INTERNET OF THINGS: STATE OF TECHNOLOGICAL ART ANALYSIS, IN ORDER TO IDENTIFY CURRENT AND FUTURE TECHNOLOGIES AND PLATFORMS OF THE IOT SECTOR

Platforms comparative with test/dev kits

Data: 09/23/2019

The ERDF project 2023 Beacon Südtirol (CUP: B31H17000060001) was funded by the European Development Fund Regional Autonomous Province of Bolzano, Investing for Growth and employment 2014-2020 ERDF.

Project partner

Planned and carried out by

GRUPPO FOS
soluzioni ad alta tecnologia

Giovanni Giannotta, Gruppo FOS

Gabriele Scarton, Gruppo FOS

Giorgio Allasia, Gruppo FOS

TECHPARK SÜDTIROL/ALTO ADIGE

Stefano Seppi, NOI S.p.a.

Patrick Ohnewein, NOI S.p.a.

In collaboration with

Fraunhofer ITALIA

Riccardo Brozzi, Fraunhofer Italia

David Forti, Fraunhofer Italia

Dominik Matt, Fraunhofer Italia

Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

Index

Methodologies and analysis methods....6
Development kits....7
ST Microelectronics B-L072Z-LRWAN1....7
Availability and delivery....8
Package details....8
Software....8
Getting Started with LoRaWAN....10
LoRaWAN Access configuration....12
Firmware Configuration....13
Upload firmware in the Board....13
LoRaWAN Tests....15
Getting started with Sigfox....16
Getting Sigfox ID and Keys for Development kit....19
Compiling example Projects....23
Charge firmware in the Board....23
Registering board on Sigfox back-end....24
Communication test....24
Common Tests....26
Power consumption....26
Stability....26
Nordic nRF52840 Development Kit....27
nRF52840 details....27
Availability and delivery....28
Package details....28
Software....28
Getting started....28
BLE Testing....29
Other connections testing....31
Conclusions on Nordic nRF52840 DK....36
DIGI XBEE S2C – DigiMesh 2.4 Kit....38
Availability and delivery....38
Package details....38
Software....39
Software management....40
Getting started with XBEE DIGIMESH and XCTU software....41
Firmware Configuration....43
XBEE Tests....47
DIGIMESH network range test....47
Indoor....47
Results....47
Outdoor....49
Results....50
Zigbee Firmware configuration....50
Evaluation of software platforms....52
SiteWhere....52
License....52

Software availability....52
Host System requirements....52
Installation....53
Sitewhere with HiveMQ brocker on Docker....53
Supported protocols....53
Databases....53
Test tools....54
DeviceHive....54
License....54
Software availability....54
Host System requirements....54
Installation....54
Supported protocols....55
Databases....55
Test tools....55
ThingsBoard Community Edition....55
License....55
Software availability....55
Host System requirements....56
Installation....56
Supported protocols....56
Databases....56
Test tools....57
WSo2 IoT....57
License....57
Software availability....57
Host System requirements....57
Installation....58
Protocols....60
Databases....61
Test tools....61
Zetta....62
License....62
Software availability....62
Host System requirements....62
Installation....62
Supported protocols....63
Databases....63
Test tools....63
Development kits conclusions table....64
SOB benchmark tests....65
Orange Pi Plus by Xunlong....65
Orange Pi i96 by Xunlong....71
Orange Pi 4G-IoT by Xunlong....72
Rock Pi 4A by Radxa....73
Nano Pi Duo2 by FriendlyArm....74
Banana Pi M2 Ultra by Sinovoip....75
Raspberry Pi....76
Results tables....77

Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

Methodologies and analysis methods

The main goal of this phase is to validate hardware and software platforms linked (or possibly linked to) IoT applications, analyzed in Phase 2.

The idea is to identify methodologies and tools for design and prototyping that make it easier and more efficient on the one hand the collection, manipulation and transfer of data, on the other hand the visualization and storage of the same data in an IoT context.

This work is the conclusion of the results of the activities carried out in Phase 1 and Phase 2 of the project, in which first a detailed analysis of the state of the art was made and then a comparative analysis of the applicable technologies was performed. From the latter activity, hardware and software technologies were selected to carry out an effective verification of the functions declared and indicated by the manufacturer and in the literature.

In Phase 2, in particular, a comparative study was first carried out to compare the pros and cons of different Internet of Things platforms available on the market. In this, those that seemed most appropriate for the purpose of the project were selected.

Subsequently, tests and validations were conducted with the platforms:

- Hardware platforms were tested on 2 different type of application - one for identified SOBs and one for kits related to transmission systems;
- Software platforms were tested using a hardware selection to determine platform behavior, working directly with different protocols and data transmission methods.

Other aspects of the platform have also been validated, such as data storage, visualization tools, integration with different databases and cloud services, alarm management and user management. The key point was to verify the actual availability and usability of the source code.

Obviously in order to be able to understand these tests it is necessary to learn how to correctly use all the technologies and the approaches used during this project.

Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

Development kits

In order to evaluate the development kits, these parameters were taken in consideration:

- Market availability, delivery methods and timing.
- Check of the content of the development kit packages
- Availability, type and ease of preparation of the software / firmware.
- First installation and preparation for first start.
- Stability on extended use of the configured system.
- Installation and/or modification of hardware and software components to make the device an End-Node or Gateway device.

ST Microelectronics B-L072Z-LRWAN1

STM32 LoRa and Sigfox Discovery kit for STM32L072 MCUCZY6TR Discovery board

This Board is a discovery and development kit for LoRaWAN and Sigfox connections by ST Microelectronics. The Board is equipage an STM32L072CZ Micro-controller , this Micro-controller Belongs to the category Of LOW Power (LP : Low Power) and 32 MHz max CPU frequency . Also Our Board Contain 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. The board ha a battery case for wireless use.

The radio modem is an CMWX1ZZABZ-091 LoRa/Sigfox™ module by Murata. The others fundamental characteristics are:

- 860-930MHz LPWA Module

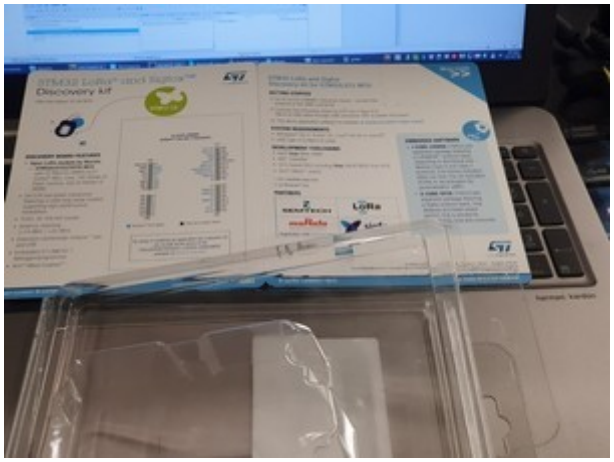
- Chipset: Semtech (SX1276) + STM (STM32L)
- Modulation: FSK, OOK and LoRa Modulation
- Small form factor LoRaWAN module
- MCU Chipset: STM32L0 Series
- CPU: ARM Cortex-M0+
- Peripheral Interfaces: I2C, UART, USB, SPI
- Pre-certified radio regulatory approvals: 868 & 915 MHz spectrum

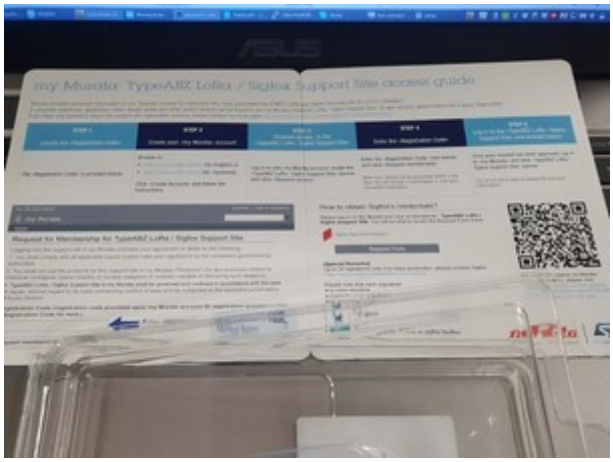
All characteristics are provided at https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html#resource.

Availability and delivery

Boards had been bought at the Italian section of Farnell online shop. Currently hundreds are available for about 40€-45€ each. It's possible to request a quote for a bulk order. Delivery is done by express courier in a week.

Package details





The package for a single board contains the Board with complete of an antenna for all frequency, in a transparent package. There is a paper instructions with a few board informations and the procedure for Sigfox activation.

Software

The board as a demonstration software, included in the STM32Cube package, is pre-loaded in the STM32 Flash memory. The latest versions of the demonstration source code and associated documentation can be downloaded from the www.st.com/i-cube-lrwan webpage.

For our test we used the firmware for STM32CubeL0, this solution is built around three independent levels that can easily interact with each other.

The system requirements are Windows® OS (7, 8 and 10), Linux® 64-bit or macOS®(a) and USB Type-A to Micro-B cable.

The firmware levels are:

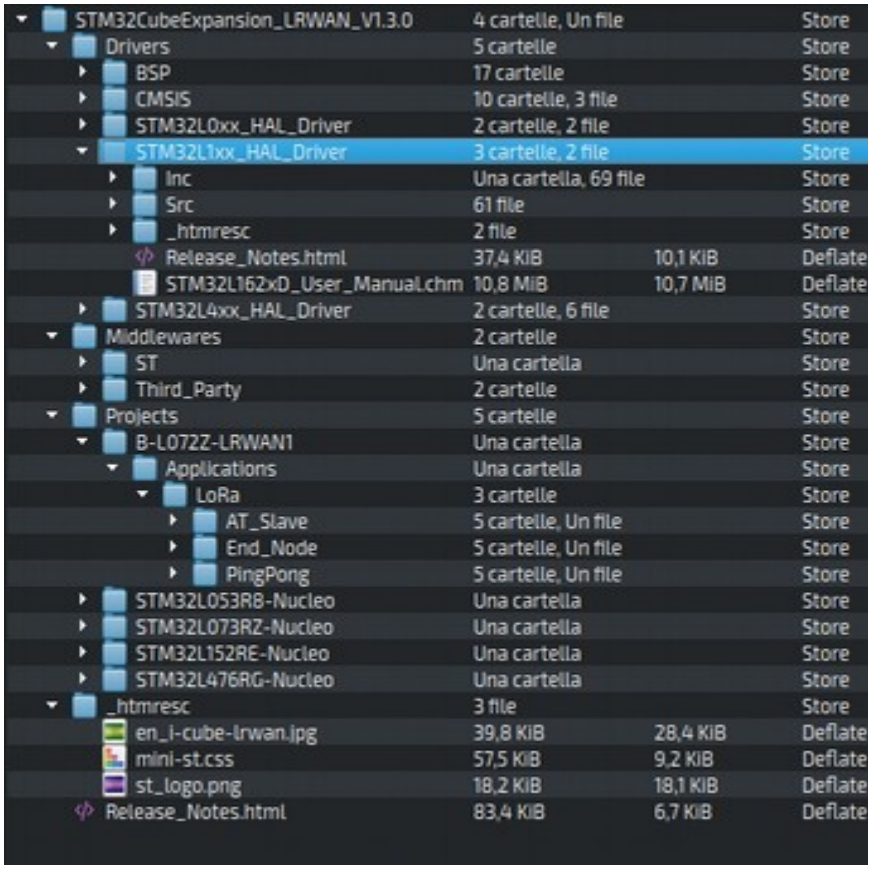
- Level 0: This level is divided into three sub-layers:

    - Board Support Package (BSP): this layer offers a set of APIs related to the hardware components on the hardware boards
    - Hardware Abstraction Layer (HAL): this layer provides the low level drivers and the hardware interfacing methods to interact with the upper layers
    - Basic peripheral usage examples: this layer contains examples of basic operation of the STM32L0 peripherals using only the HAL and BSP resources.
- Level 1: This level is divided into two sub-layers:
    - Middleware components: a set of Libraries covering USB Device Libraries, STMTouch touch sensing library, FreeRTOS and FatFS.
    - Examples based on the Middleware components: each Middleware component comes with one or more examples (called also Applications)
- Level 2: This level is composed of a single layer which is a global real-time and graphical demonstration based on the Middleware service layer, the low level abstraction layer and the applications that make basic use of the peripherals for board-based functions.

The ST-LINK/V2-1 requires a dedicated USB driver, which, for Windows® 7, 8 and 10 is available at the www.st.com website. In case the B-L072Z-LRWAN1 Discovery kit is connected to the PC before the driver is installed, some B-L072Z-LRWAN1 interfaces may be declared as "unknown" in the PC device manager. In this case the user must install the driver files and update the driver of the connected device from the device manager. We used a GNU Linux Debian distro for testing, and the board was directly recognized as device */dev/ttyACM0*.

The STM32CubeL0 firmware solution is provided in a single zip package.

For each board, a set of examples are provided with pre-configured projects for EWARM, MDK-ARM and either TrueSTUDIO or SW4STM32 toolchains.



The structure is identical for any other additional supported board.

