEO_IKS01A1**
5. **Click Options for Group “Driver/BSP/X_NUCLEO_IKS01A1”…**
6. **Click Include in Target Build**

X-Nucleo IKS01A1

- Continue (3/3)



7. **Click Options for target … icon**
8. **Add "X_NUCLEO_IKS01A1" into C/C++ -> Define:**

Reading sensors

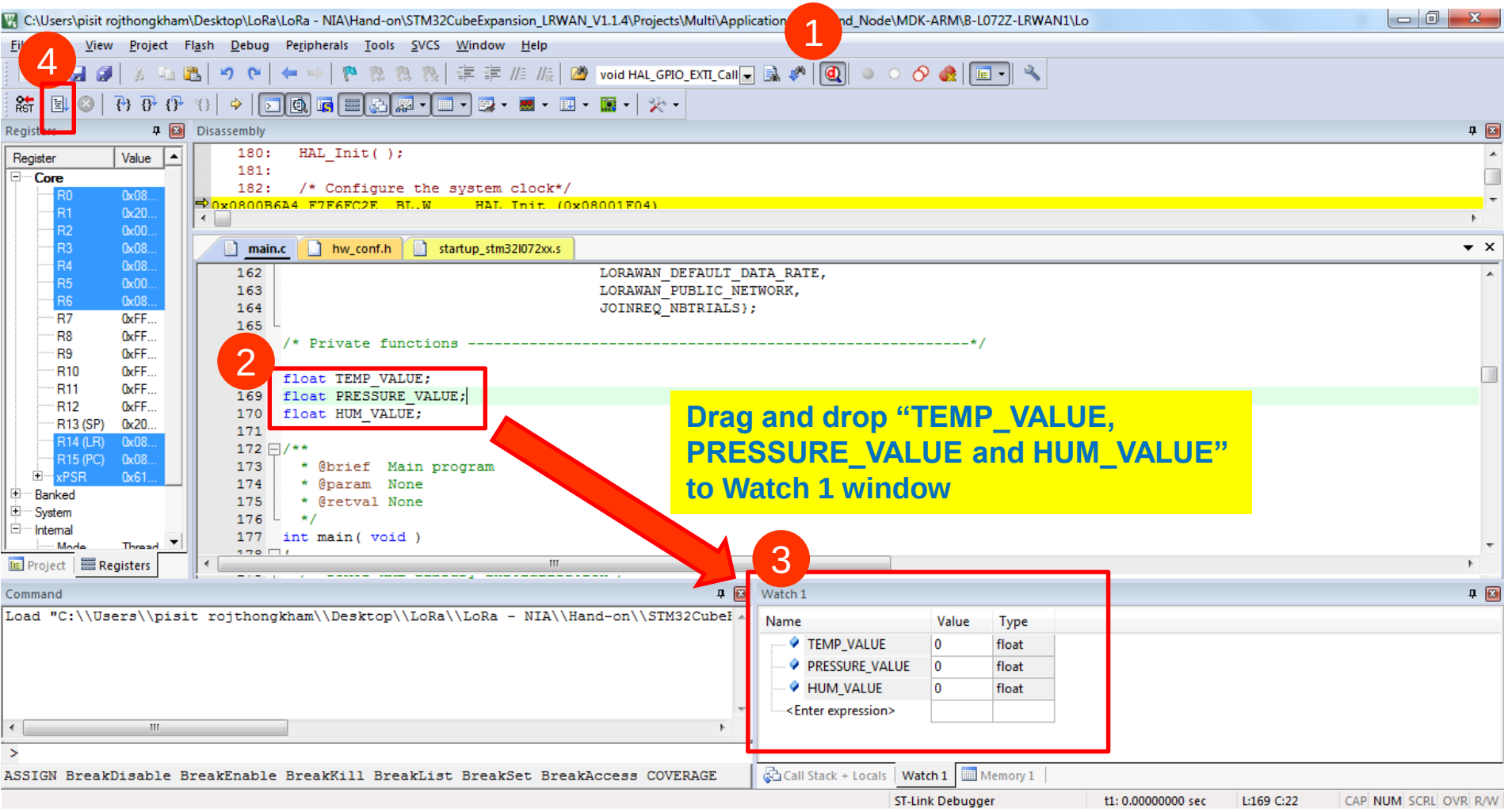
- Adding global variables in “main.c” , TEMP_VALUE, PRESSURE_VALUE, HUM_VALUE

```
166  /* Private functions --------
167
168  float TEMP_VALUE;
169  float PRESSURE_VALUE;
170  float HUM_VALUE;
171
172  /**
173    * @brief  Main program
174    * @param  None
175    * @retval None
176    */
177  int main( void )
```

- Get data from sensor in “main.c”

```
261
262    BSP_sensor_Read( &sensor_data );
263
264    TEMP_VALUE = sensor_data.temperature;
265    PRESSURE_VALUE = sensor_data.pressure;
266    HUM_VALUE = sensor_data.humidity;
```

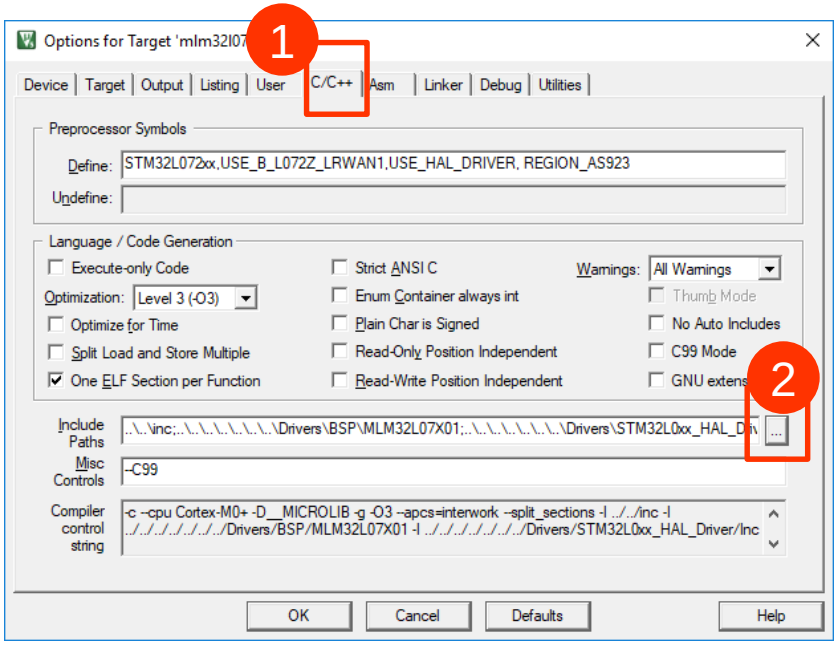
Debugging

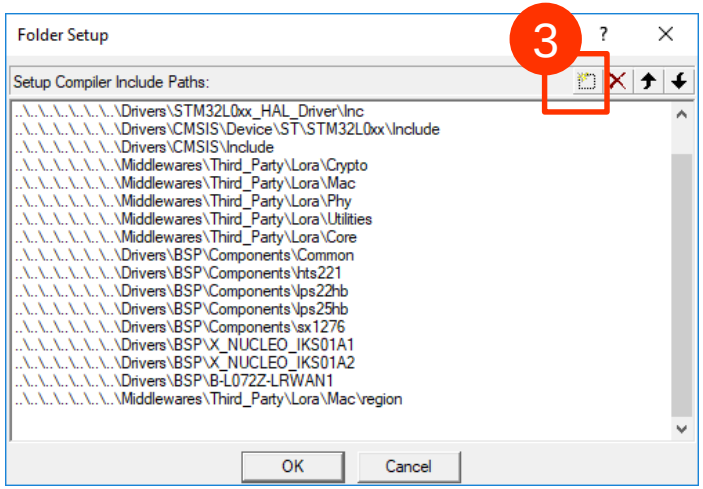

1
2
3
4
Drag and drop “TEMP_VALUE, PRESSURE_VALUE and HUM_VALUE” to Watch 1 window
float TEMP_VALUE;
float PRESSURE_VALUE;
float HUM_VALUE;
Watch 1
Name
Value
Type
TEMP_VALUE
PRESSURE_VALUE
HUM_VALUE
<Enter expression>

Add more Sensor “Magneto”

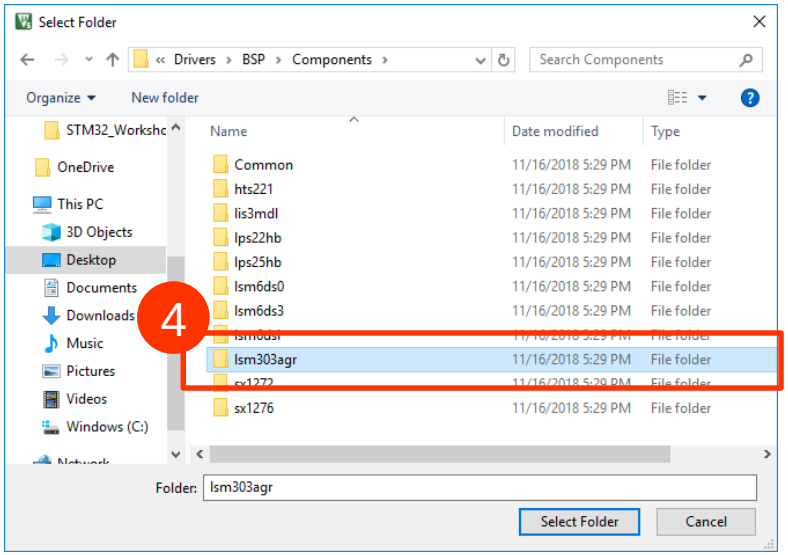
Configure Linker files

- Click Option for Target…

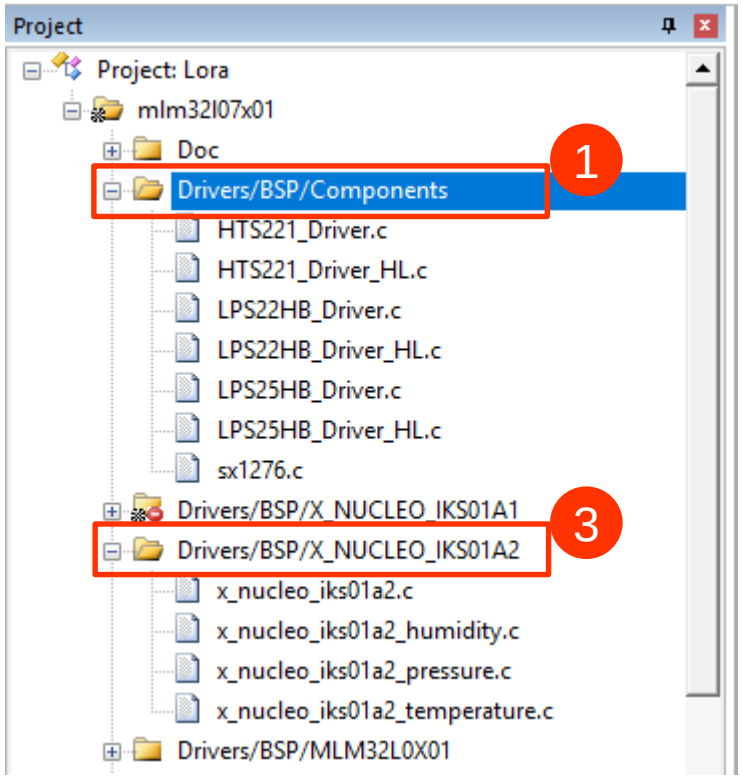




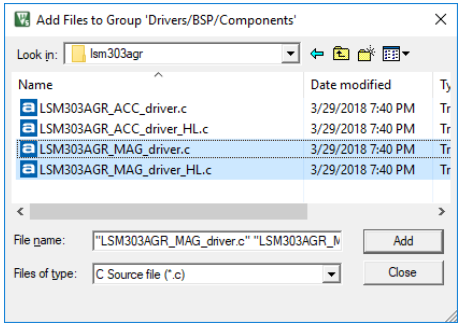
STM32CubeExpansion_LRWAN_V1.1.5\Drivers\BSP\Components



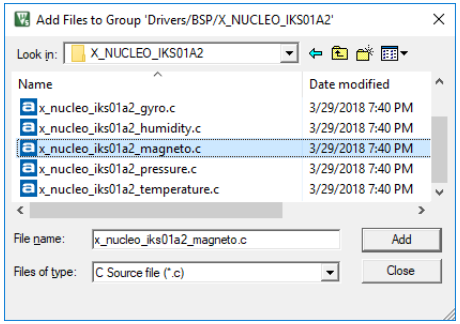
Adding Library



1. **Double Click Driver/BSP/Components**
2. **Add file from \STM32CubeExpansion_LRWAN_V1.1.5\Drivers\BSP\Components\lsm303agr**



3. **Double Click Driver/BSP/X_NUCLEO_IKS01A2**
4. **Add file from \STM32CubeExpansion_LRWAN_V1.1.5\Drivers\BSP\X_NUCLEO_IKS01A2**