The examples are classified depending on the STM32Cube level they apply to, and are named as follows:

- Examples in level 0 are called Examples, that use HAL drivers without any Middleware component
- Examples in level 1 are called Applications, that provide typical use cases of each Middleware component
- Examples in level 2 are called Demonstration, that implement all the HAL, BSP and Middleware components

A template project is provided to users for the quick build of any firmware application on a given board.

Getting Started with LoRaWAN

This section explains how to run our test. The tests are made for verifying the LoRaWAN Connection using the Beacon Südtirol-Alto Adige LoRaWAN Network (https://lorawan.beacon.bz.it/)

1. Download the STM32CubeL0 firmware package. Unzip the package into a directory of your choice. Make sure not to modify the package structure.
2. Browse to \Projects\STM32L053R8-Nucleo\Examples.

3. Open the ../STM32L072CZLorawanSigfox/STM32CubeExpansion_LRWAN_V1.3.0/Projects/B-L072Z-LRWAN1/Applications/LoRa/
4. Open the project with your preferred toolchain.
5. Select the Node Application and configure all LoRa parameters.
6. Rebuild all files and load your image into target memory.
7. Run the example: each second there are a various board information sent by LoRa network to the LoRa Südtirol – Alto Adige portal.

For opening and running the example we followed the instruction on “UM1754

User manual - Getting started with STM32CubeL0 firmware package for STM32L0 series” by ST Microelectronics. Below the common toolchains and how to use them:

- EWARM
  - Under the example folder, open the \EWARM sub folder
  - Open the Project.eww workspace(a)
  - Rebuild all files: Project->Rebuild all
  - Load the project image: Project->Debug
  - Run the program: Debug->Go (F5)
- MDK-ARM
  - Under the example folder, open the \MDK-ARM sub folder
  - Open the Project.uvproj workspace(a)
  - Rebuild all files: Project->Rebuild all target files
  - Load the project image: Debug->Start/Stop Debug Session
  - Run the program: Debug->Run (F5)
- SW4STM32
  - Open the \SW4STM32 toolchain
  - Click File->Switch Workspace->Other and browse to the SW4STM32
  - workspace directory
  - Click File->Import, select General->Existing Projects into Workspace and then
  - click Next
  - Browse to the SW4STM32 workspace directory and select the project
  - Rebuild all project files: select the project in the “Project explorer” window then

- click the Project->build project menu
- Run program: Run->Debug (F11)

- TrueSTUDIO
  - Open the TrueSTUDIO toolchain
  - Select on File->Switch Workspace->Other and browse to the TrueSTUDIO
  - workspace directory
  - Click on File->Import, select General->'Existing Projects into Workspace' and
  - then click “Next”.
  - Browse to the TrueSTUDIO workspace directory, select the project
  - Rebuild all project files: Select the project in the “Project explorer” window then
  - click on Project->build project menu.
  - Run the program: Run->Debug (F11)

Our choice was TrueSTUDIO.

LoRaWAN Access configuration

1. https://lorawan.beacon.bz.it/
2. Select a Register button and accept the Disclaimer

3. Insert all required informations and Register

4. Login in the portal

5. Add a new application or select existing one

6. Add a new device or select existing one.
7. Get the device informations
8. Follow the instruction in Get data for accessing to the data.

Firmware Configuration

1. In TrueSTUDIO in the project we made the following changes:
    1. Projects\End_Node\main.c\Commissioning.h.
    2. In the Commissioning.h file :
        1. #define OVER_THE_AIR_ACTIVATION 0
        2. #define STATIC_DEVICE_EUI 1
        3. #define STATIC_DEVICE_ADDRESS 1
        4. #define LORAWAN_DEVICE_EUI with (Assigned LoRaWAN Device EUI)
        5. #define LORAWAN_JOIN_EUI with (Our LoRaWAN Application EUI)
        6. #define LORAWAN_APP_KEY with (Assigned LoRaWAN App Session Key)
        7. #define LORAWAN_DEVICE_ADDRESS with (Our LoRaWAN Device Address)
        8. #define LORAWAN_F_NWK_S_INT_KEY with (Assigned Network Session Key)
        9. #define LORAWAN_S_NWK_S_INT_KEY with (Assigned Network Session Key)
        10. #define LORAWAN_NWK_S_ENC_KEY with (Assigned Network Session Key)
        11. #define LORAWAN_APP_S_KEY with (Assigned LoRaWAN App Session Key)

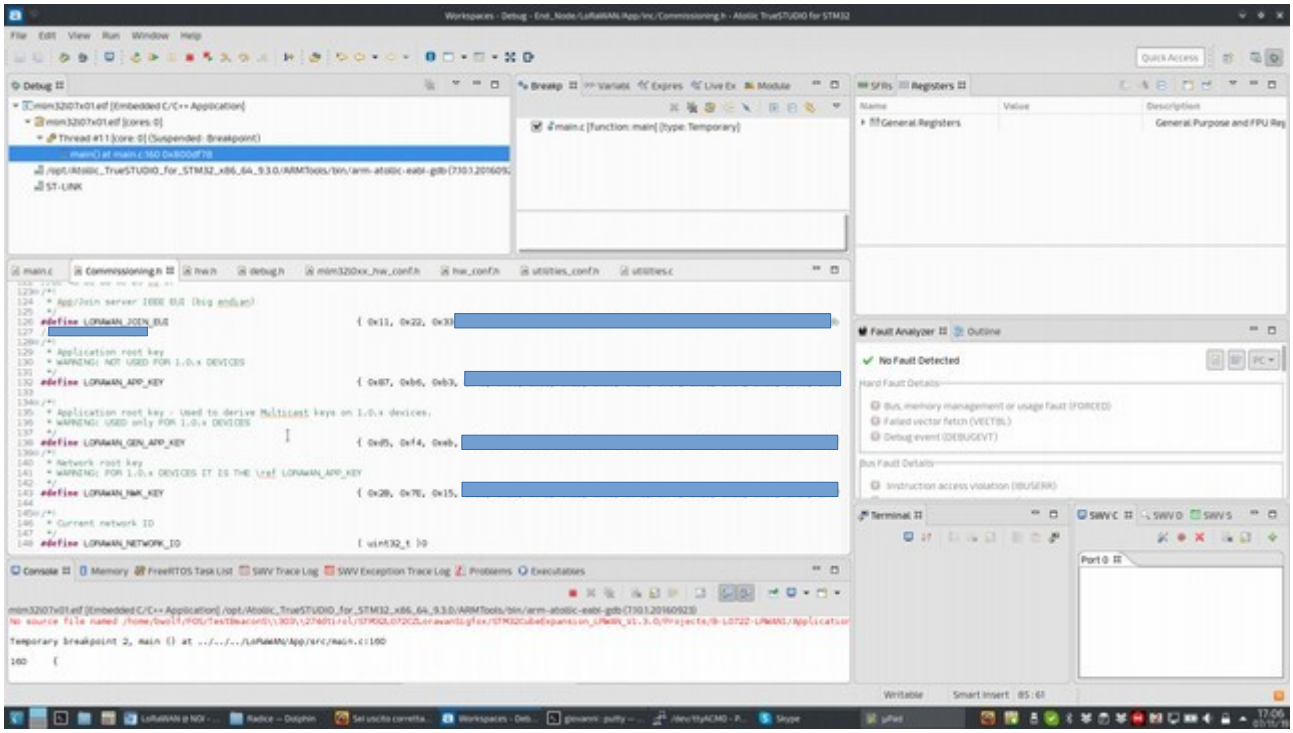
Make attention to use a Hex Format for:

- LoRaWAN Device EUI
- LoRaWAN Application EUI
- LoRaWAN App Session Key
- Network Session Key

Upload firmware in the Board

- Rebuild all project files: Select the project in the “Project explorer” window then
- click on Project->build project menu.
- Connect the board to the PC
- Select in debug window the ST-LINK debug probe

- Run the program: both Run->Debug (F11) and push the reset button on the board



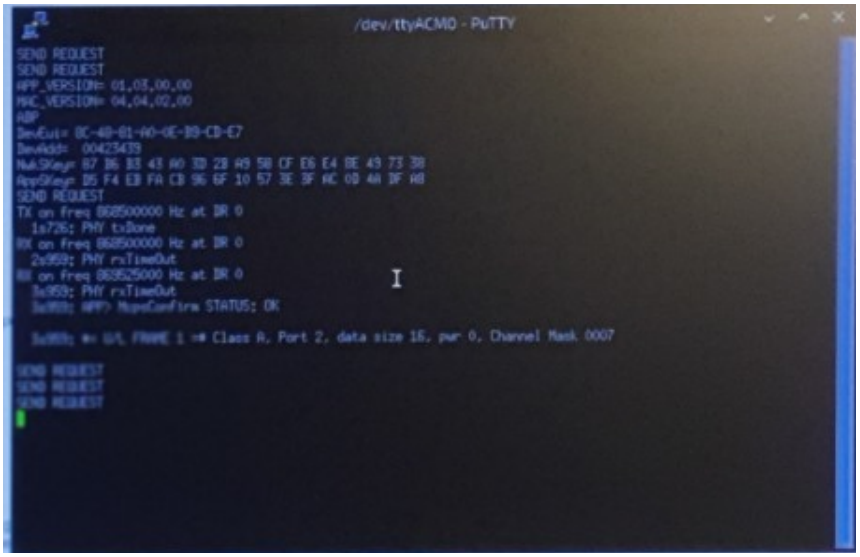
Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

LoRaWAN Tests

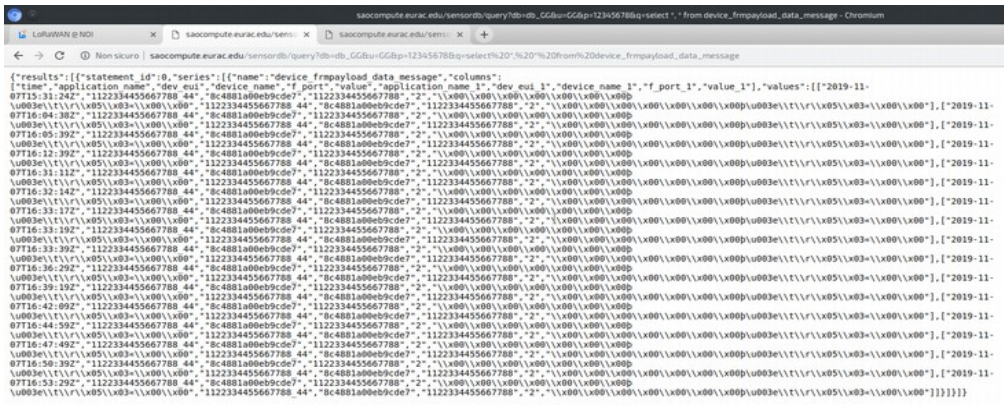
Data transfer and link quality

Some time the data didn’t received, we supposing there are a slot time limit in the LoRaWAN network.

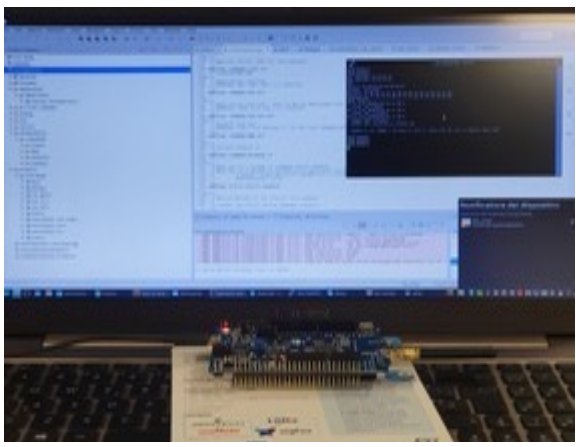
Sending data:

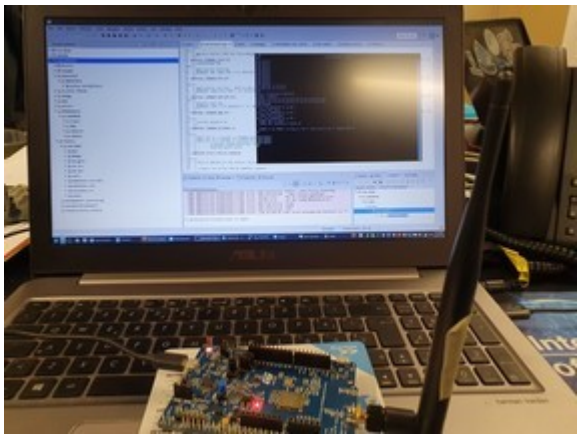


Received data



Link quality in real conditions





Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

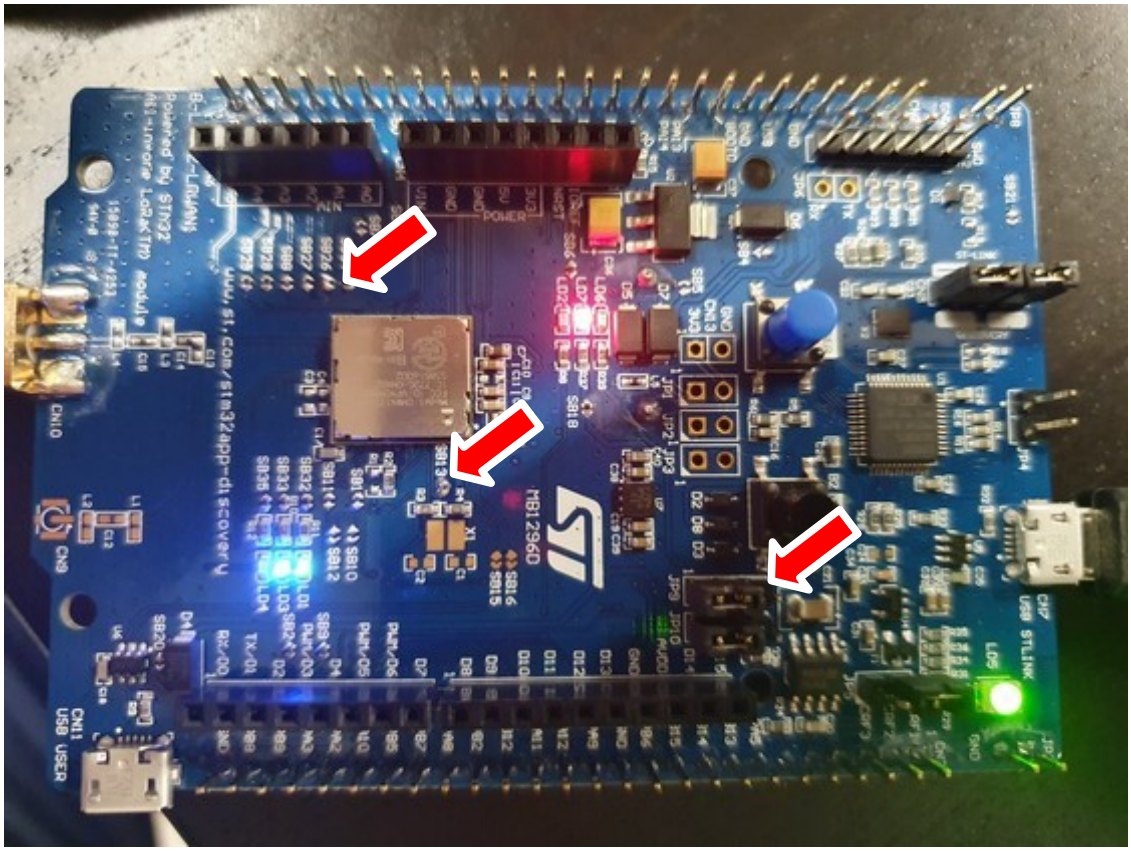
For this test we used 2 different configurations, with 2 kind of antennas: the original one, ad the other one more powerful. The test is made at the limit of LoRaWAN Gateway coverage and in both cases we made the connection and received the data.

Getting started with Sigfox

- Use the preferred toolchain.
- X Cube Sigfox software expansion from ST (http://www.st.com/content/st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32cube-expansion-packages/x-cube-sfox.html)

In order to use Sigfox connection we did the hardware kit.

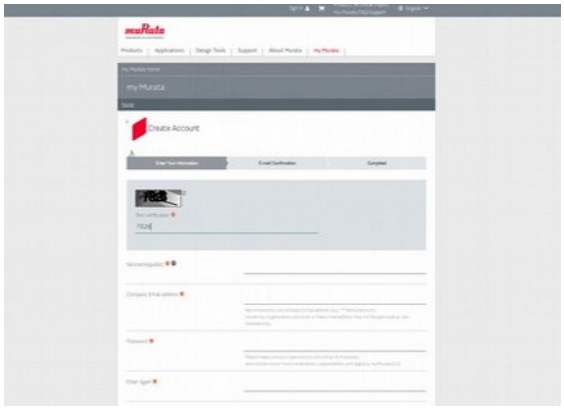
A few hardware modifications need to be done to prepare the board. Bridges SB13 and SB26 need to be soldered. Also jumper JP9 must be connected on pins 1 and 2:

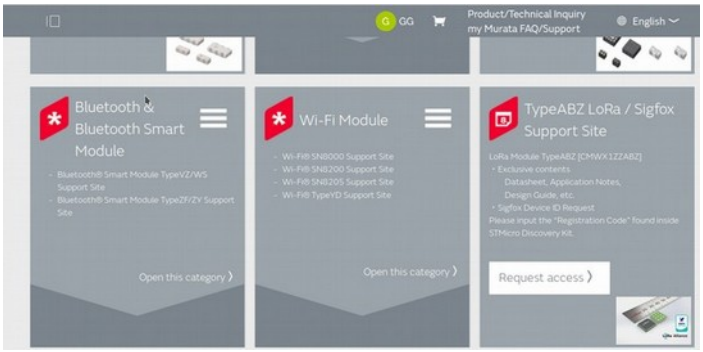


Getting Sigfox ID and Keys for Development kit

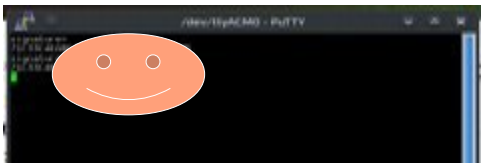
First step is to retrieve the specific signature of the Murata module.

Firstable sign in in my.murata website using a Registration code present in the paper instruction annex to the development kit.





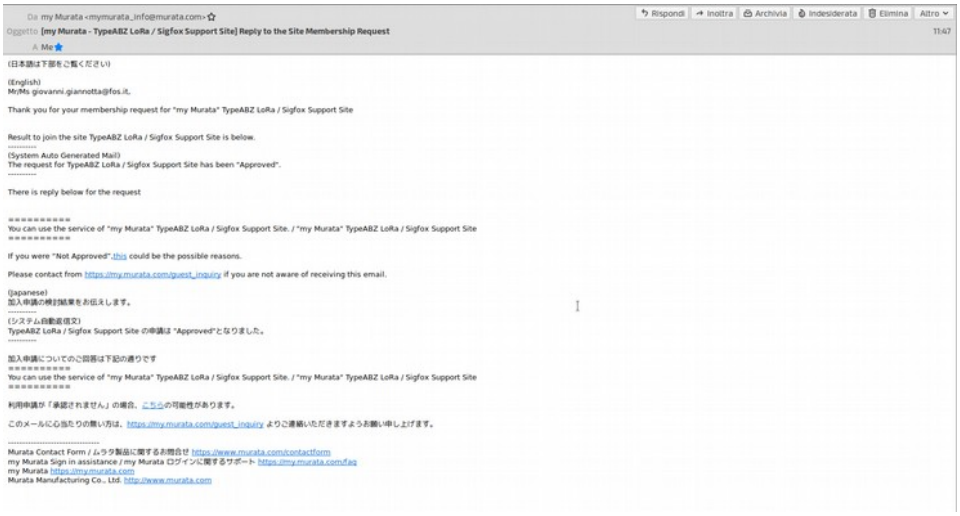
1. open the X-Cube-Sfox package previously downloaded. Browse to "Projects\B-L072Z-LRWAN1\Applications\Sgfx\SignatureGenerator" and copy SignatureGenerator.bin into the board (the board should be seen as an external drive):



Once the file is transferred, open the */dev / ttyACM0* port using a terminal (baud-rate: 9600) application and reboot the board. Signature will be displayed in the terminal:

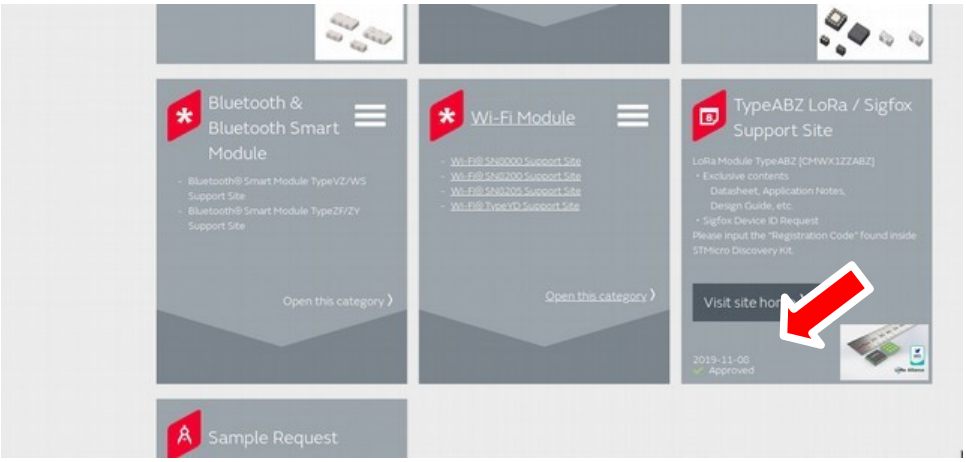
Through my.murata website, is it possible request up to 20 Sigfox IDs for the dev kit / custom board. To do so, register on https://my.murata.com, then request to have access to "TypeABZ LoRa / Sigfox Support Site".

Here the email answers for my.murata registration:

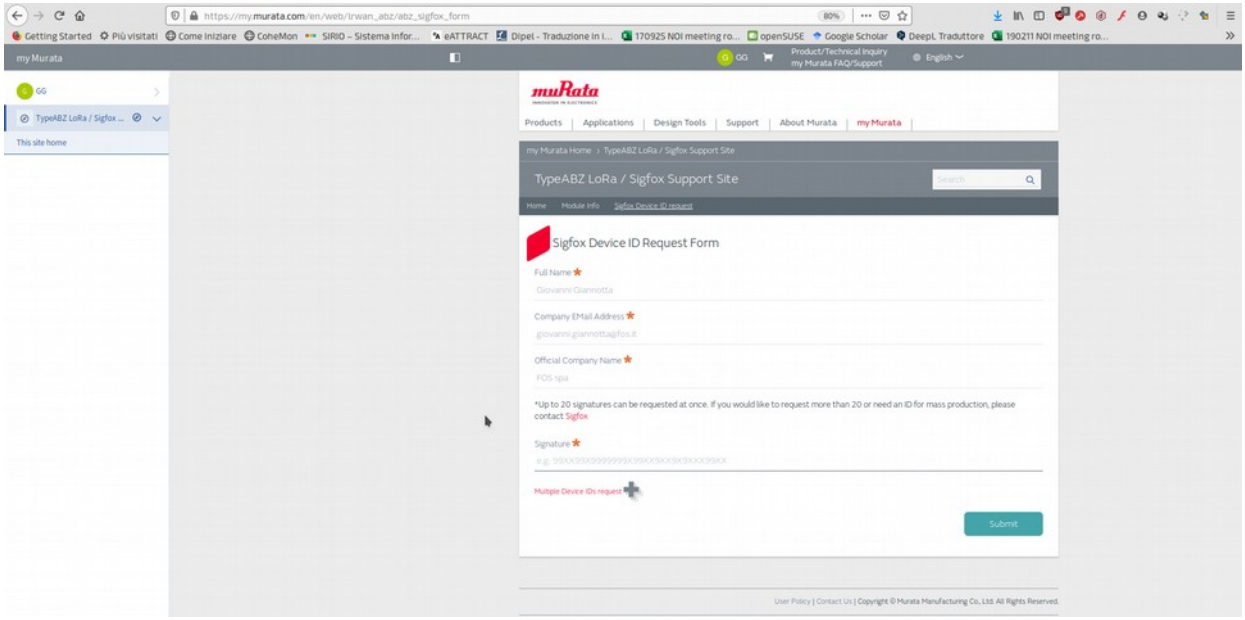


As mentioned by the website and the documents attached to the board, it may take a few days to then receive the email from Murata with the sigfox_data.h file. Our request was satisfied less than 2 hours.