5. **Click Rebuild** 

Add code in bsp.c

hw_conf.h | bsp.c*

```
57  #else  /* not LRWAN_NS1 */
58  #if defined(SENSOR_ENABLED)
59  #if defined (X_NUCLEO_IKS01A1)
60  #warning "Do not forget to select X_NUCLEO_IK
61  #include "x_nucleo_iks01a1_humidity.h"
62  #include "x_nucleo_iks01a1_pressure.h"
63  #include "x_nucleo_iks01a1_temperature.h"
64  #else  /* not X_NUCLEO_IKS01A1 */
65  #include "x_nucleo_iks01a2_humidity.h"
66  #include "x_nucleo_iks01a2_pressure.h"
67  #include "x_nucleo_iks01a2_temperature.h"
68  #include "x_nucleo_iks01a2_magneto.h"
69  #endif  /* X_NUCLEO_IKS01A1 */
```

1

hw_conf.h | bsp.c*

```
86  #if defined(SENSOR_ENABLED) || defined (LRWAN_NS1)
87  void *HUMIDITY_handle = NULL;
88  void *TEMPERATURE_handle = NULL;
89  void *PRESSURE_handle = NULL;
90  void *MAGNETO_handle = NULL;
91  #endif
92
93  SensorAxes_t MAG_Value;          /*!< Magnetometer Value */
94
95  void BSP_sensor_Read( sensor_t *sensor_data)
96  {
97    /* USER CODE BEGIN 5 */
98    float HUMIDITY_Value = 0;
99    float TEMPERATURE_Value = 0;
100   float PRESSURE_Value = 0;
101
102 #if defined(SENSOR_ENABLED) || defined (LRWAN_NS1)
103   BSP_HUMIDITY_Get_Hum(HUMIDITY_handle, &HUMIDITY_Value);
104   BSP_TEMPERATURE_Get_Temp(TEMPERATURE_handle, &TEMPERATURE_Value);
105   BSP_PRESSURE_Get_Press(PRESSURE_handle, &PRESSURE_Value);
106   BSP_MAGNETO_Get_Axes(MAGNETO_handle, &MAG_Value);
107 #endif
```

2

3

4

hw_conf.h | bsp.c*

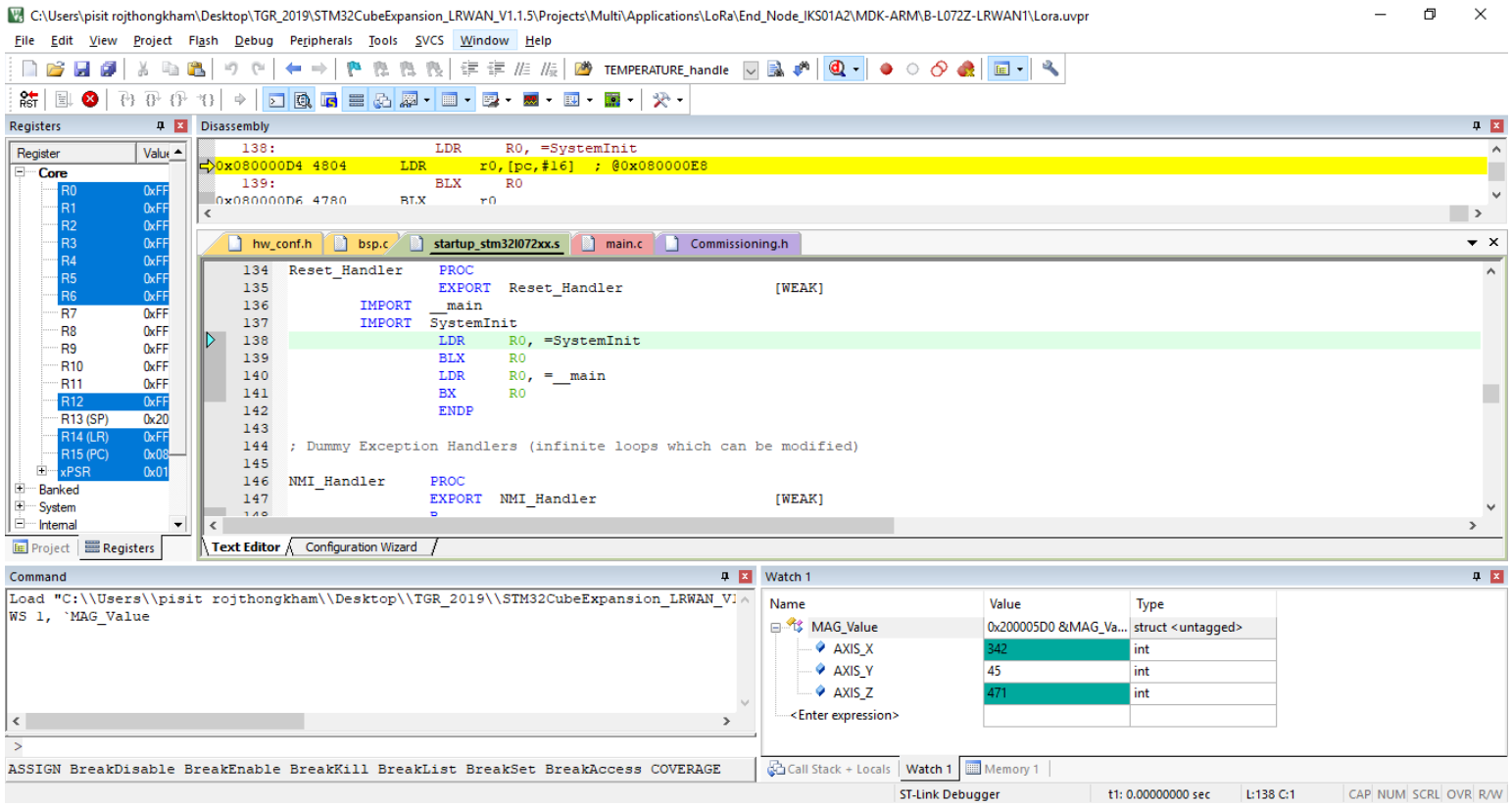
```
112   sensor_data->latitude  = (int32_t) ((STSOP_LATTITUDE  * MAX_GP
113   sensor_data->longitude = (int32_t) ((STSOP_LONGITUDE  * MAX_GP
114   /* USER CODE END 5 */
115 }
116
117 void  BSP_sensor_Init( void  )
118 {
119   /* USER CODE BEGIN 6 */
120
121 #if defined(SENSOR_ENABLED) || defined (LRWAN_NS1)
122   /* Initialize sensors */
123   BSP_HUMIDITY_Init( HTS221_H_0, &HUMIDITY_handle );
124   BSP_TEMPERATURE_Init( HTS221_T_0, &TEMPERATURE_handle );
125   BSP_PRESSURE_Init( PRESSURE_SENSORS_AUTO, &PRESSURE_handle
126   BSP_MAGNETO_Init(MAGNETO_SENSORS_AUTO, &MAGNETO_handle);
127
128   /* Enable sensors */
129   BSP_HUMIDITY_Sensor_Enable( HUMIDITY_handle );
130   BSP_TEMPERATURE_Sensor_Enable( TEMPERATURE_handle );
131   BSP_PRESSURE_Sensor_Enable( PRESSURE_handle );
132   BSP_MAGNETO_Sensor_Enable ( MAGNETO_handle );
133 #endif
134    /* USER CODE END 6 */
135 }
```

5

6

Running the code

- Rebuild and Debug



STMStudio run-time variables monitoring and visualization tool for STM32 microcontrollers

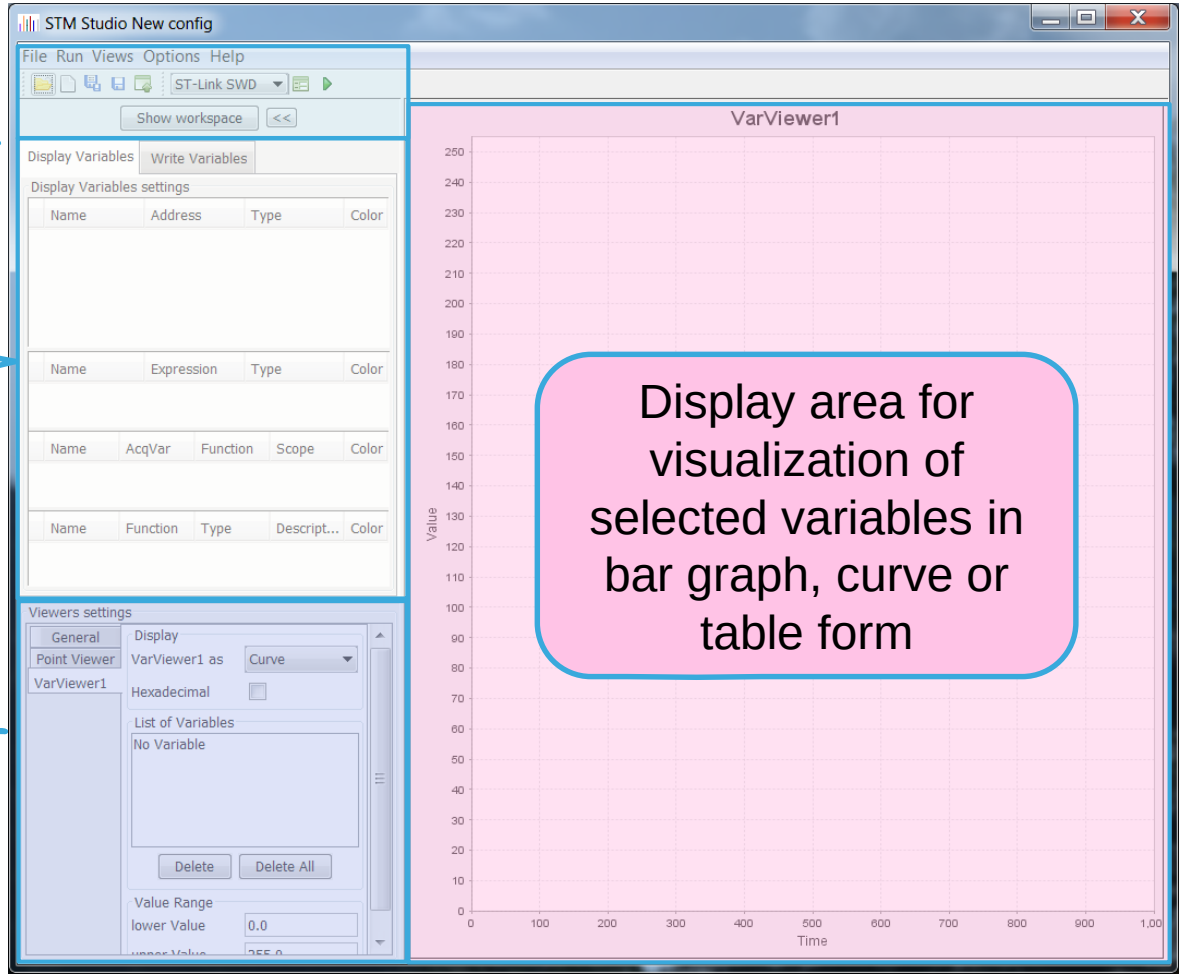
STMStudio overview

- Application allows non-intrusive sampling and real time visualization of user's variables while the application is running (it doesn't affect application timing).

User interface area: configuration, start/stop acquisition

Variables manipulation data, setting expressions and functions

Display area settings



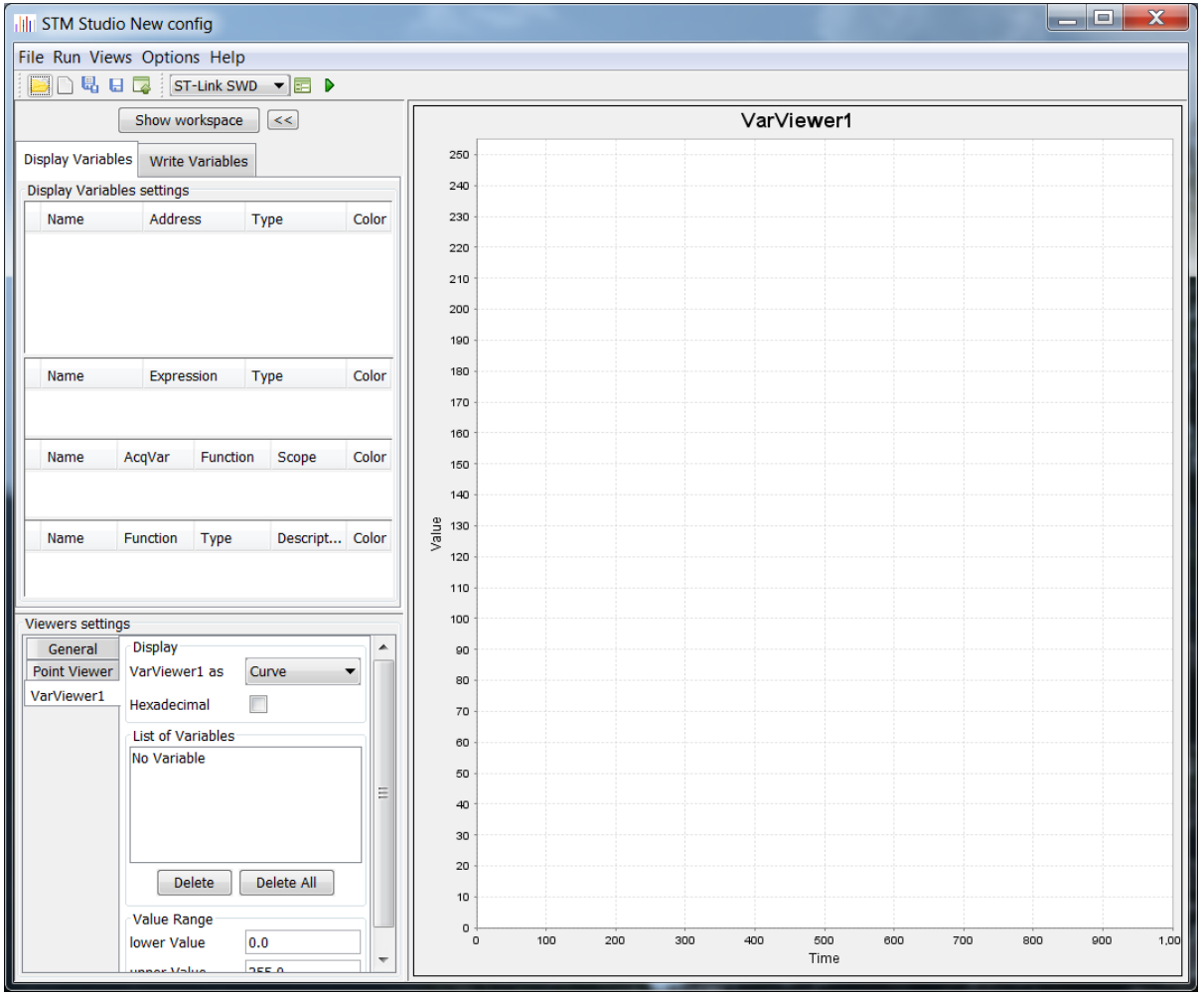
Display area for visualization of selected variables in bar graph, curve or table form