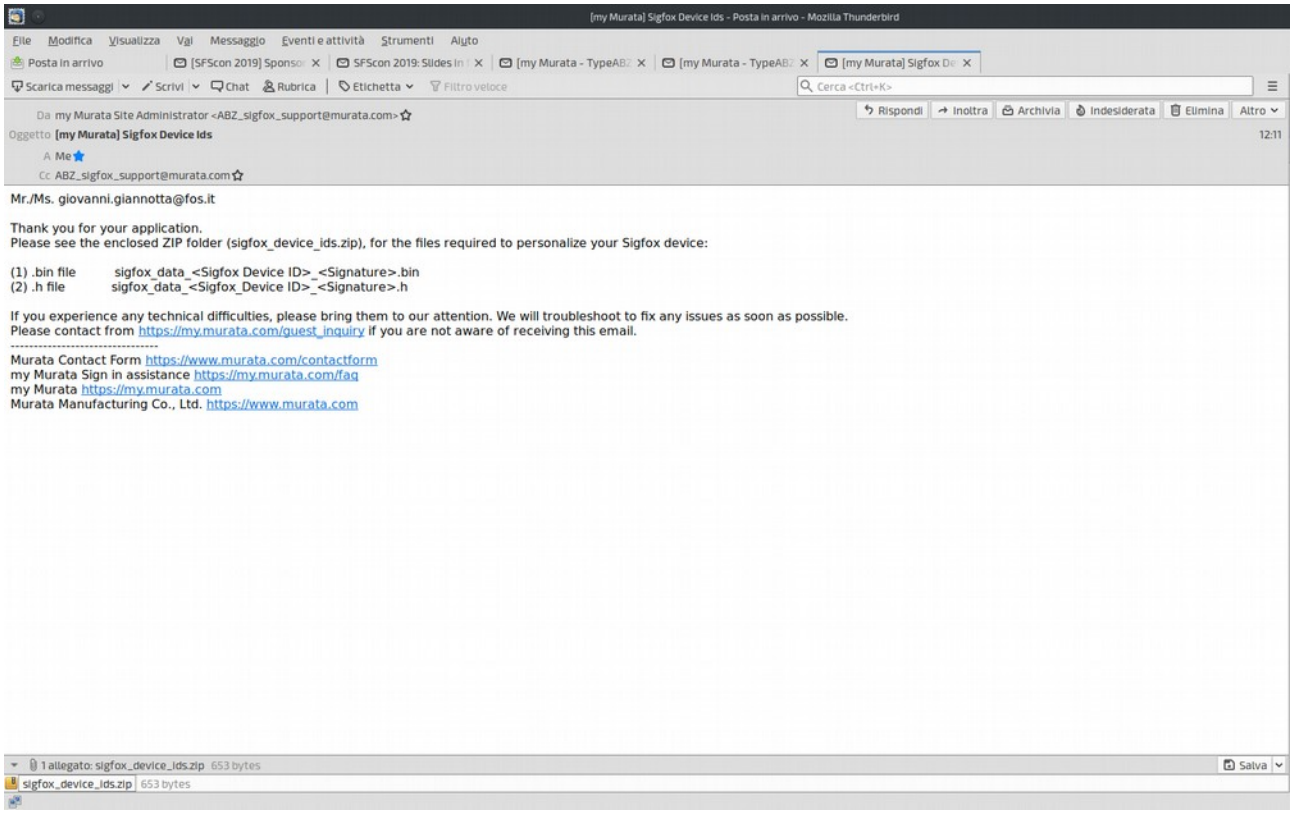
On the my.murata site we see:



3. Follow the Visit site home and select Sigfox Device ID Request



After 10minutes we had the email with sigfox_data.h file annex, here the email:



Compiling example Projects

- From the IDE importing a new project (File => Import projects from file system). Select for instance the PushButton project from X-Cube package, subdirectory SW4STM32: *../STM32CubeExpansion_SFOX_V1.1.0/Projects/B-L072Z-LRWAN1/Applications/Sgfx/Sgfx_push_button/*
- *Select the mlm32xxxx Eclispe project and import.*
- Once imported, right click on the PushButton project and select *Build Project*:

By default, project is using a Sigfox default ID = 78563412. This can be used as a first step just if it present a Sigfox SDR Dongle to test the communication. To use the real ID, it is need to copy the sigfox_data.h sent by Murata, and copy it in the following directory:
../STM32CubeExpansion_SFOX_V1.1.0/Projects/B-L072Z-LRWAN1/Applications/Sgfx/Sgfx_push_button/inc

- Rebuild the project to take into account the new ID and PAC information.

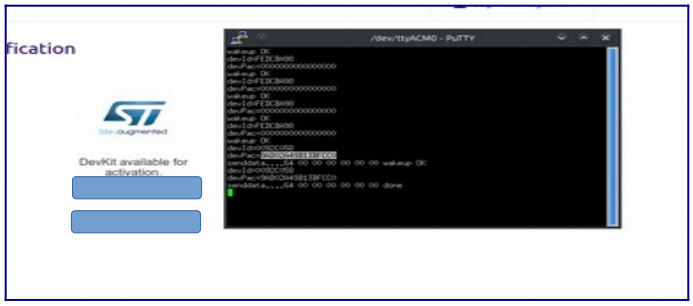
Charge firmware in the Board

- Rebuild all project files: Select the project in the “Project explorer” window then
- click on Project->build project menu.
- Connect the board to the PC
- Select in debug window the ST-LINK debug probe
- Run the program: both Run->Debug (F11) and push the reset button on the board

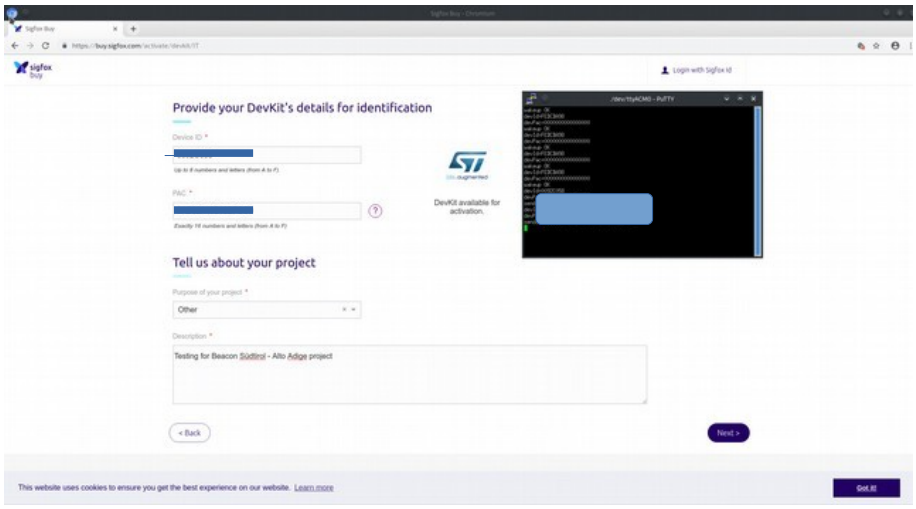
- Last step is to flash the code on the board. To do so, right click on the project and select: *Target => Program chip*

Registering board on Sigfox back-end

With the Push Button application loaded on the board, open the COM port in a terminal (baudrate: 9600). Push the black button to display ID and PAC information of your board:

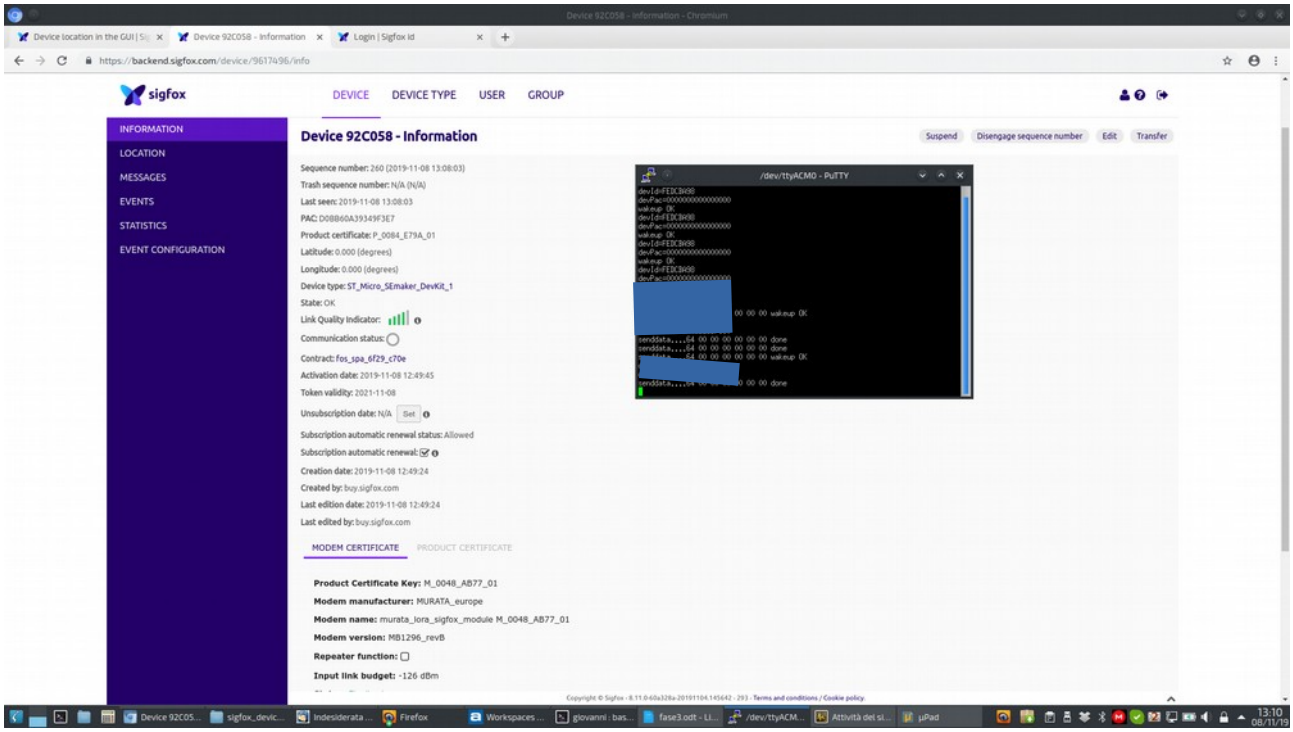


Go to https://backend.sigfox.com/activate to register your board. Select muRata as provider, pick your country and enter your ID and PAC data. The device is now ready to communicate with the network.



Communication test

To test communication with Push Button application, push the Blue Button to send a random payload of 7 bytes:



In the image we can see the connection quality, the last connection, and other nice informations.

Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

Common Tests

Power consumption

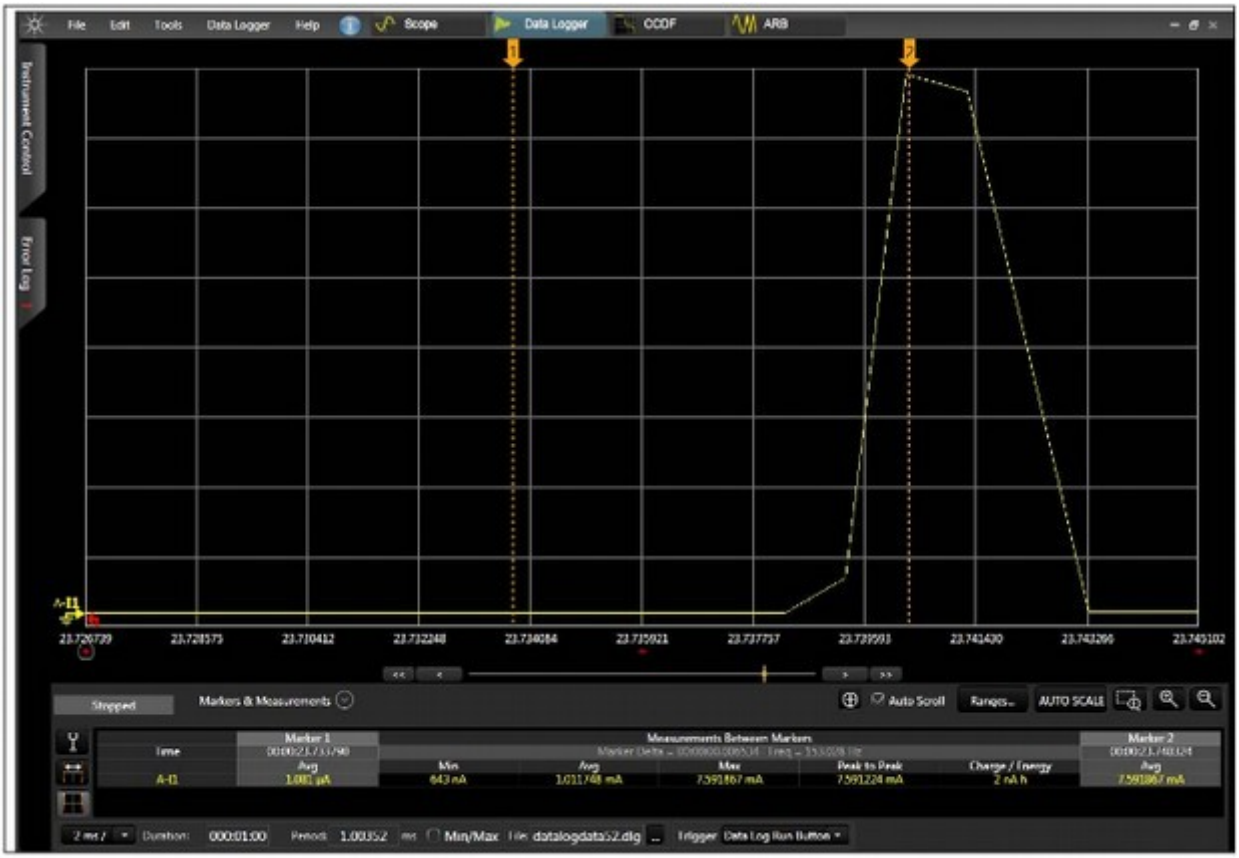
Measurements setup:

- No DEBUG
- No TRACE
- No SENSOR_ENABLED

Measurements results:

- Typical consumption in stop mode: 1.3 µA
- Typical consumption in run mode: 8.0 mA

In Figure shows an example of the current consumption against time on a microcontroller of the STM32L0 Series



Stability

The system was in test for one day with connection and some tests without make any problems or freezing.

Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

Nordic nRF52840 Development Kit

The development kit is a single board that facilitates nRF52840 development, exploiting all features. The nRF52840 is a SoC with protocol support for Bluetooth 5, Bluetooth mesh, Thread, Zigbee, 802.15.4, ANT and 2.4 GHz proprietary stacks. It is Arduino Uno Revision 3 compatible, making it possible to mount their 3rd-party shields. The kit has access to all I/Os (48) and interfaces via connectors and there is an integrated PCB trace antenna and an RF connector for direct RF test measurements. There is also a connector for an external NFC antenna (included in kit).

nRF52840 details

- 64 MHz Arm® Cortex-M4 with FPU1 MB Flash + 256 KB RAM
- Bluetooth 5 multiprotocol radioy (2 Mbps, CSA #2, Advertising Extensions, Long Range, +8 dBm TX power, -95 dBm sensitivity, integrated balun with 50 Ω single-ended output)
- IEEE 802.15.4 radio, supporting Thread and Zigbee
- 1.7-5.5 V supply voltage range
- Full-speed 12 Mbps USB
- NFC-A tag
- Arm CryptoCell CC310 security subsystem
- QSPI/SPI/TWI/I²S/PDM/QDECHigh speed 32 MHz SPI
- Quad SPI interface 32 Mhz
- EasyDMA for all digital interfaces
- 12-bit 200 ksps ADC
- 128 bit AES/ECB/CCM/AAR co-processor
- On-chip DC-DC buck converter
- Regulated supply for external components up to 25 mA

Availability and delivery

Boards had been bought at the Italian section of Farnell online shop[1]. Currently[2] hundreds are available for about 40€-45€ each. It's possible to request a quote for a bulk order. Delivery is done by express courier in a week.

Package details

The package for a single board contains the nRF52840 board, an NFC antenna to plug in the board and a coin cell battery inserted in the battery holder and isolated from the board by a plastic string. There is no paper instructions, only a link printed to the package:
https://nordicsemi.com/start52840dk

Software

In order to control and develop applications, these tools had been downloaded/installed in a ArchLinux-based operating system, but they are also available for Linux-based and Windows operating systems:

- ARM Segger Embedded Studio (tested version 430a for Linux x64): https://www.segger.com/products/development-tools/embedded-studio/
- nRF Command Line Tools (tested version 10.2.1-2 installed via AUR): https://www.nordicsemi.com/Software-and-Tools/Development-Tools/nRF-Command-Line-Tools
- Jlink software and documentation (tested version 21:6.54a-0 installed via AUR): https://www.segger.com/downloads/jlink#J-LinkSoftwareAndDocumentationPack
- nRF5 SDK (this is a folder, no installation needed. Tested version 16.0.0): https://infocenter.nordicsemi.com/index.jsp?topic=%2Fstruct_sdk%2Fstruct%2Fsdk_nrf5_latest.html

For connecting the board to the computer an USB type-A to microUSB is required (it's not included in package). The board should be automatically recognized as removable media by any OS.

Getting started

The very first test had been done following the "Getting started" chapter of the official user guide[3]. After connecting the NFC antenna, checking that power source switch is on "VDD", connecting the USB cable and then switching on the power switch, the board was recognized as device on pc and LED1 started blinking. Also pushing one of the four input buttons, the blinking led changed to the led with the same number of the button pressed, confirming that the installed firmware is running correctly.

1 https://it.farnell.com/nordic-semiconductor/nrf52840-dk/dev-kit-bluetooth-low-energy-soc/dp/2842321
2 November 2019
3 https://infocenter.nordicsemi.com/pdf/nRF52840_DK_User_Guide_v1.2.pdf

In order to run the very first example from an external source, the hex file for leds blinking example from SDK was copied in the device using the computer file explorer and the power button is switched off and then on. All the 4 leds started blinking at different timings, confirming that example had been successfully loaded.

BLE Testing

This section explains how to run a Bluetooth 5.0 test, assuming that all softwares indicated above had been installed and an SDK had been downloaded. If you would like to maintain an original copy of the SDK, make a copy of it before starting using its examples.

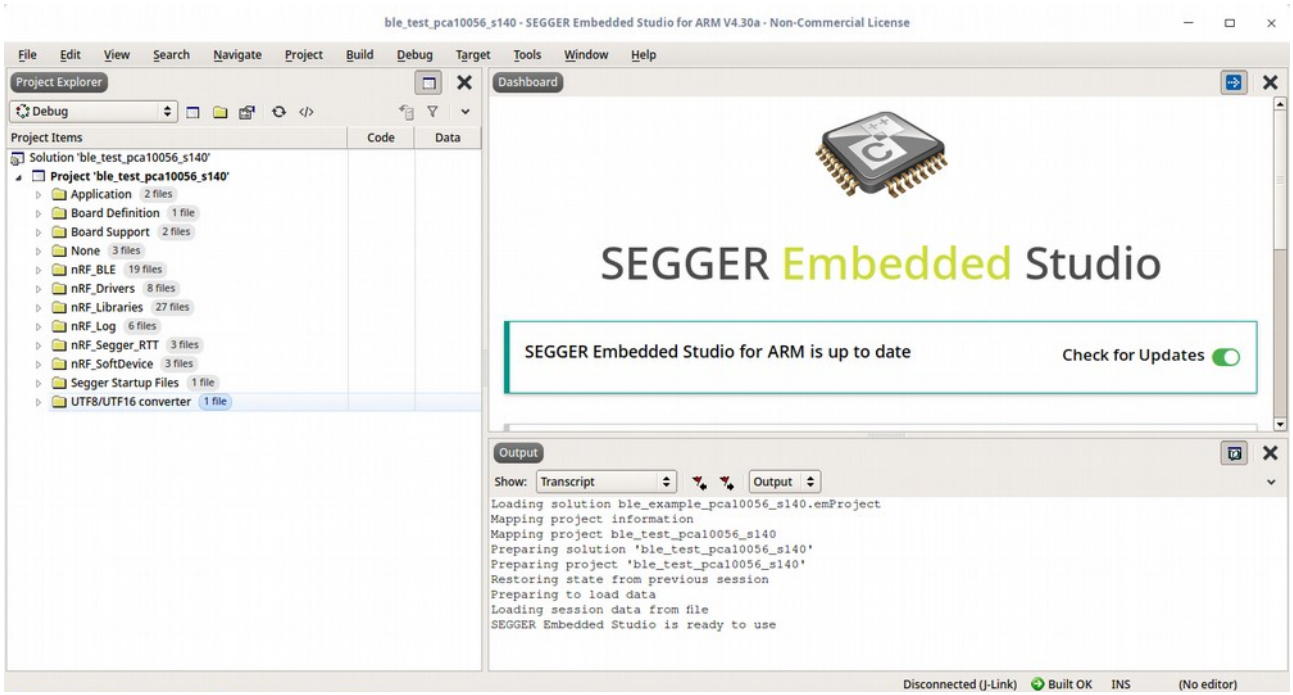
1. Open ARM Segger Embedded Studio and import the ble_app_template solution clicking on File → Open solution and following this path:

   a) Browser the SDK to examples/ble_peripheral/ble_app_template

   b) Browser to the folder with the code of the board (*PCA10056)*

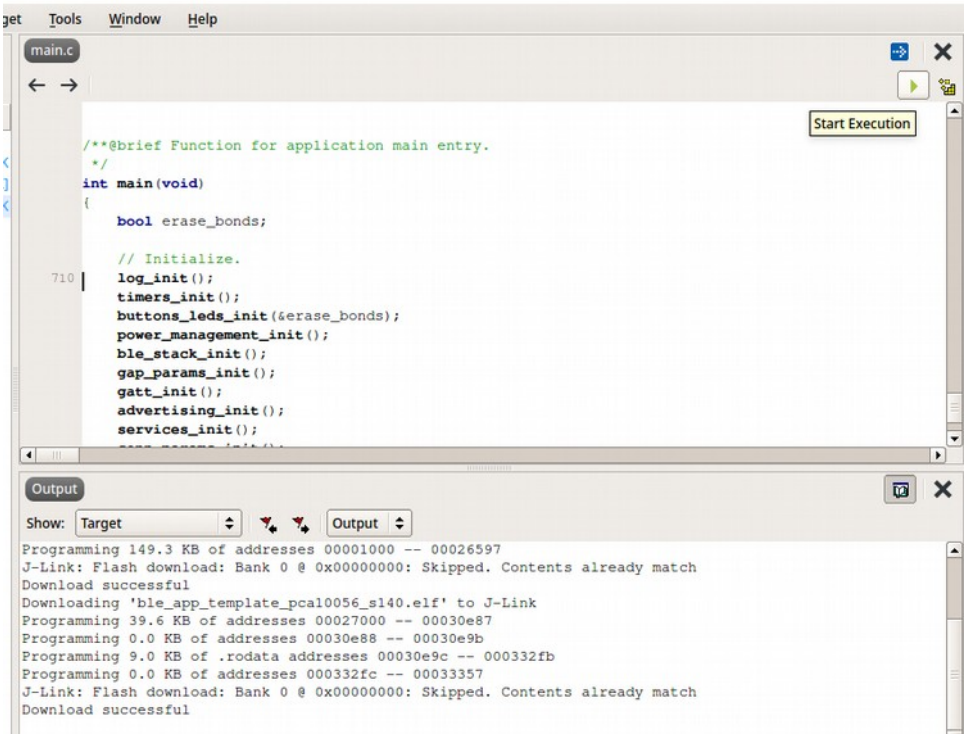
   c) Browser the s140 folder and the folder associated to the IDE used (*ses*)

   d) Open the .emProject with a text editor and modify all paths to match the SDK location. Using a “find and replace all” tool could be useful to replace the “../../” to the actual SDK path.

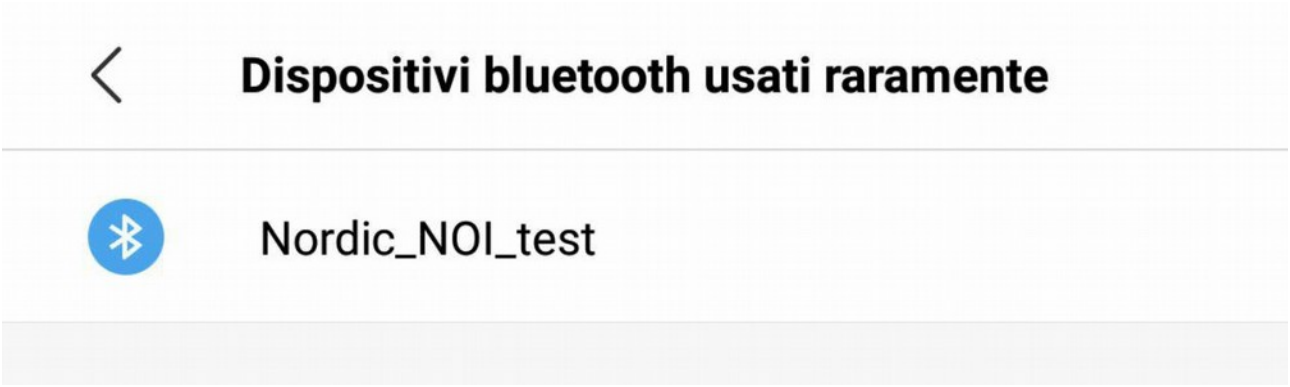


2. Inside Application→main.c, modify #define DEVICE_NAME to something more specific. You can also modify timing parameters.
3. Build the project rightclicking or using the build tab
4. Connect via usb the board and switch on the power button
5. Connect the board via Jlink feature clicking Target → Connect Jlink

6. Now you can run the code pressing the green triangle over the code:



Now the device is running and the device is visible:



Other connections testing

Using the method described above and using 2 boards with the examples provided in the SDK, other connections had been tested:

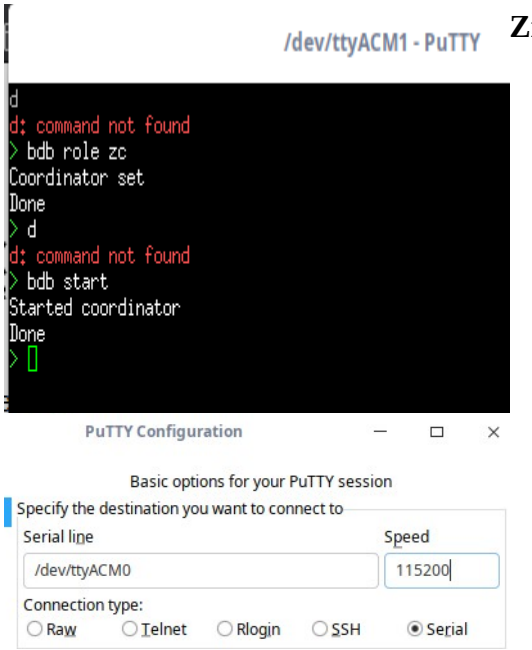
Wireless:

1) Installed PuTTY for serial communication
2) Prepared 2 boards:
   a) One with build from {SDK}\examples\802_15_4\wireless_uart\secure\first
   b) One with build from {SDK}\examples\802_15_4\wireless_uart\secure\second

3) Used PuTTY to connect both boards to pc (baud rate 38400)
4) Type something in one terminal to see the output in the other one

Thread:

1) Downloaded the SDK specified for Thread and ZigBee: https://www.nordicsemi.com/Software-and-Tools/Software/nRF5-SDK-for-Thread-and-Zigbee
2) Prepared 2 boards:
   a) One as server, with build from {SDK}\examples\thread\simple_coap_server
   b) One as client, with build from {SDK}\examples\thread\simple_coap_client
3) The client board was switched on at different distances and control of LED4 on the server via a button on the Client was tested to determine the connection.

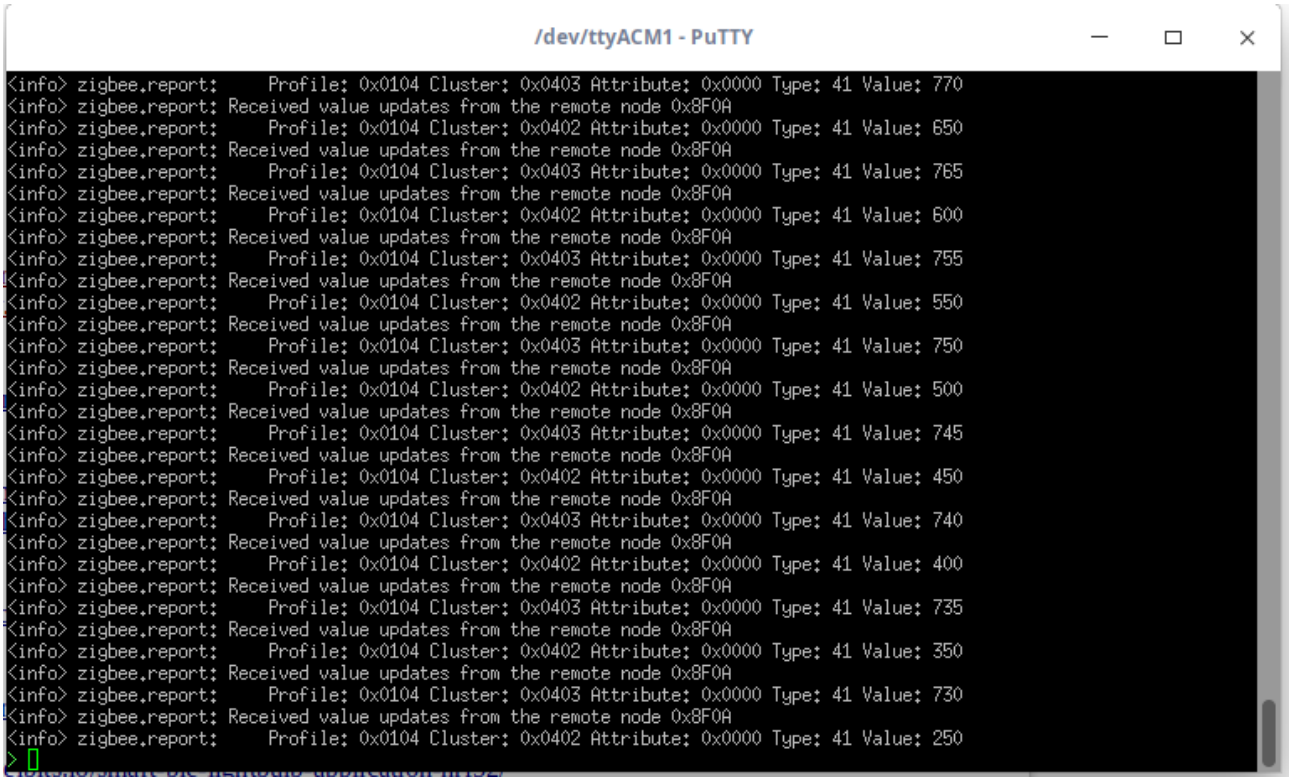


ZigBee:

1. Downloaded the SDK specified for Thread and ZigBee and installed PuTTY for serial communication
2. Prepare 2 boards:
   a) One as coordinator with build from {SDK}\examples\zigbee\experimental\cli\cli_agent_router
   b) One as client, with build from {SDK}\examples\zigbee\experimental\multi_sensor
3. Switch on both boards and connect the coordinator one via serial to the computer. In order to do so, you have to connect your pc to the lateral microUSB port (not the one used for power and flashing rom). Be careful: powering the board with the lateral port don't switch on the board, for startup it must be powered by the coin battery or the other microUSB port. After doing that, you can set up PuTTY for connection and use CLI comands (https://infocenter.nordicsemi.com/index.jsp?topic=%2Fsdk_tz_v3.2.0%2Fzigbee_example_cli_reference.html) to set up the board as coordinator.
4. Scan the network for the other board using `zdo match_desc 0xffff 0xffff 0x0104 2 0x0402 0x0403 0` command. The answer should be `src_addr=XXXX,` where XXXX is the address of the board

Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

5. Use the address of the client board to get the long address: `zdo ieee_addr 0xXXXX.` The answer should be a exadecimal address with 16 characters
6. Get the address of the coordinator board: zdo eui64
7. Bind the remote endpoint clusters to the CLI: `zdo bind on <coordinator addr> 10 <client addr> 64 0x0402 0xXXXX; zdo bind on <coordinator addr> 10 <client addr> 64 0x0402 0xXXXX`
8. Subscribe the coordinator to both clusters: `zcl subscribe on <coordinator> 10 0x0402 0x0104 0 41, zcl subscribe on <coordinator> 10 0x0403 0x0104 0 41`



9. Enable log to CLI: `log enable info zigbee.report` and see the results (the example is based on two sensors, since none was linked to the board, data is meanless but it proofs connectivity).

ANT:

Regarding the ANT protocol, the board supports the communication and some code examples had been found for testing the protocol, however all examples refers to ANT+ Alliance softwares that are now restricted to members of alliance. Some documentations refer to them as free software but most of the links are now expired or unavailable without subscription. The minimum subscription level for using them is the free one and then the download and the use of softwares are free too, but you have to accept their contract.

Conclusions on Nordic nRF52840 DK

The development kit really exploits all features of the nRF52840 chip. Thanks to an comprehensive documentations, updated at every SDK version, verifying the connections of all protocol was simple. All protocols except one had been successfully tested with results almost identical to the one expected reading the documentation. The ANT connectivity was the only one impossible to test with open source softwares, due to restrictions of ANT+ Alliance that had made softwares still free but restricted to members. Another plus for this board is the official community forum for developers of Nordic boards (https://devzone.nordicsemi.com/), which put together the official support with the community support.

DIGI XBEE S2C – DigiMesh 2.4 Kit

As described in the data-sheet, Digi XBee S2C DigiMesh 2.4 Kit offers a great way to learn how to use Digi XBee RF modules for device connectivity and ZigBee-based mesh networking.

Digi XBee S2C DigiMesh Included in the KitDigi XBee and Digi XBee-PRO S2C DigiMesh modules are embedded solutions providing wireless connectivity to devices. These modules use the proprietary Digi International, Inc. mesh networking protocol DigiMesh for peer-to-peer and mesh networking. DigiMesh is a robust mesh networking protocol designed to reduce power consumption and increase throughput by reducing the housekeeping chatter between modules. Module function is based on the tasks assigned to it rather than by the specific firmware sub-version loaded making it a true peer to peer mesh network.

Availability and delivery

The Kit had been bought at the Mouser online shop. Currently there are 5 kits available on Mouser and Digi-key as can see on the digi.com portale, for about 80€-90€ each. It’s possible to request a quote for a bulk order. Delivery is done by express courier in a week.

Package details

The package consisting in a plastic box with:

- 3 Digi XBee Grove Development Boards
- 2 Digi XBee DigiMesh Modules (XB24CDMPIT-001)
- 3 Micro-USB Cables
- 2 Digi XBee Stickers

and an isolated vacuum packed with 1 Digi XBee DigiMesh modules XB24DMPIS-001.

This last module need a particular attention for it’s stability measurement.

Both plastic box and vacuum package are in a cardboard packaging.

No instruction are inside the packages.

Software

Digi recently built out a collection of tools for every phase of the wireless application lifecycle, from prototyping the wireless design to configuring and testing, building the configured devices and deploying them in the wireless network.

The full set of tools and resources on the Digi XBee Tools page.

Here we are reporting all interesting described tools on the https://www.digi.com/blog/digi-xbee-tutorials-and-resources-for-developing-wireless-applications/ site.

Application development:

- Digi XCTU – The next generation configuration platform for XBee/RF solutions, providing multiple tools in one. See “How to find Digi XCTU Tutorials” below.
- Digi XBIB-C Development Boards – These full-featured boards provide a prototyping environment for rapidly designing and testing the functionality of your wireless design.
- Digi XBee GPS Daughter Board – An expansion module for the Digi XBIB-C board that enables you to add and test GPS functionality in your wireless design.
- Digi XBee MicroPython PyCharm IDE Plugin – A tool to simplify and expedite the processes involved in developing, compiling and flashing code, using PyCharm IDE’s built-in developer tools such as code completion, error checking and project navigation.
- Digi XBee MicroPython – Many Digi XBee modules integrate with MicroPython, and to support rapid development, Digi supplies MicroPython code libraries. Visit our XBee MicroPython GitHub site to explore the resources.

Building and programming:

- Digi XBee Multi Programmer – A tool that enables technicians to program up to six Digi XBee modules at once to accelerate time-to-market and deployment. These devices are available for all Digi XBee form factors.

Deploying personal wireless network:

- Digi XBee Network Assistant – A desktop utility that helps you visualize and map your local Digi XBee network, analyze network strength between nodes and send batch firmware updates to all nodes on the network.
- Digi XBee 3 USB Adapter - A programmable device that works with any of Digi’s supported wireless protocols, and is designed to provide connectivity to a local Digi XBee network from a laptop or PC to commission the network and connected devices.
- Digi XBee Mobile App – A downloadable app that offers the same functionality as Digi XCTU, but is Bluetooth-enabled for field work, enabling you to quickly configure, commission and update firmware wirelessly at the deployment site.

Managing the deployed Digi XBee network:

- Digi Remote Manager® – A cloud-based IoT device and network management tool that provides all the functionality you need to monitor your network in real time, perform firmware updates, reboot devices and manage your device network.

The test that we did are made for verifying the platform simplicity and connection quality.

Software management.

XCTU is a free multi-platform application that enables developers to manage Digi radio frequency (RF) modules through a simple-to-use graphical interface. The application includes embedded tools that make it easy to set up, configure, and test Digi RF modules[4].

Use XCTU to:

- Deploy on multiple platforms: XCTU is compatible with the most-used operating systems, including Microsoft Windows, Linux, and Mac OS.
- Discover your modules: Automatically discover local and remote radio modules connected to your PC, regardless of their port connections or configured settings.
- Configure any module: Manage and configure multiple RF devices, including remote devices communicating with XCTU over the air.
- Communicate with your modules: Use API and AT command consoles to communicate with your radio modules. Record your console sessions and load them at any time.
- Explore your network: Visualize the topology of your RF networks, displaying all network nodes and connections graphically or in a table.
- Access a range of tools: Use embedded tools to perform operations, from recovering modules to performing range tests.
- Get automatic updates: Automatically update the application itself, as well as the radio firmware library, without downloading any extra files.

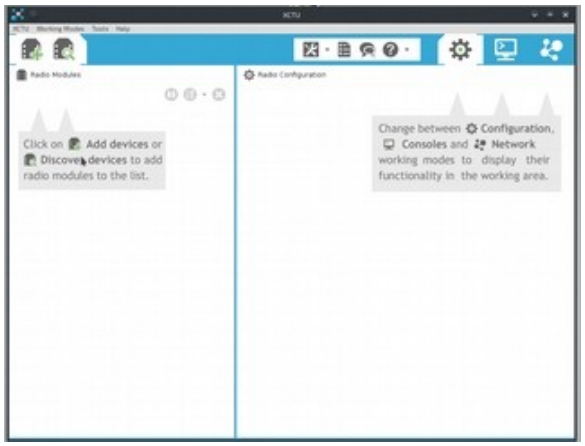
Regardless of the size of your device network, from just a few nodes to many, XCTU can simplify your device management tasks. Get started by downloading and installing XCTU, or navigate the topics on the left to learn how to use the tool.

There is a good tutorial provided by Lybellium[5] at http://www.libelium.com/development/waspmote/documentation/x-ctu-tutorial/ address.