life.augmented

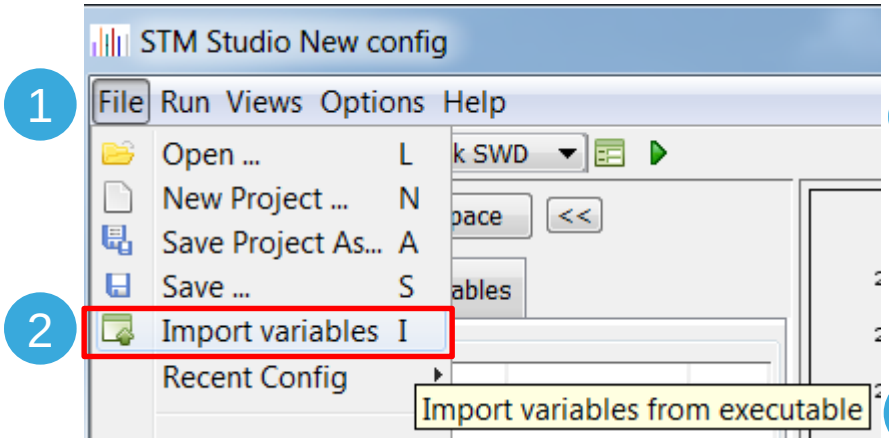
STMStudio - procedure

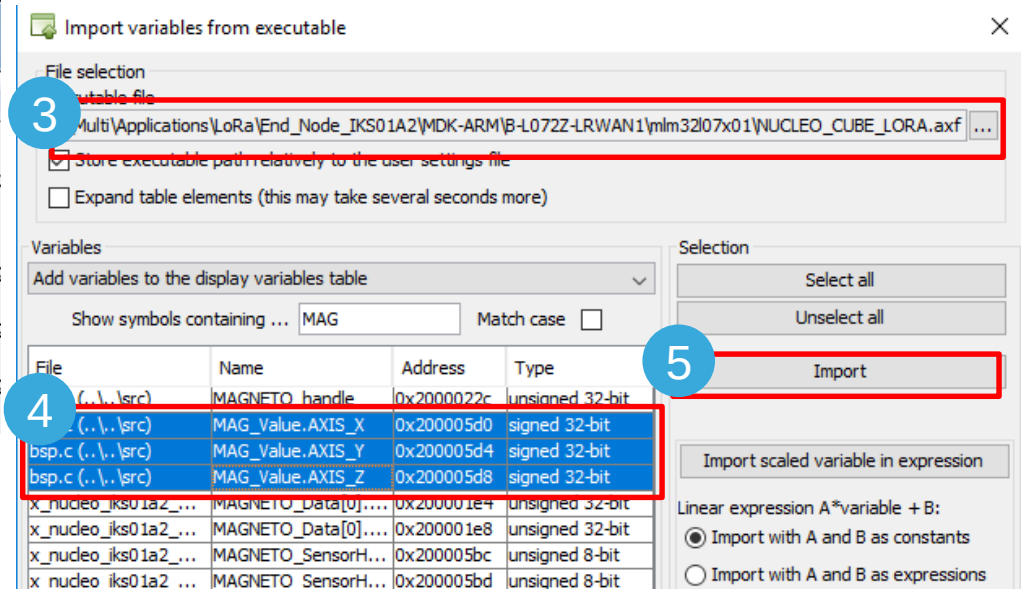
- In order to start monitoring, the following procedure should be performed:
  - Embedded application should be programmed into the MCU
  - MCU should be connected to the computer (where STMStudio is run) via STLink or Rlink programmers over SWD/JTAG for STM32 and SWIM for STM8 devices
  - The same executable file (.elf, .out, .axf) should be opened by STMStudio and desired global variable should be imported.
  - Acquisition should be started by *Run→Start* or clicking "**Play**" button
  - Acquisition can be stopped anytime by *Run→Stop* or clicking "**Stop**" button

For detailed information refer to dedicated user manual UM1025



Importing global variables





- Application can import global variables from executable files (.elf, .out, .axf) generated by any tool for STM32 and STM8 MCUs.
- To select proper file and variables, use *File→Import variables* shortcut or dedicated button
- To select particular components within the table, check “Expand table elements” option.

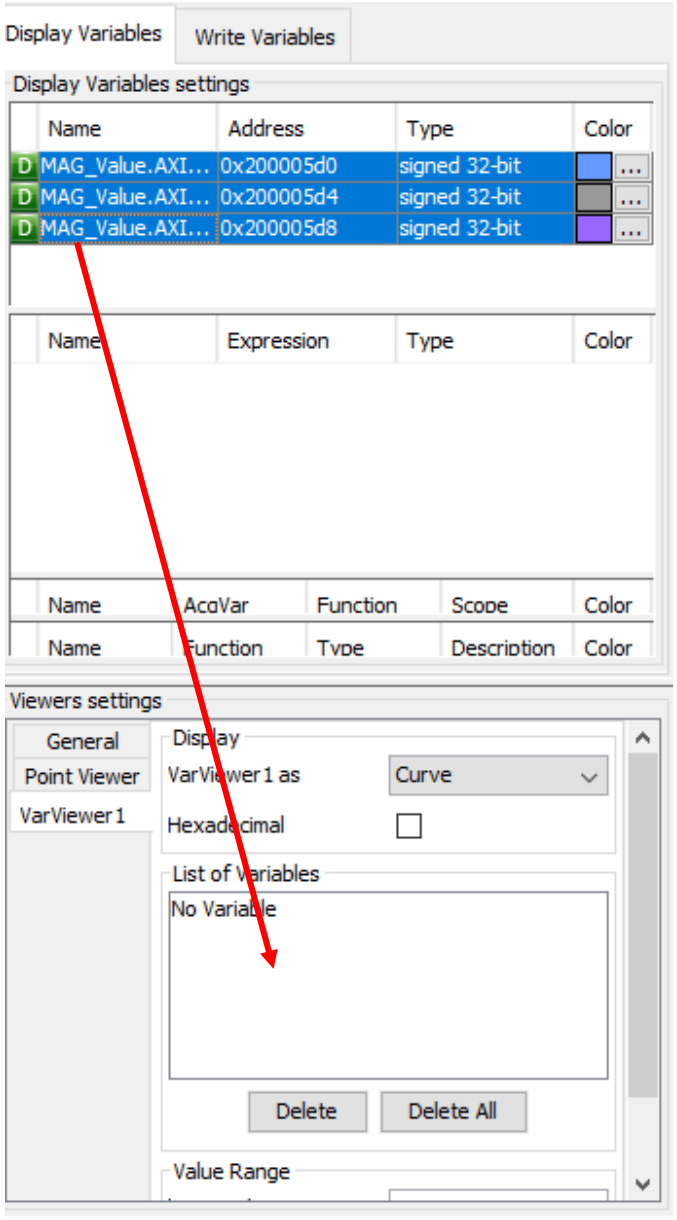
Task:
Import MAG_Value_AXIS_X, MAG_Value_AXIS_Y and MAG_Value_AXIS_Z variables

life.augmented

Adding variables to viewers

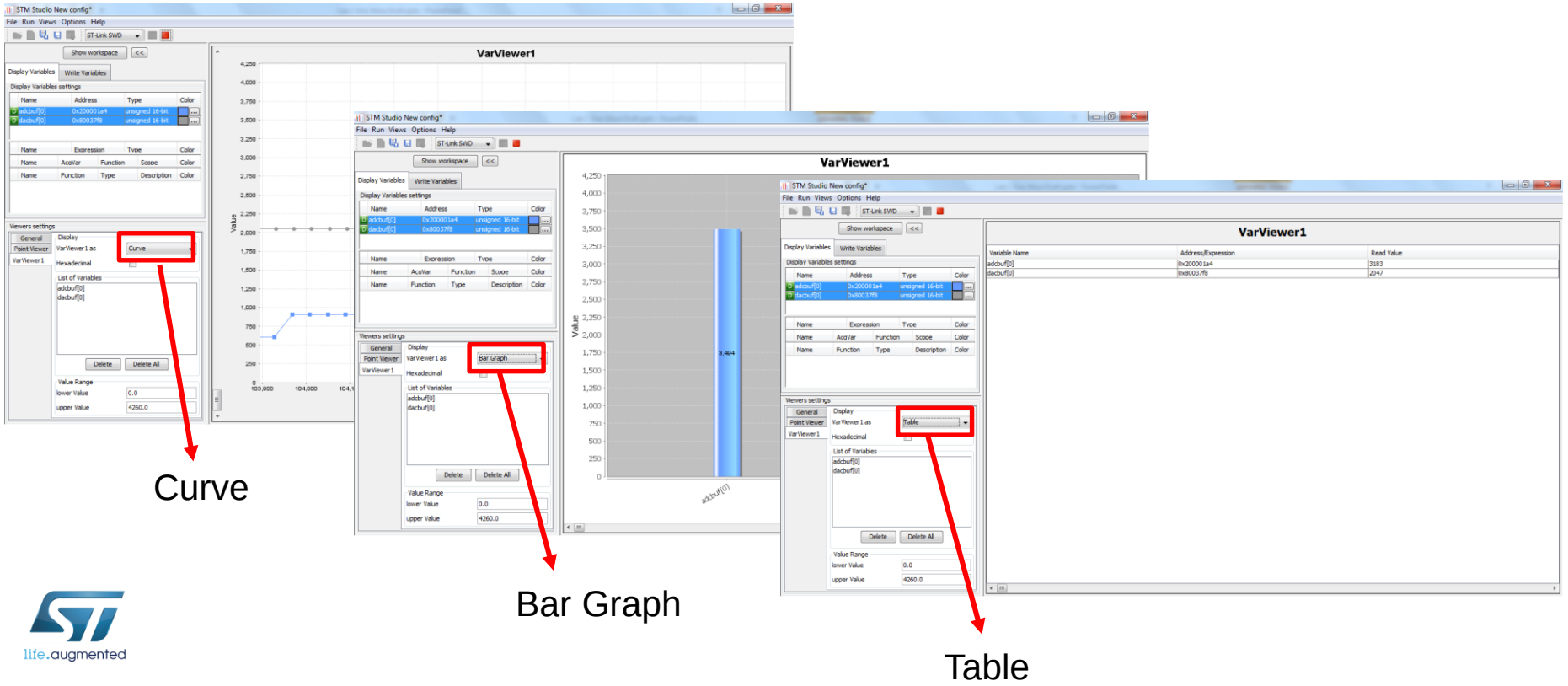
After selection of variables from executable file there are few options possible:

- Use of **drag-and-drop** to add them directly to the Viewers settings window
- Creation of **an expression** based on the variable(s) – described on next slides
- Use one of **predefined functions** (i.e. max, min, …) on the selected variable – described on next slides



STMStudio viewers

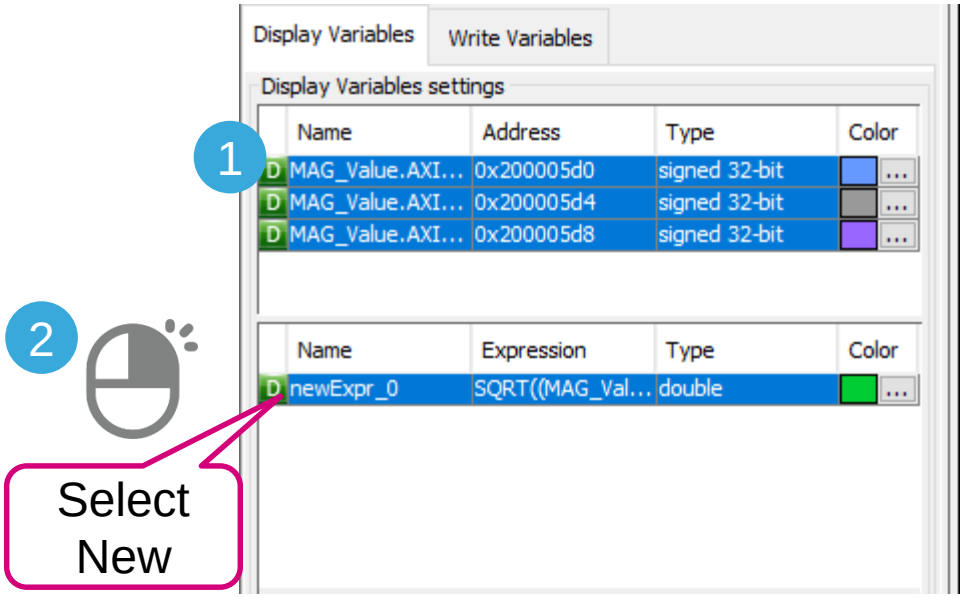
- There are 3 possible viewers available in STMStudio: **bar graph**, **curve** and **table**
- Switching among the viewers can be done anytime, even during acquisition without necessity to stop it
- All selected data are automatically stored in text log file that can be analyzed afterwards

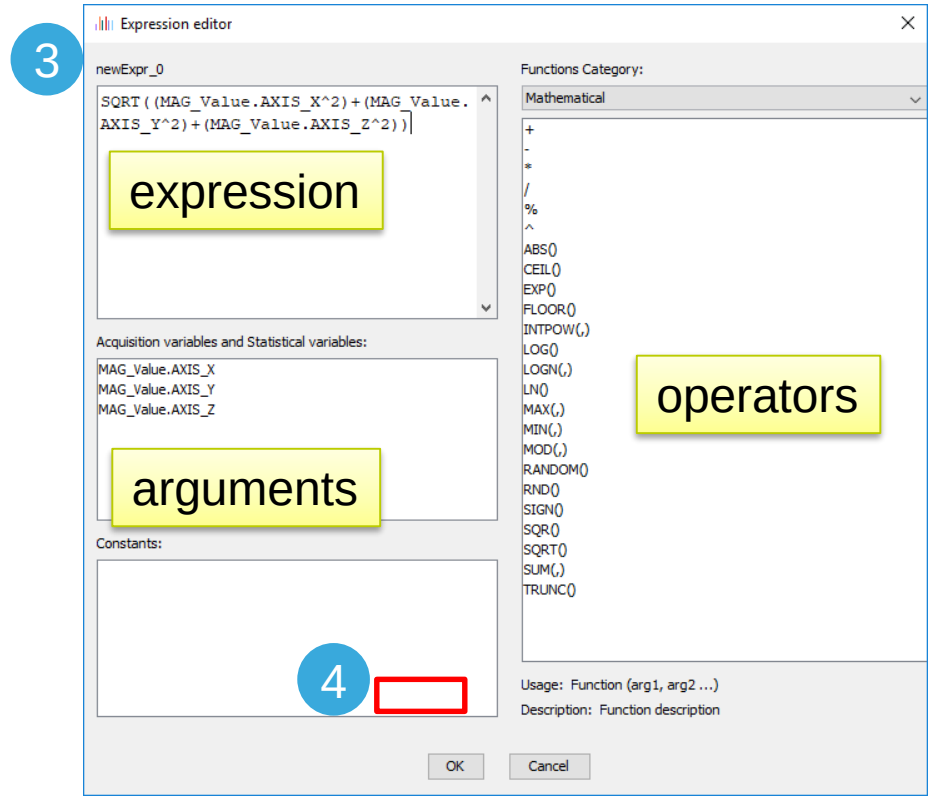


Adding expressions

- It is possible to define own expressions based on the monitored variables

Task:
Add a new expression
SQRT((MAG_Value.AXIS_X^2)+(MAG_Value.AXIS_Y^2)+(MAG_Value.AXIS_Z^2))



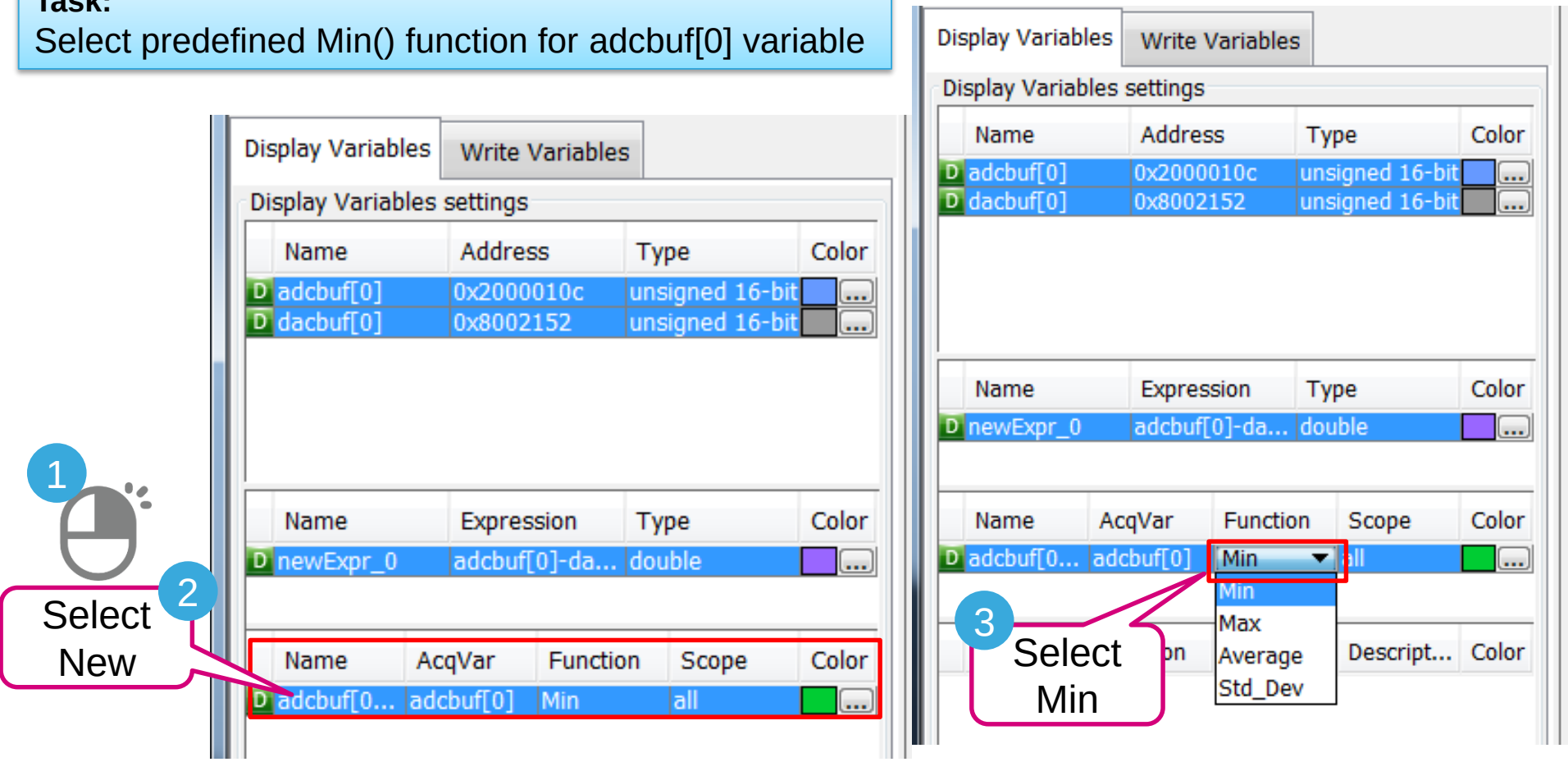


life.augmented

Adding predefined function

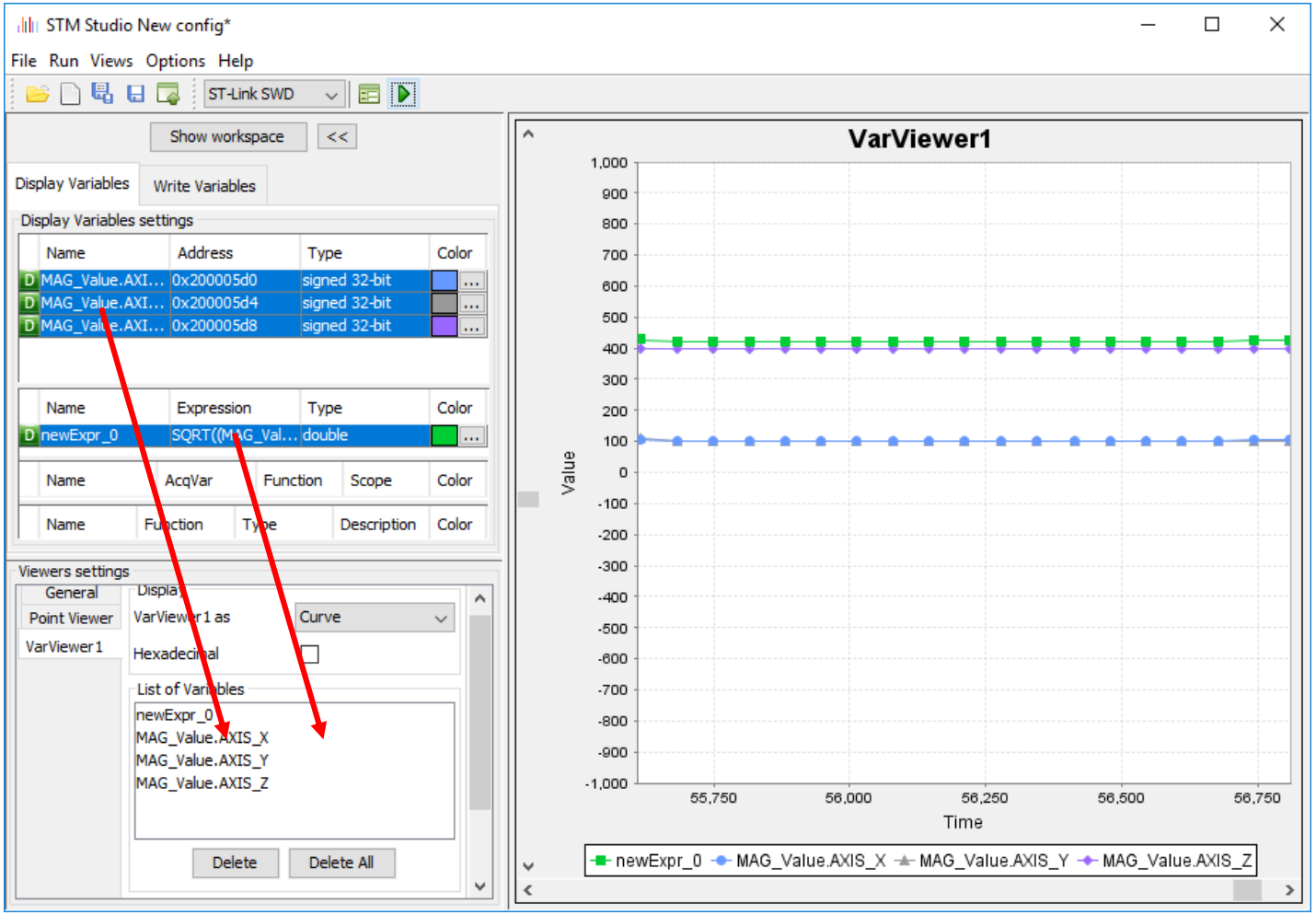
- It is possible to use some predefined functions on the monitored variables

Task:
Select predefined Min() function for adcbuf[0] variable



Add expressions/function results to viewers

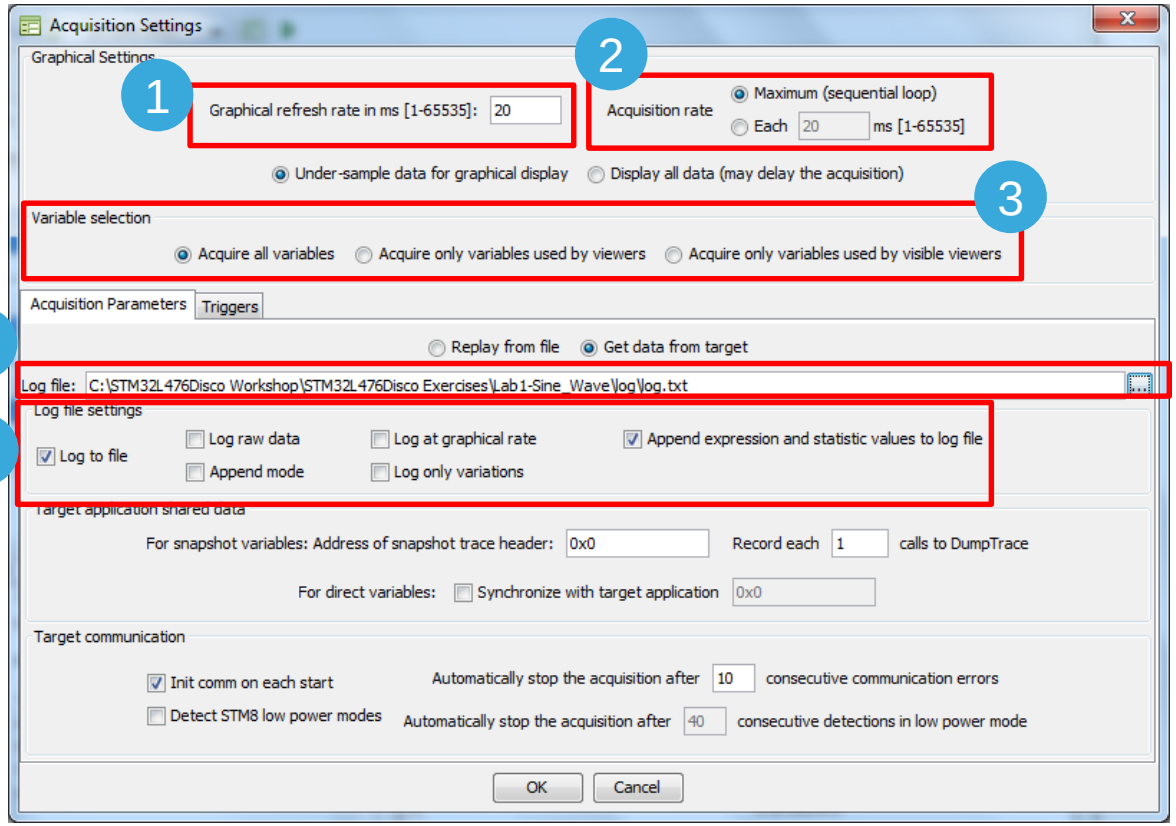
- It is possible to add all defined expressions and functions to the viewers for monitoring (like single variables) by drag & drop mechanism



life.augmented

STMStudio configuration

- Configuration of STMStudio is accessible by selecting ***Options>Acquisition Settings*** (or using button)
- Main parameters which can be configured there are:
    1. Graphical refresh rate
    2. Acquisition rate
    3. Variable selections
    4. Location of the text log file
    5. Log file settings