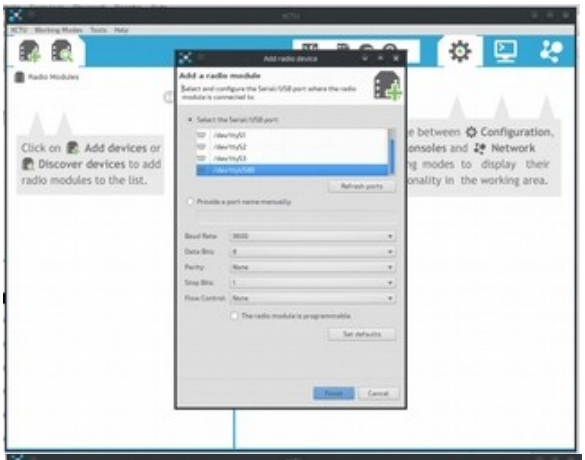
4 https://www.digi.com/resources/documentation/digidocs/90001458-13/default.htm
5 http://www.libelium.com/development/waspmote/documentation/x-ctu-tutorial/

Getting started with XBEE DIGIMESH and XCTU software

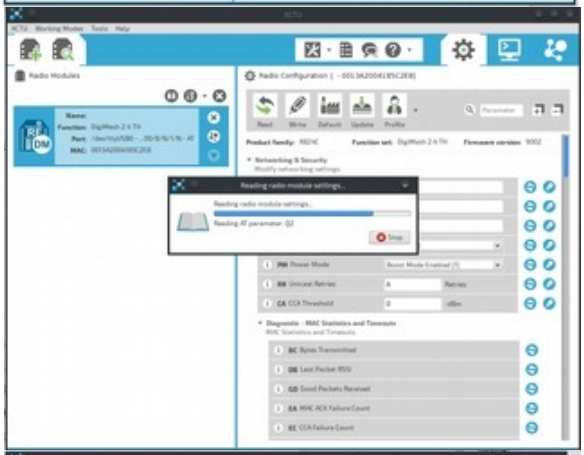
1. Download the software from DiGi site
2. Installing and start it
3. Connect via USB one Xbee Board with Xbee Module installed



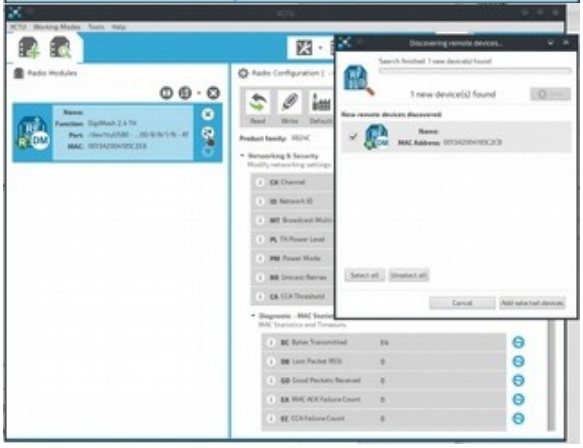
4. Add a new device by clicking on the tty port



5. Get all configuration about node



6. By clicking on the network button XCTU start to search the other neighbor devices, in our case are only one



Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

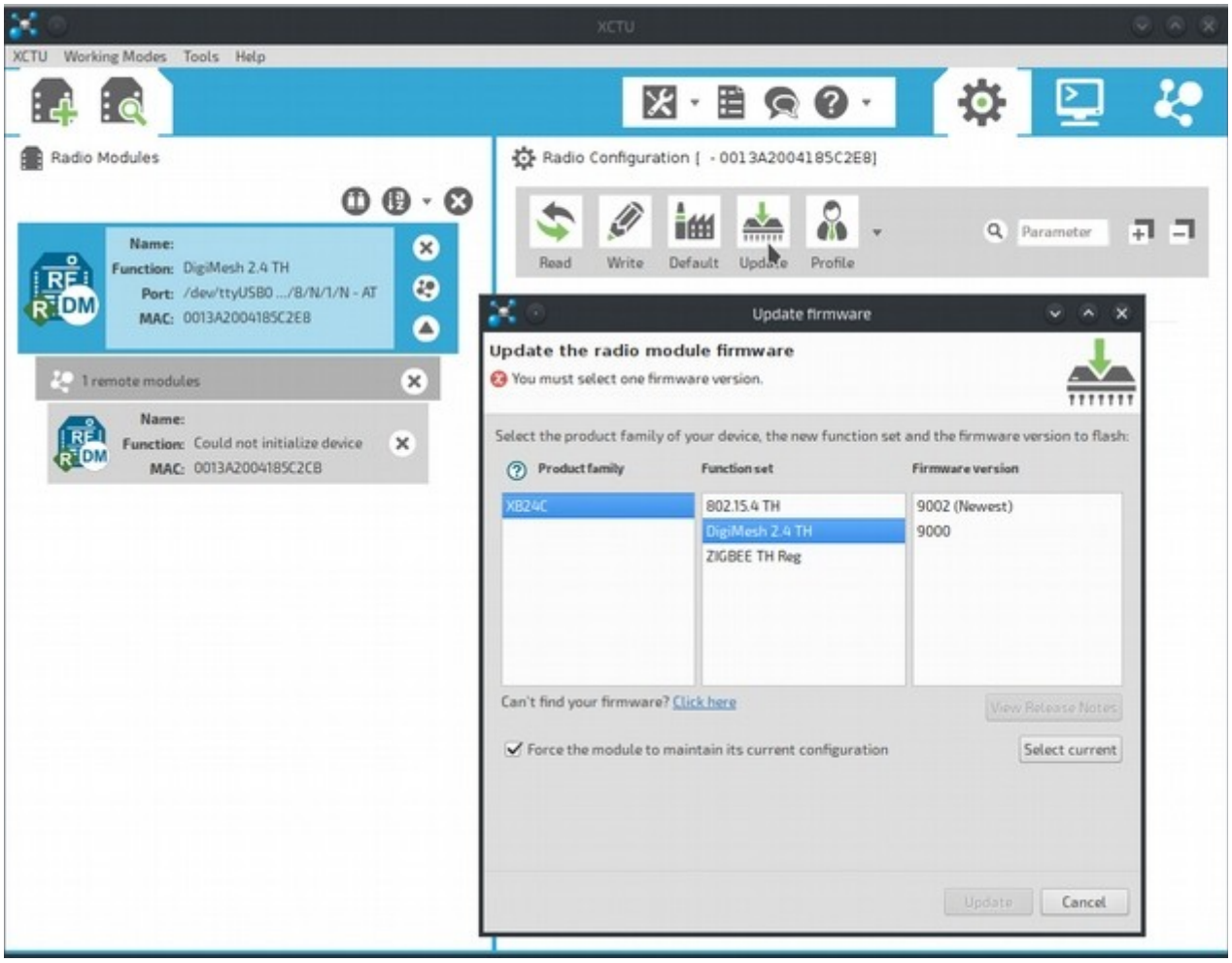
7. Select the device and add it

8. All tools are selectable by clicking on the tools button

The software manage the upgrade, on the air upgrading and all function that are needed for having a configured network.

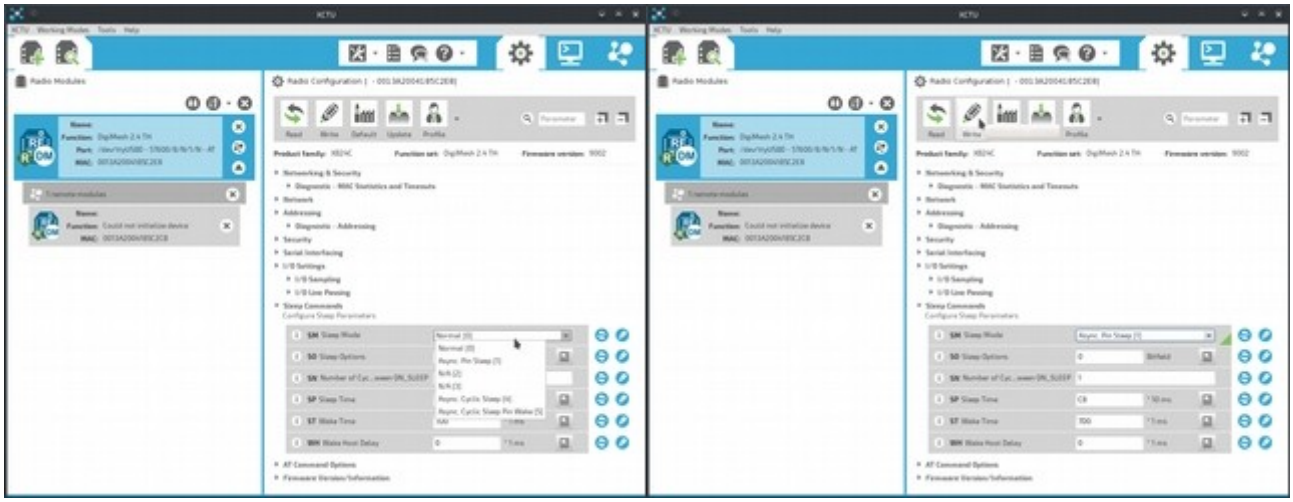
Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

Firmware Configuration

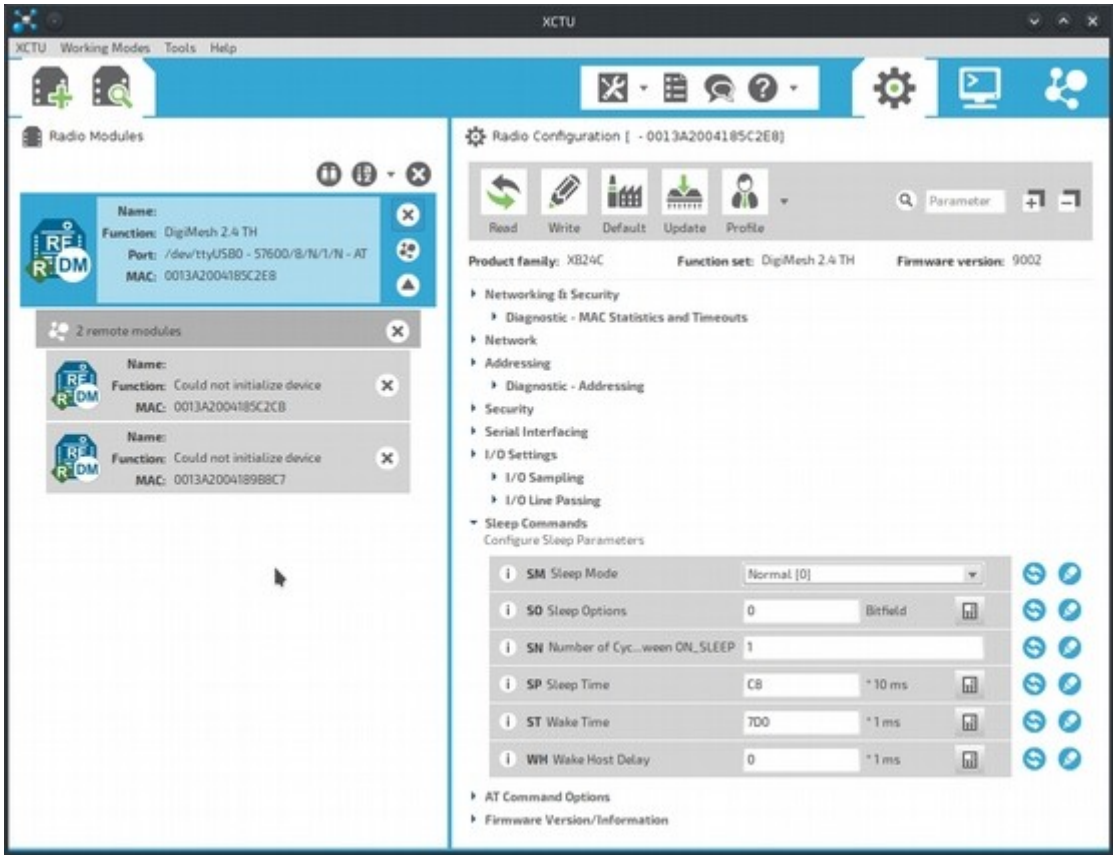


In the image above it is possible to see all options:

- Read
- Write
- Default
- Update
- Profile



For example, by selecting Update is it possible to see all firmware that can be upload on the connected radio, by Write options is possible upload the modified configuration.



In the last figure we can see all nodes in connected.

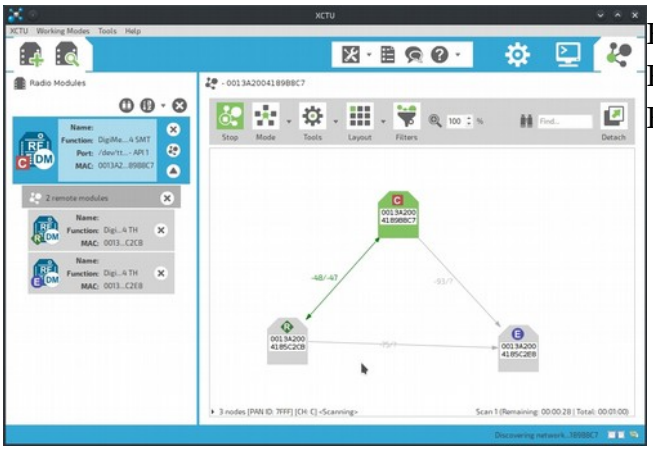
Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

XBEE Tests.

DIGIMESH network range test

Indoor

Test configuration:



End node at 30m indoor to the Coordinator, Router at 5m to the Coordinator and 30m to the End node.