Using log file for data analysis

- All selected data are stored automatically “runtime” in log text file which can be analyzed afterwards

log.txt - Notepad

File Edit Format View Help

```
Syntax version=4
Fri Nov 16 20:44:01 2018
Starting process (2343749 clocks per sec)

LogRawData=false
VarHeader=1
D:              MAG_Value.AXIS_X        MAG_Value.AXIS_Y        MAG_Value.AXIS_Z        !newExpr_0!
D:      time(ms)        (0x200005d0,5)  (0x200005d4,5)  (0x200005d8,5)  !SQRT((MAG_Value.AXIS_X^2)+(MAG_Value.AXIS
D:      0.18    102     97      396     420.27253062744893
D:      0.94    102     97      396     420.27253062744893
D:      1.68    102     97      396     420.27253062744893
D:      2.29    102     97      396     420.27253062744893
D:      2.95    102     97      396     420.27253062744893
D:      3.57    102     97      396     420.27253062744893
D:      4.15    102     97      396     420.27253062744893
D:      4.69    102     97      396     420.27253062744893
D:      5.16    102     97      396     420.27253062744893
D:      5.60    102     97      396     420.27253062744893
D:      6.05    102     97      396     420.27253062744893
D:      6.50    102     97      396     420.27253062744893
D:      6.93    102     97      396     420.27253062744893
D:      7.35    102     97      396     420.27253062744893
D:      7.77    102     97      396     420.27253062744893
D:      8.26    102     97      396     420.27253062744893
```

Setup End node

- Edit the ***comissioning.h*** file in End_Node/inc to set the end device comissioning parameters.

Commissioning.h*

```
74   * When set to 1 the application uses the Over-the-Air activation procedure
75   * When set to 0 the application uses the Personalization activation procedure
76   */
77  #define OVER_THE_AIR_ACTIVATION                     0
78
79  /*!
80   * Indicates if the end-device is to be connected to a private or public network
81   */
82  #define LORAWAN_PUBLIC_NETWORK                      true
83
84  /*!
85   * When set to 1 DevEui is LORAWAN_DEVICE_EUI
86   * When set to 0 DevEui is automatically generated by calling
87   *         BoardGetUniqueId function
88   */
89  #define STATIC_DEVICE_EUI                           1
90
91  /*!
92   * Mote device IEEE EUI (big endian)
93   *
94   * \remark see STATIC_DEVICE_EUI comments
95   */
96  #define LORAWAN_DEVICE_EUI                          { [redacted] }
```

OTAA disable

Static device EUI enable

Device

Name	DevEUI	Profile ID	Device Address	Connectivity Instances	Payload Format	Routing Profile
train54	[redacted]	LORA/GenericA.1_AS923_Rx2-SF10	[redacted]	Training	Cayenne LPP	Cayenne

Setup End node

- ***comissioning.h (con.)***

Device

Name	DevEUI	Profile ID	Device Address	Connectivity Instances	Payload Format	Routing Profile
train54	[redacted]	LORA/GenericA.1_AS923_Rx2-SF10	[redacted]	Training	Cayenne LPP	Cayenne

Commissioning.h*

```
118   * When set to 0 DevAdd is automatically generated using
119   *         a pseudo random generator seeded with a value derived from
120   *         BoardUniqueId value
121   */
122  #define STATIC_DEVICE_ADDRESS                       1
123  /*!
124   * Device address on the network (big endian)
125   *
126   * \remark see STATIC_DEVICE_ADDRESS comments
127   */
128  #define LORAWAN_DEVICE_ADDRESS                      ( uint32_t ) [redacted]
129
130  /*!
131   * AES encryption/decryption cipher network session key
132   */
133  #define LORAWAN_NWKSKEY                             { 0x28, 0xAE, 0xD2, 0x2B, 0x7E, 0x15, 0x16, 0xA6, 0x09, 0xCF, 0xAB, 0xF7, 0x15, 0x88, 0x4F, 0x3C }
134
135  /*!
136   * AES encryption/decryption cipher application session key
137   */
138  #define LORAWAN_APPSKEY                             { 0x16, 0x28, 0xAE, 0x2B, 0x7E, 0x15, 0xD2, 0xA6, 0xAB, 0xF7, 0xCF, 0x4F, 0x3C, 0x15, 0x88, 0x09 }
139
```

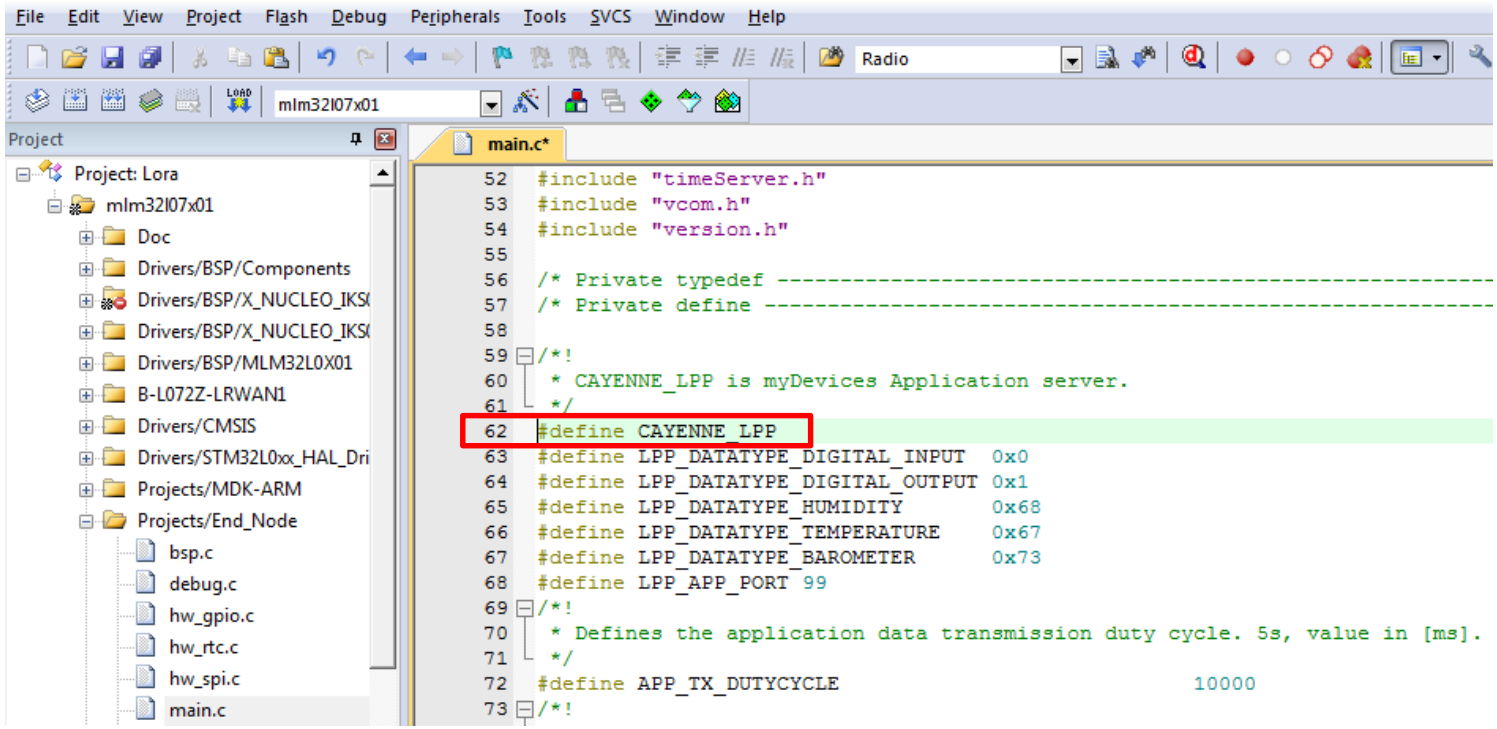
Static device Address enable

#define LORAWAN_NWKSKEY { 0x28, 0xAE, 0xD2, 0x2B, 0x7E, 0x15, 0x16, 0xA6, 0x09, 0xCF, 0xAB, 0xF7, 0x15, 0x88, 0x4F, 0x3C }

#define LORAWAN_APPSKEY { 0x16, 0x28, 0xAE, 0x2B, 0x7E, 0x15, 0xD2, 0xA6, 0xAB, 0xF7, 0xCF, 0x4F, 0x3C, 0x15, 0x88, 0x09 }

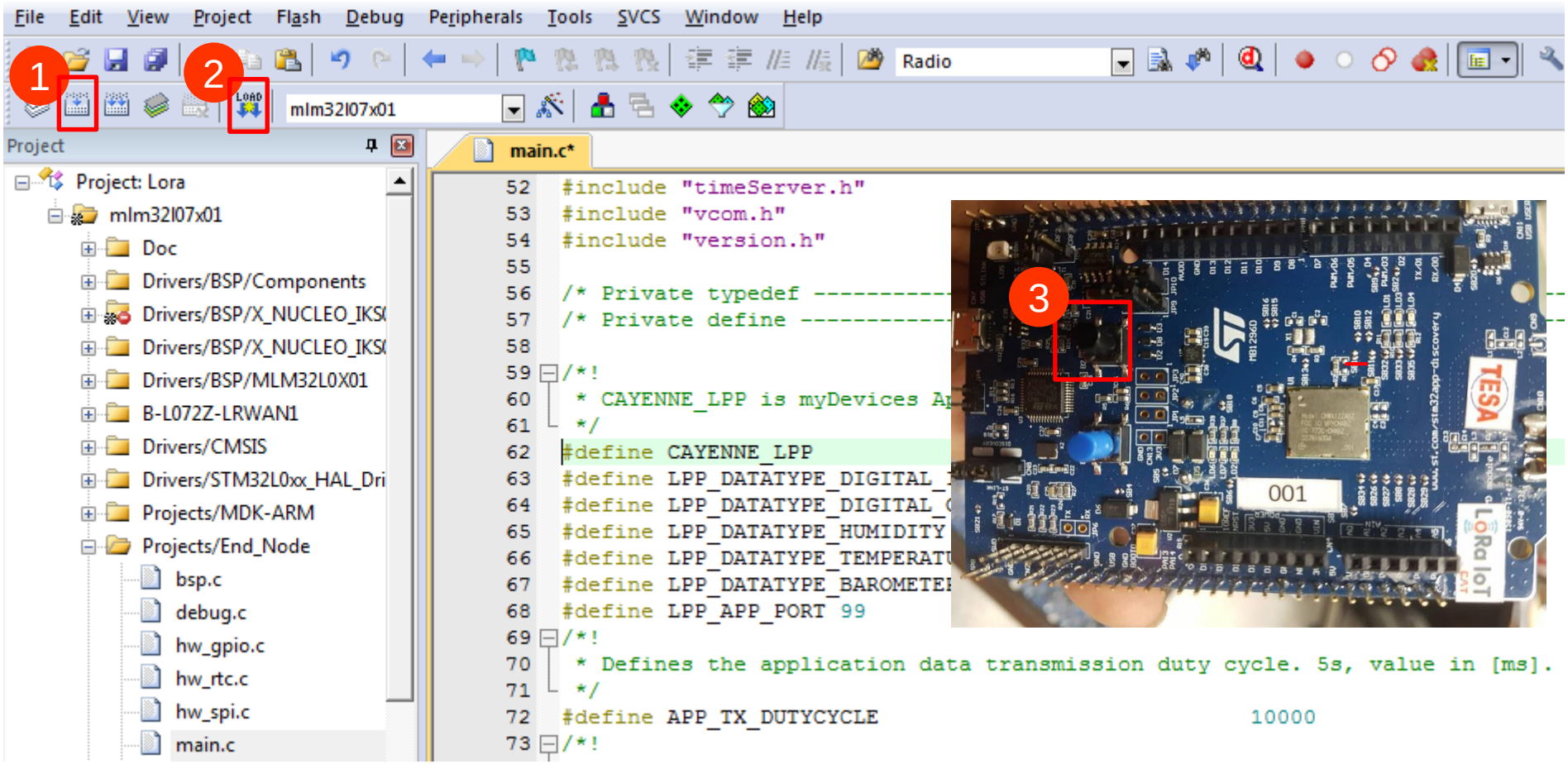
Cayenne LLP payload format

- By default, the Cayenne LLP payload format is disable (//). This is to be compatible with the myDevices Cayenne Dashboard application which will be used later for the visualization of the data. (https://mydevices.com/) , back to “main.c” uncomment line CAYENNE_LPP



Compile and Download

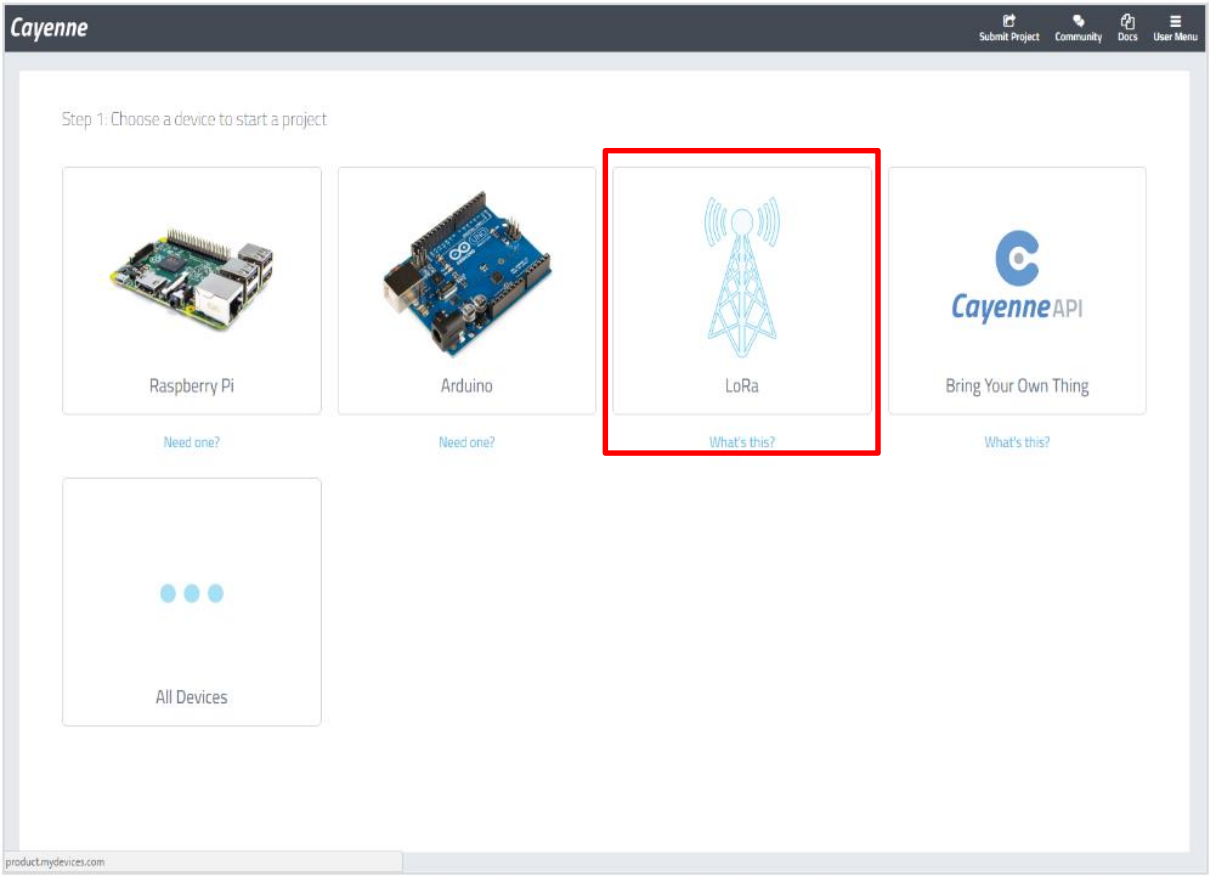
1 Click -> Build 2 Click -> LOAD 3 Press Reset button on board



life.augmented

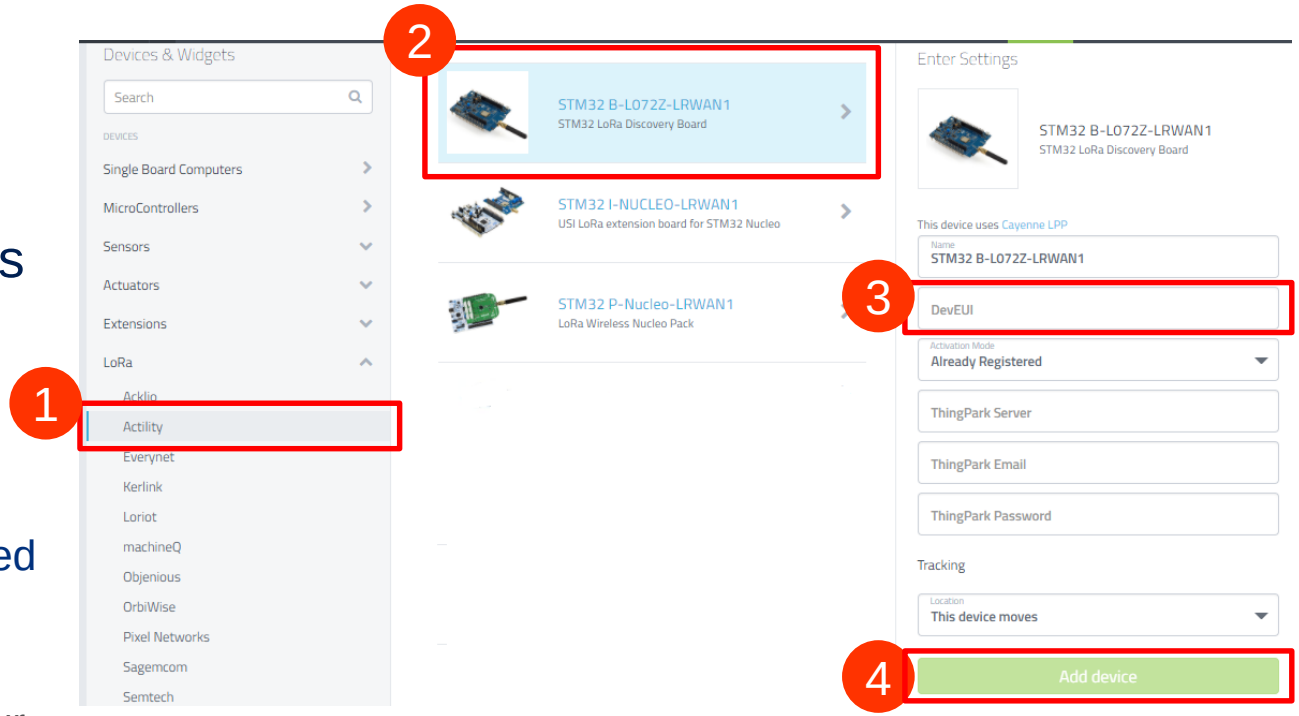
myDevices Cayenne Application Setup

- Create a free myDevices Cayenne account (https://mydevices.com/)
- Login and add a LoRa device.



myDevices Cayenne Application Setup

- Choose Actility Network from the list
- Select the ST device
- Enter the device settings and then add device.
  - Name
  - DevEUI
  - ThingPark Server (server you used when you created the CAT's LoRa account)
  - Tracking
    - This device moves (if your device has a GPS)
    - This device doesn't move (input fixed address)



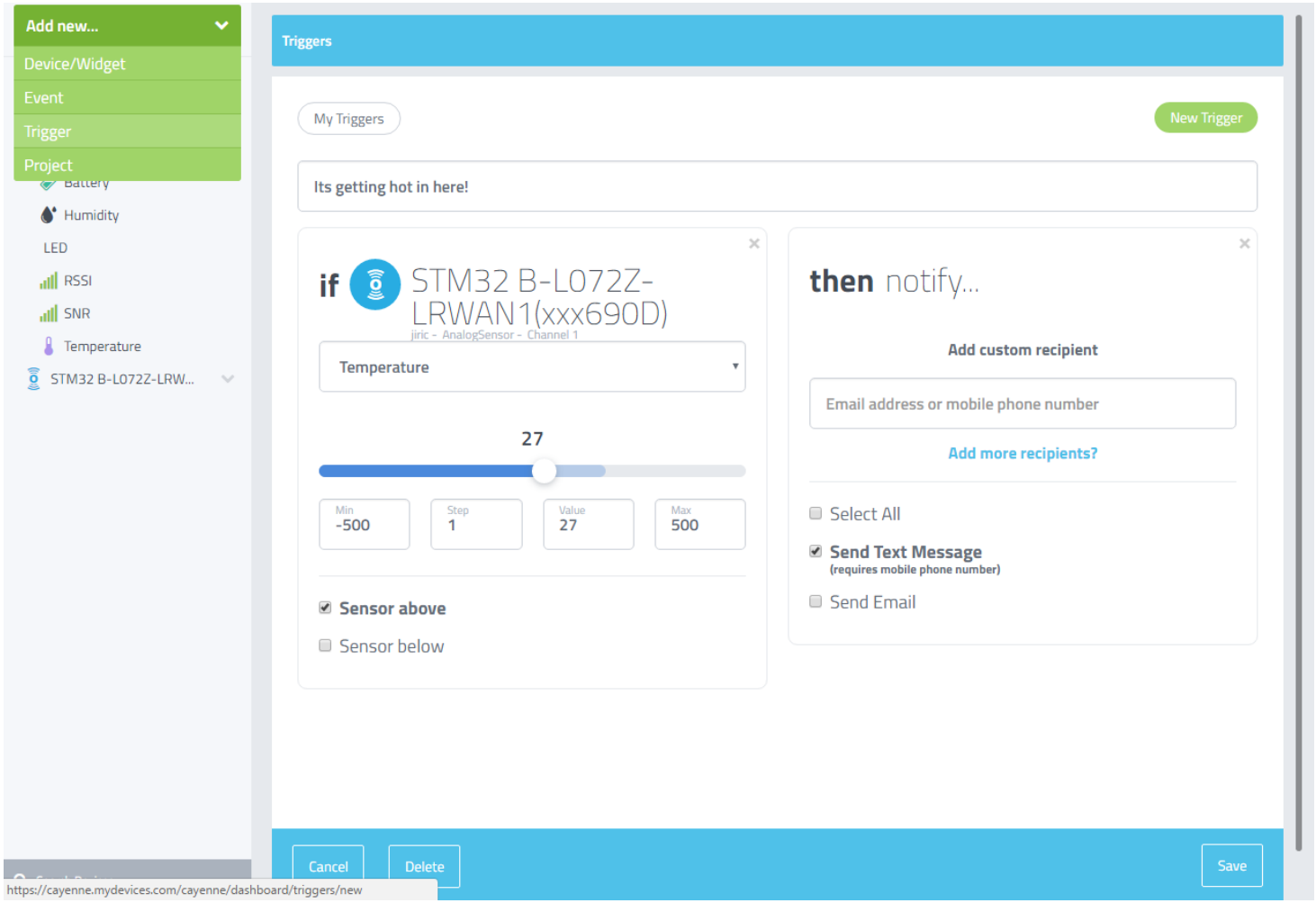
myDevices Cayenne Application Setup

- The widgets will automatically appear as soon as data are received from the device.
- Customize the dashboard to your liking.

life.augmented

Setup Alerts in Cayenne

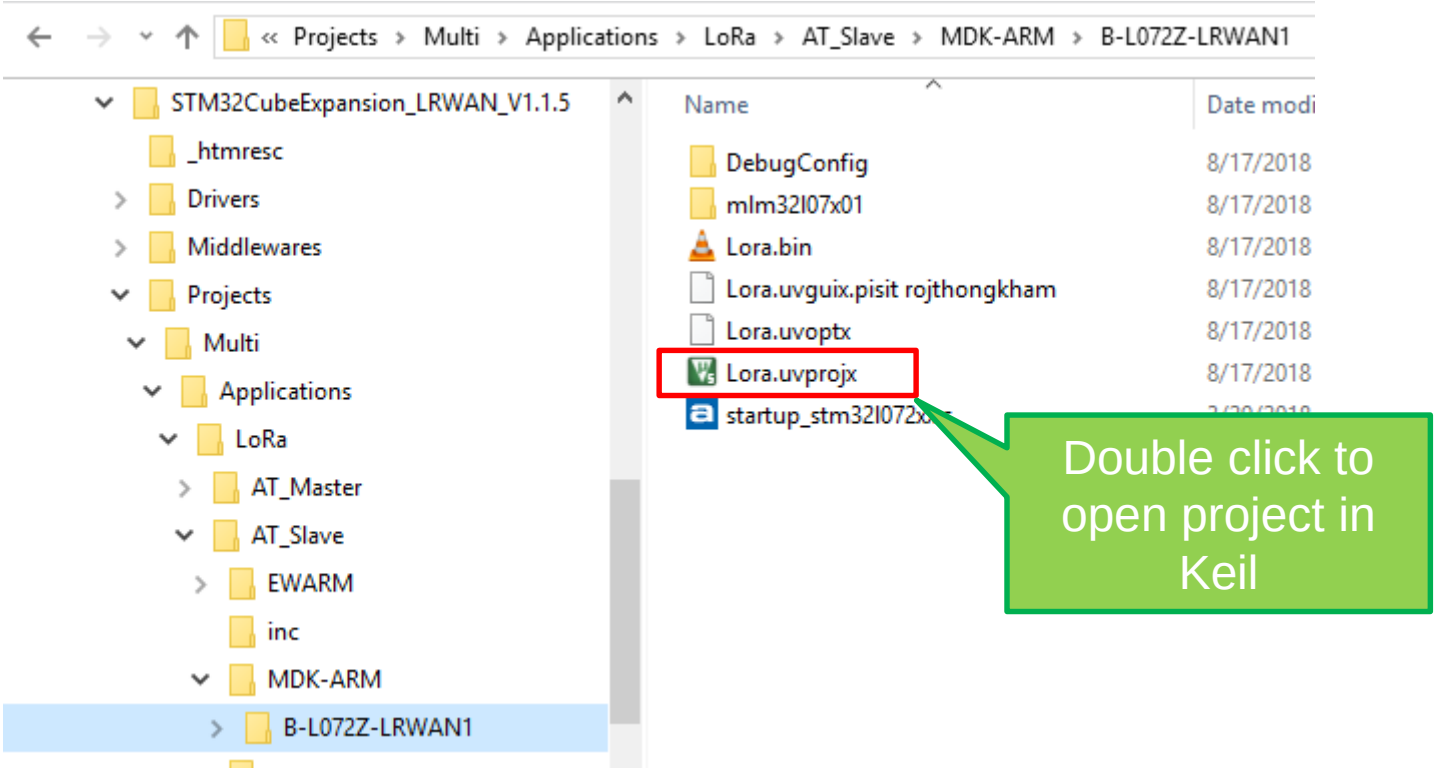
- You can set alerts based on the data coming from your device.



AT-Command OTAA mode

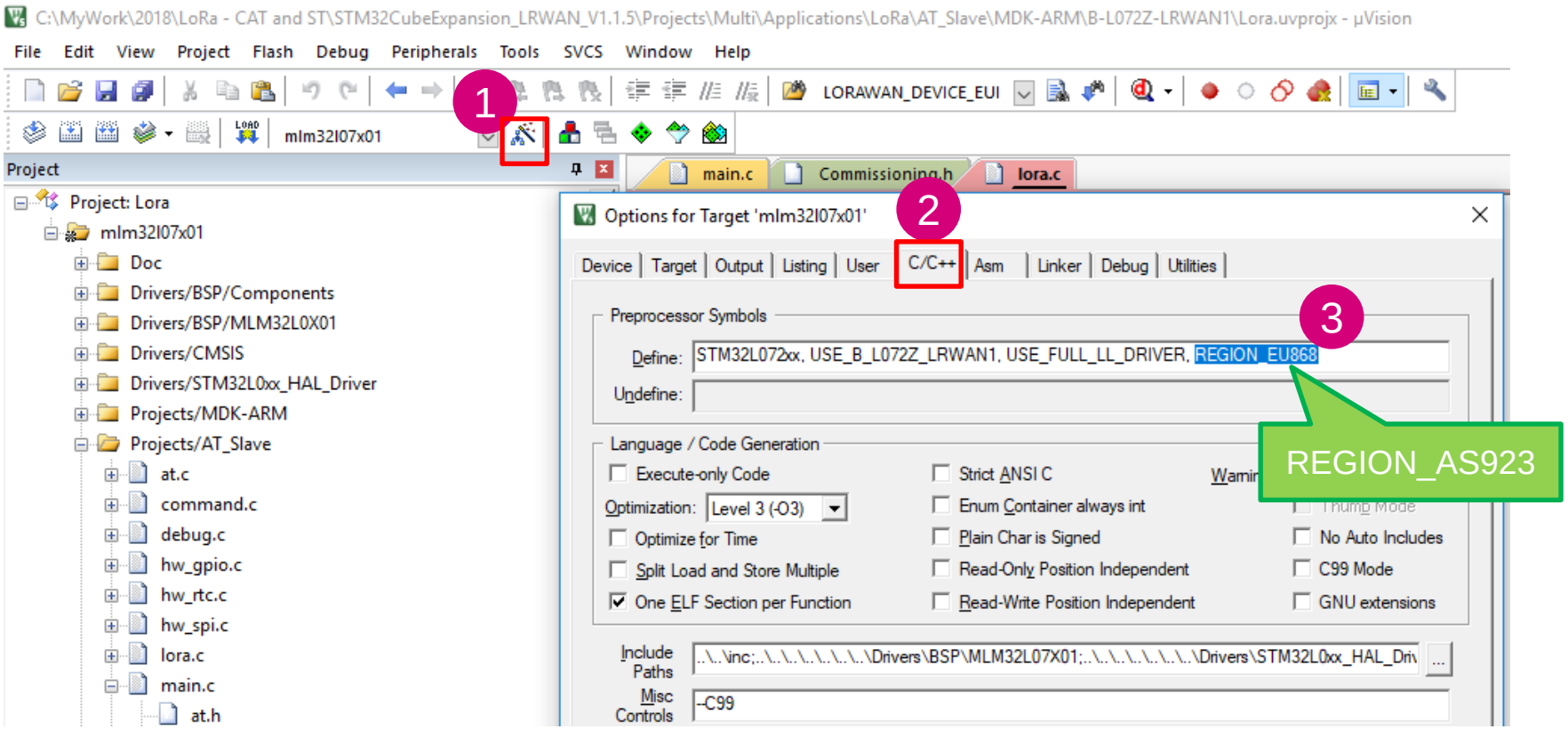
- ST LoRa package provided the example application to program muRata module to be a modem and support AT-Command.

- 1st . Open project
  - STM32CubeExpansion_LRWAN_V1.1.5\Projects\Multi\Applications\LoRa\AT_Slave\MDK-ARM\B-L072Z-LRWAN1



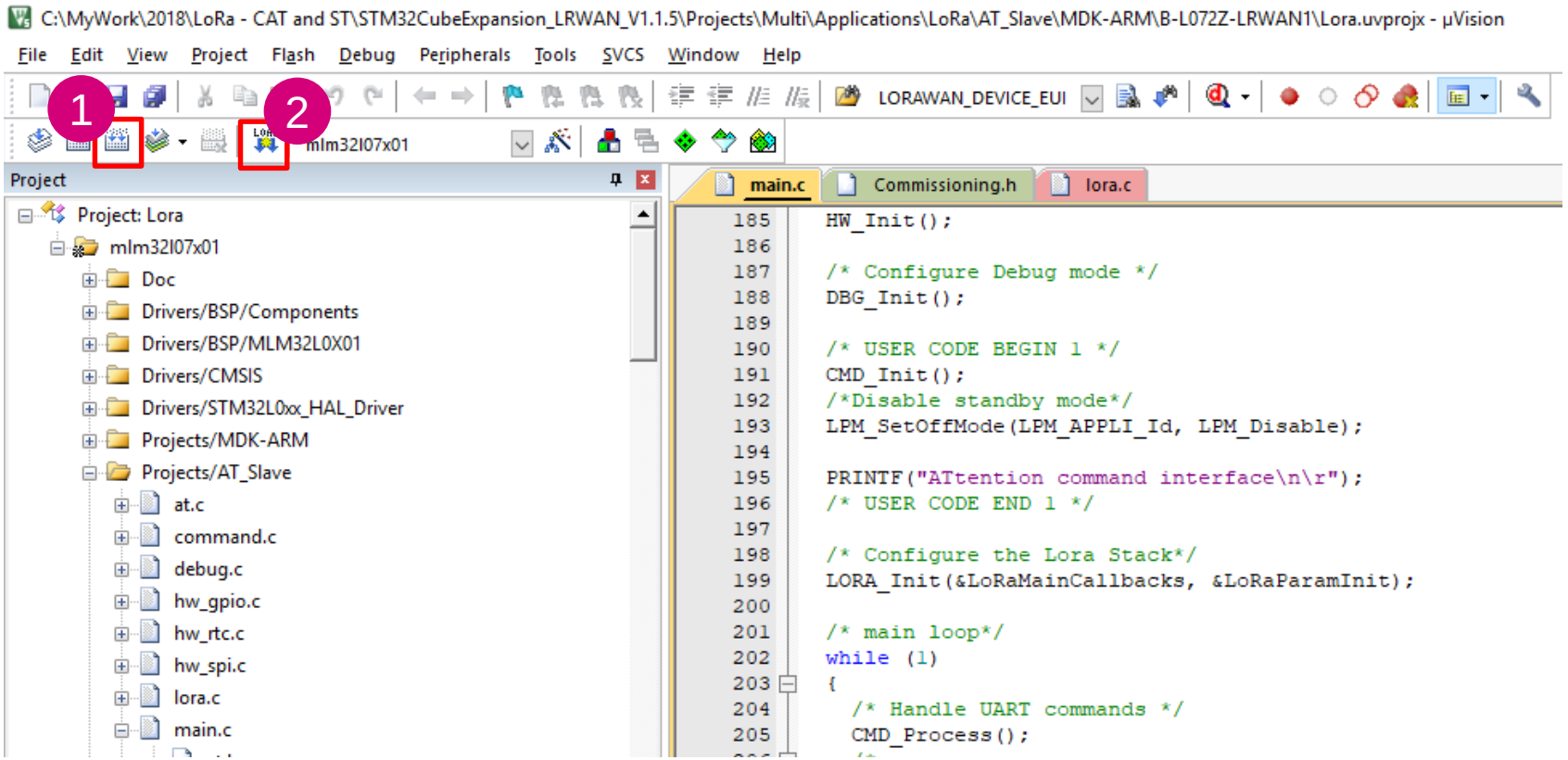
AT-Command OTAA mode

- 2nd configure frequency region from EU868 to AS923



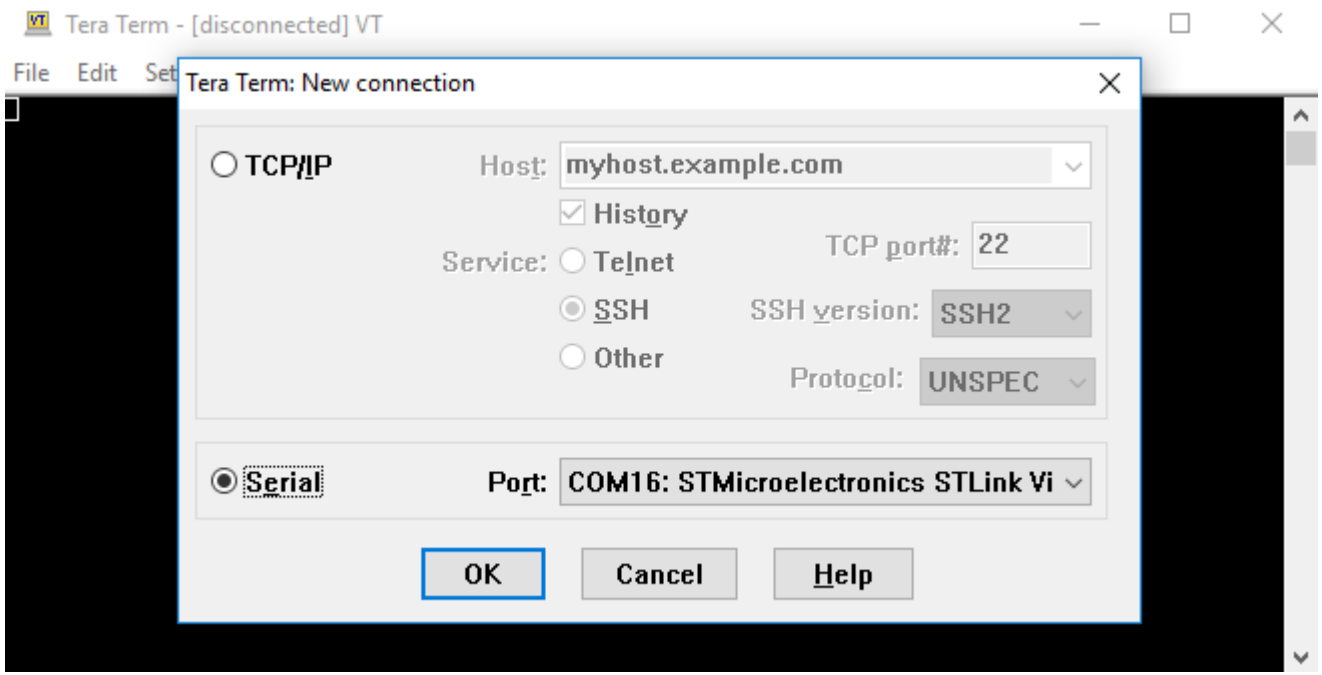
AT-Command OTAA mode

- 3rd re-compile and download



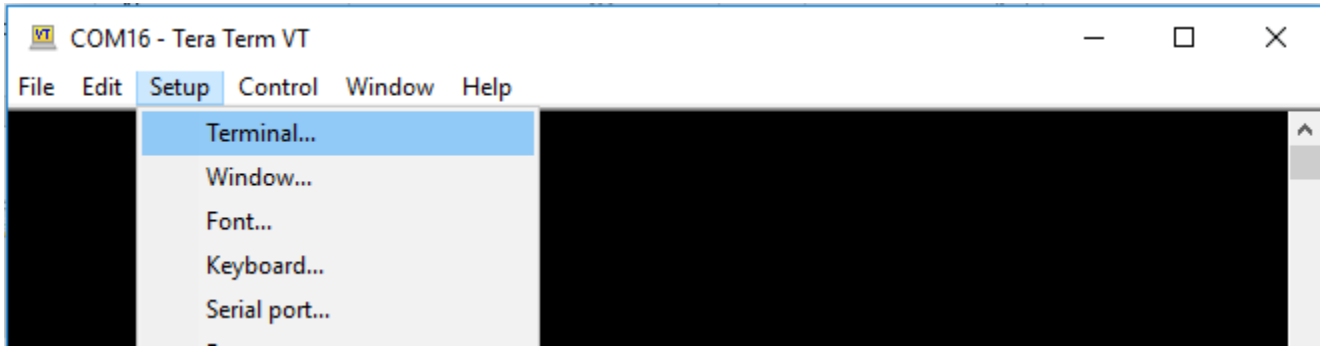
AT-Command OTAA mode

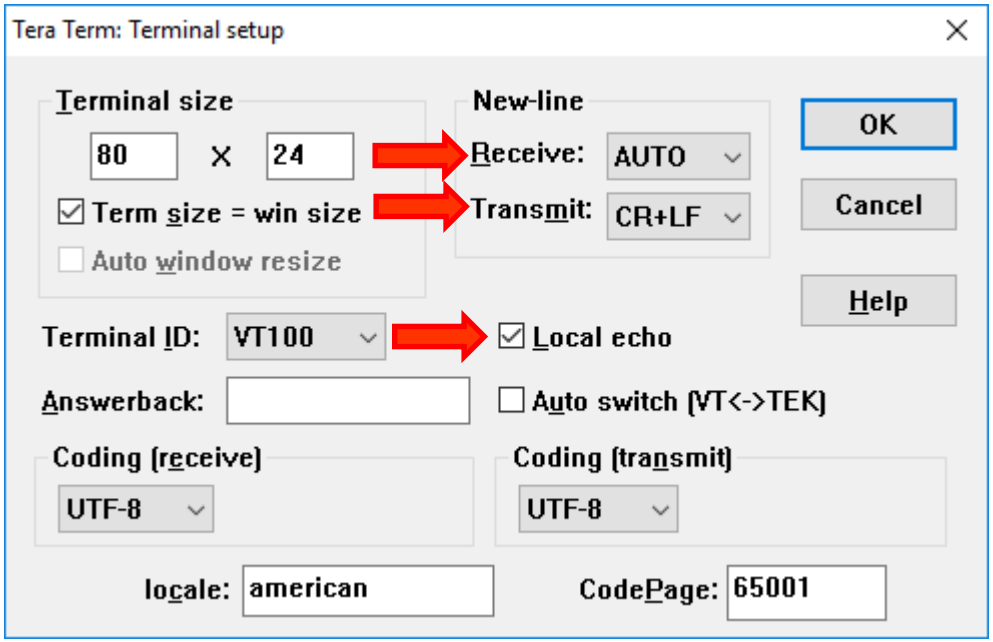
- Open Tere Term on desktop 



AT-Command OTAA mode

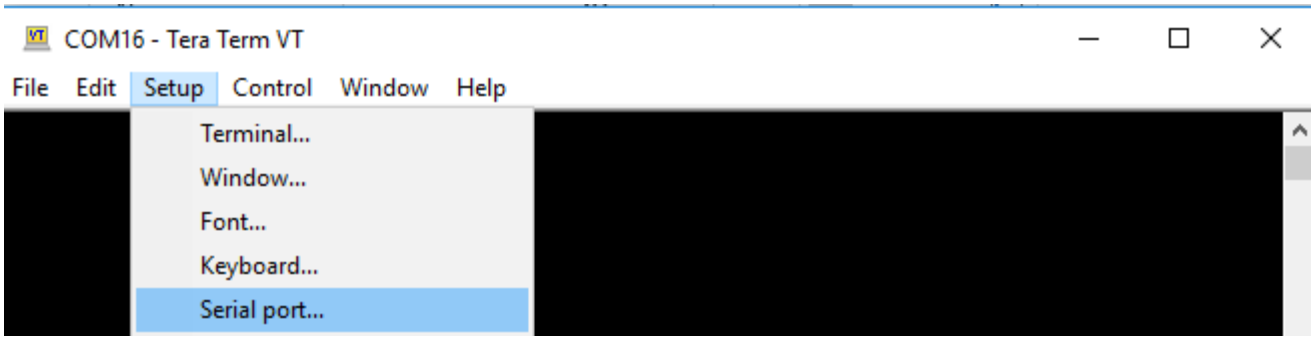
- Setup -> Terminal

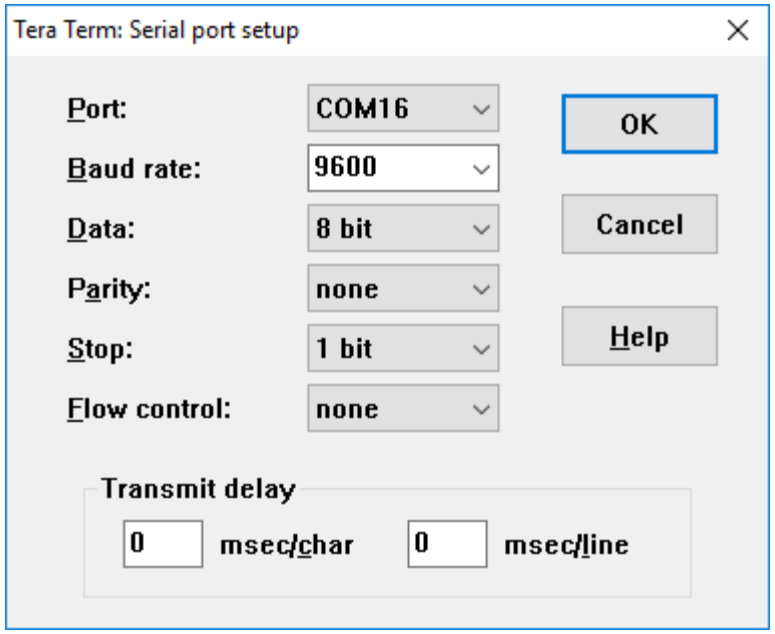




AT-Command OTAA mode

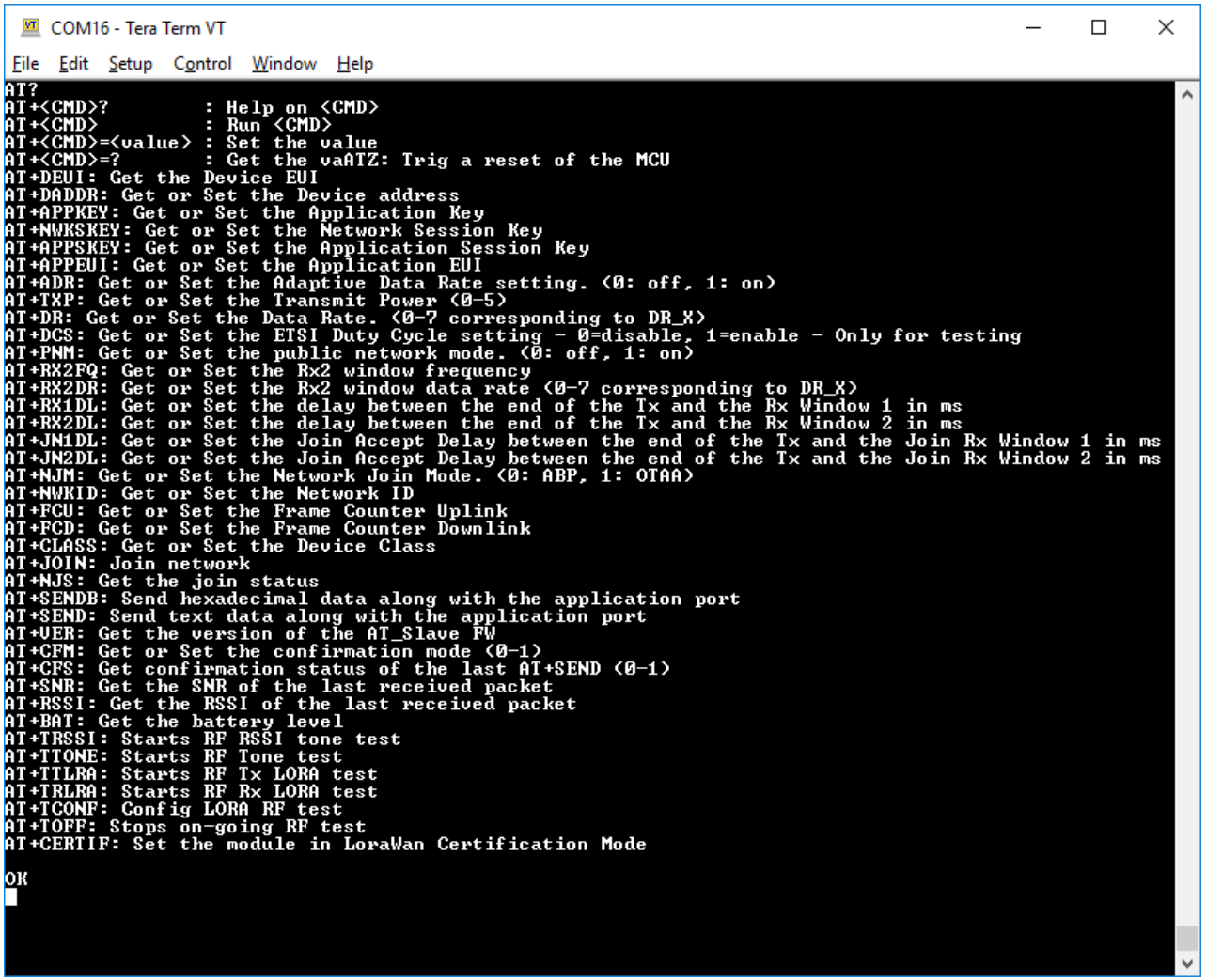
- Setup -> Serial port





AT-Command

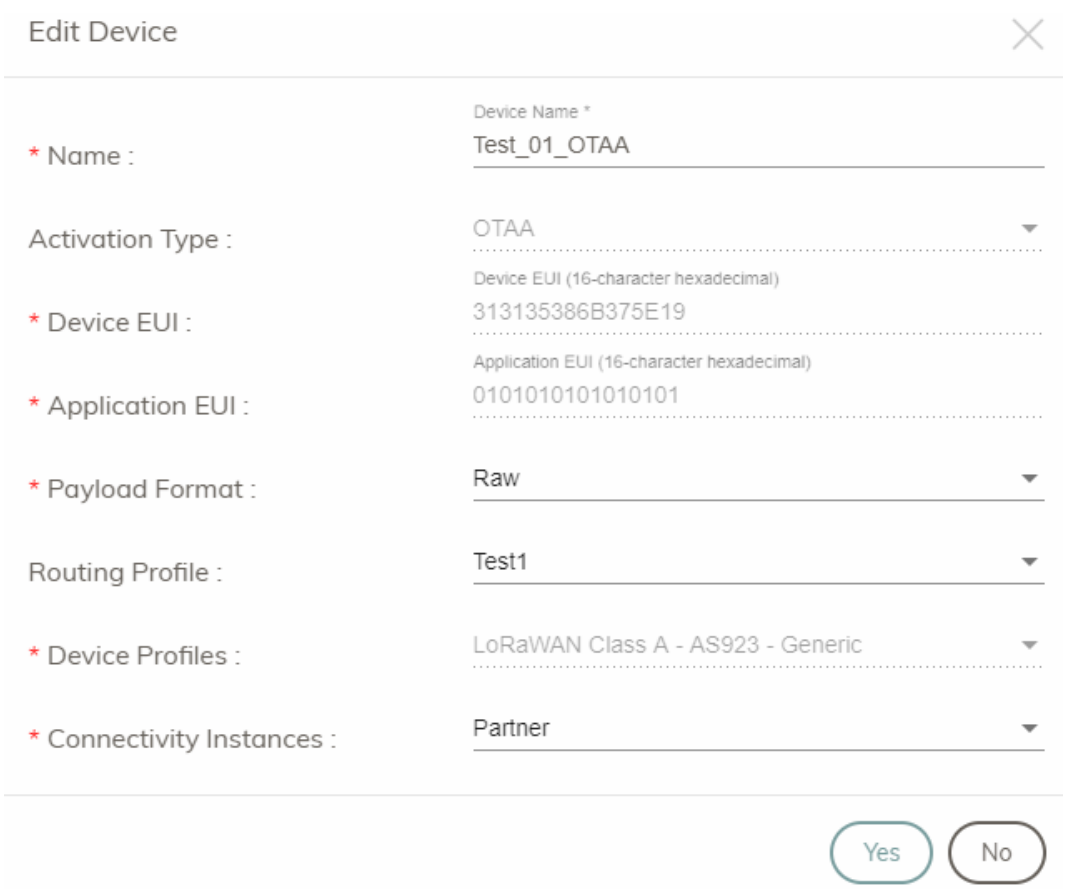
- Test AT command by send AT? and press enter on keyboard



AT-Command OTAA mode

- Example

1st get Device EUI on the board: AT+DEUI=?
2nd get App EUI on the board: AT+APPEUI=?
3rd get App Key on the board: AT+APPKEY=?
4th Join: AT+JOIN
5th check join status: AT+NJS=?
6th Send data out: AT+SEND=99:hello



End of the hands-on