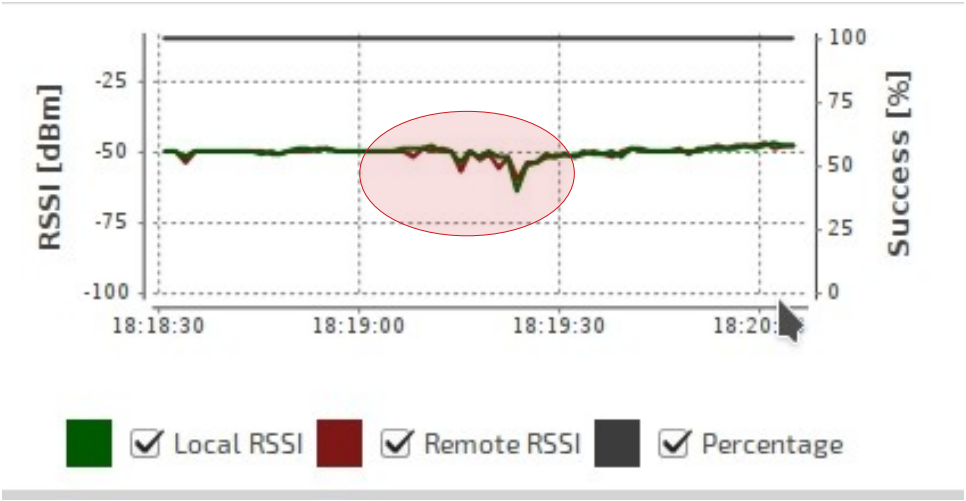
Once the range test process has started, XCTU represents the retrieved data in three ways:

- **RSSI Chart** represents the RSSI values of the local and remote devices during the range test session. The chart also contains the percentage of success for the total packets sent.
- **Local and Remote instant RSSI value** display the instant RSSI value of the local and remote devices. This value is retrieved for the last packet sent/received.
- **Packet summary** displays the total number of packets sent, packets received, transmission errors, and packets lost. It also displays the percentage of success sending and receiving packets during the range test session.

In the figure it is possible to see a situation with people that walk between the nodes: red area in the graphical view.

Results

Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001



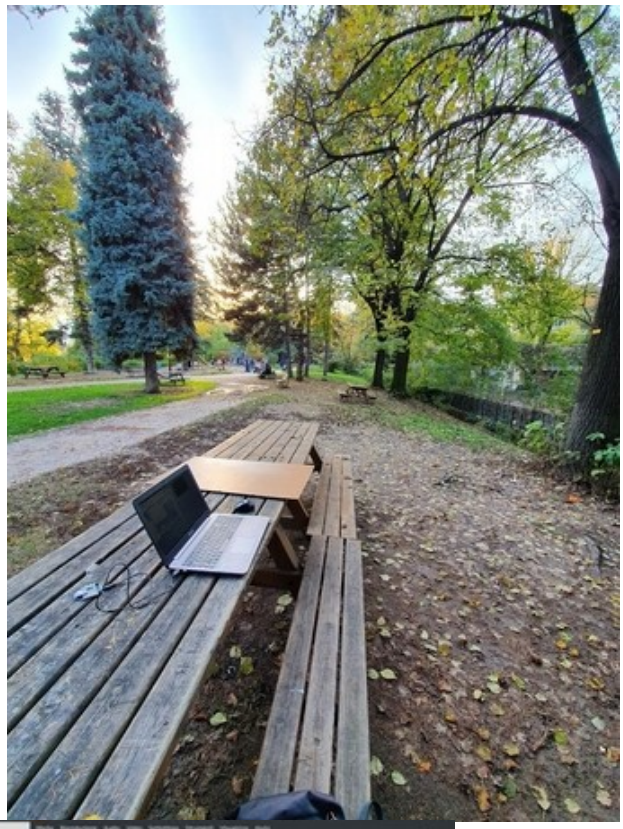
We verified that in the same configuration with a glasses wall between the Router – End-Node and Coordinator, the End-Node signal disappear, this because the glass is opaque for signals.

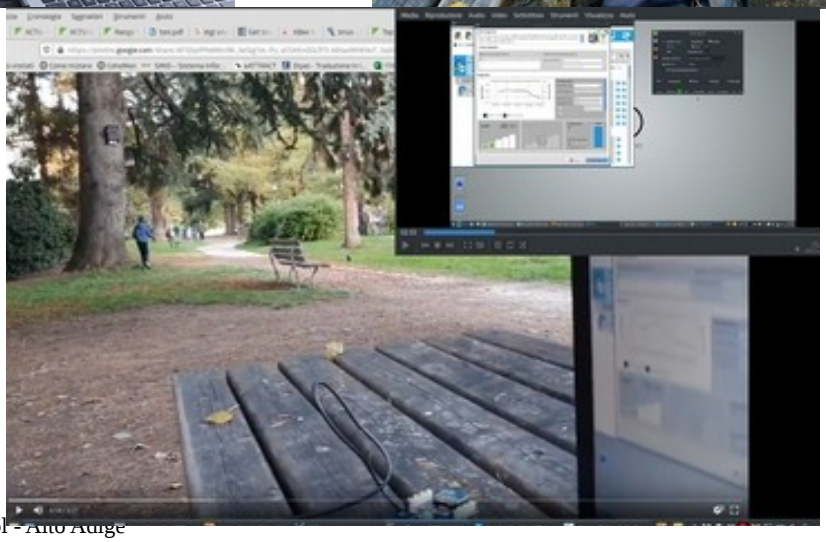
Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

Outdoor

1. First operation was configuring the devices: Coordinator and routers and setting the AT API for having a remote control of all devices in the network

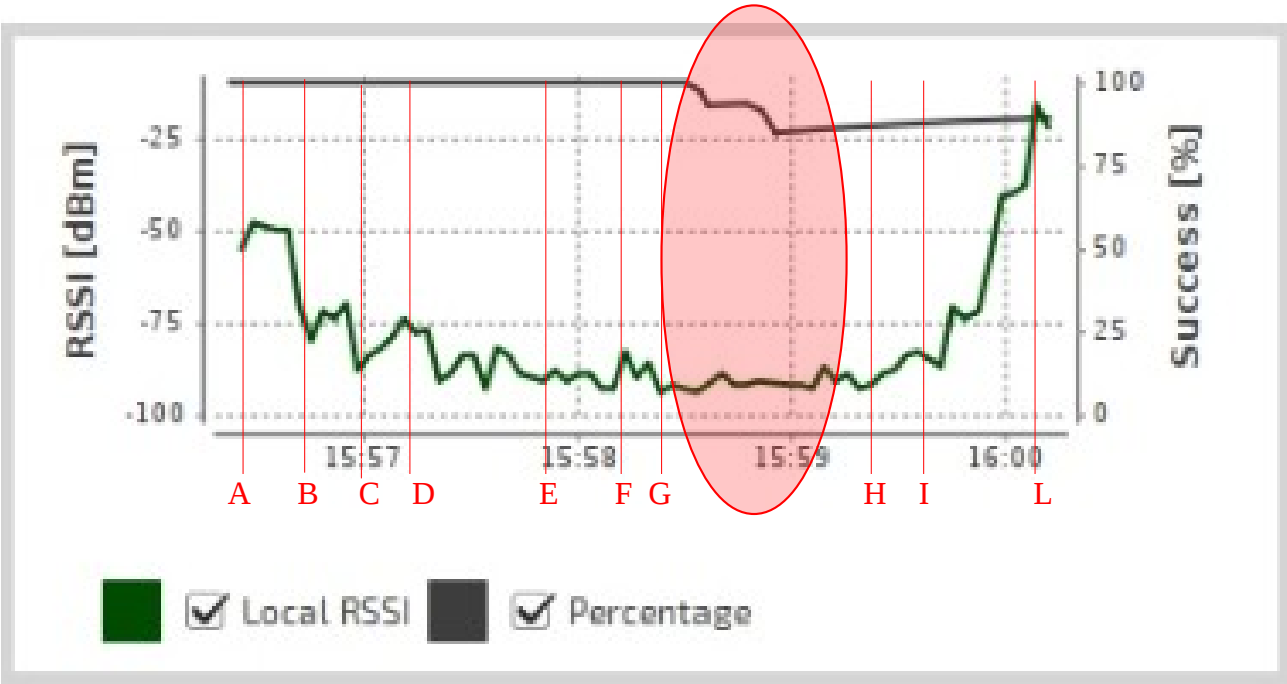
2. Starting the test and moving in a real outdoor environment (tree, statues, people, etc.)
    1. Opening the Radio Range test tool
    2. Move on the environment





Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

Results



Where:

A - starting point – Human body between the radios

B - Radios distance is 50m – A human body between the radios.

B-C positions - Radios distance is about 50-100m - Free see view between the radios

C - Radios distance about 100m – A big tree between the radios

D - Radios distance about 150m – Free see view between the radios

From E to H - Radios distance about 200m – Different radio positions

F – Radios distance 200m – Free see view between the radios

G-H - Radios distance about 200m

H-I-L - Fast return

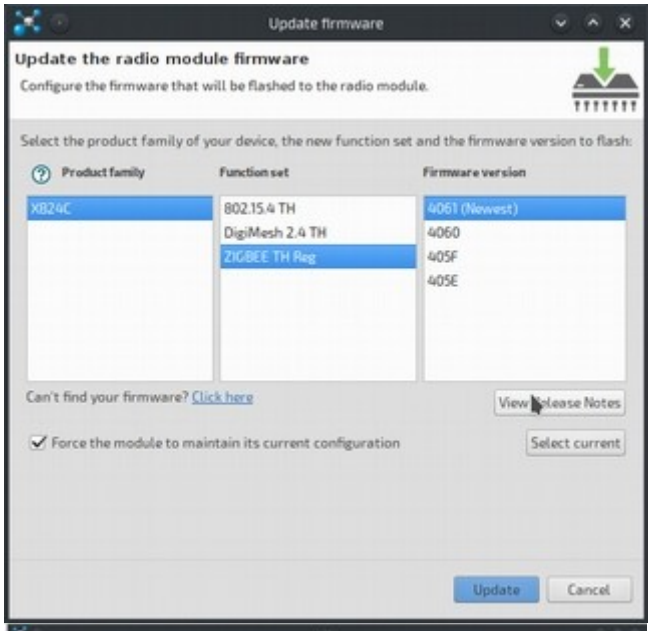
Zigbee Firmware configuration

XBEE XB24C modules in the kit are compatible with the other stack protocols: Zigbee TH Reg., DigiMesh and 802.15.4 TH

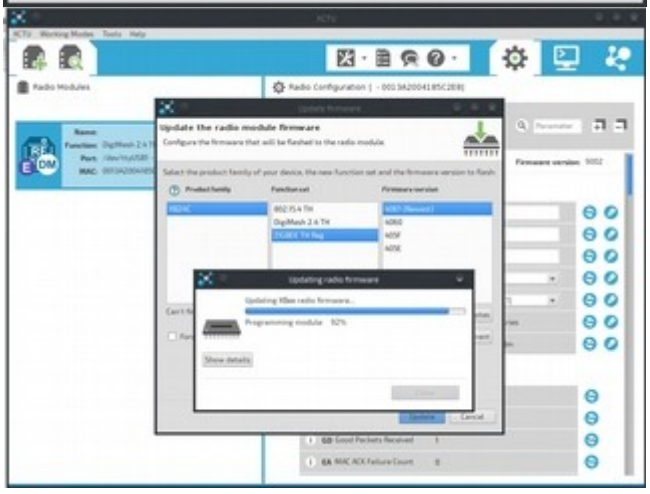
The XB24CSE as the Zigbee smart energy too.

Here the images that demonstrating the success of the operation

1. Firmware selection



2. Select and upload the Zigbee firmware



3. Upload done.

Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

Evaluation of software platforms

In this Phase we did the following evaluations for the software platforms:

- Check of a real Open Source Compliant License.
- Check the availability of all software components.
- Analysis of the specs needed.
- Analysis of the simplicity of the installation.
- Evaluation of compatibility with device protocols.
- Evaluation of compatibility with Databases.
- Test tools

The Software platforms that we considered are::

- SiteWhere
- DeviceHive
- ThingsBoard Community Edition
- WSo2 IoT platform
- Zetta

SiteWhere

License

Common Public Attribution License Version 1.0 (CPAL-1.0):

https://github.com/sitewhere/sitewhere/blob/master/LICENSE.txt

Software availability

All software sources are downloadable here:

https://github.com/sitewhere/sitewhere

Host System requirements

SiteWhere 2.1 uses a microservices architecture, the number of processes running concurrently has increased, which in turn requires more memory and processing power. The minimum hardware specifications for a single node Kubernetes cluster running a SiteWhere instance is:

Resource	Min Value
Memory	16GB RAM
CPU	4 CPUs
Hard Disk/SSD	100GB

In most real-world deployment scenarios, a multi-node Kubernetes cluster will be used so that the system can be made highly available. In cases where microservices are distributed across multiple k8s nodes, the per-node requirements can be adjusted based on the number of microservices per node (generally 500MB of memory per microservices).

Installation

SiteWhere 2.1 architecture is based on many microservices which are deployed into a Kubernetes infrastructure.

The installation for single-node is performed by Docker/Kubernetes Integration or Minicube.

The installation require a knowledge of Kubernetes infrastructure, Docking technologies and virtualization.

Following the instructions from https://sitewhere.io/docs/2.1.0/deployment/#system-requirements site, the installation can be performed step by step.

Sitewhere with HiveMQ brocker on Docker

Open the Docker client and type the following:

docker run -p 80:8080 -p 1883:1883 -p 61623:61623 sitewhere/standalone:1.8.0

Once all of the server components have started, navigate to the following URL:

http://your.docker.server.ip/sitewhere/admin/

This configuration makes SiteWhere available on port 80, so verify that you do not have another web server running on that port. To change the web server port change the port mapping from *-p 80:8080* to *-p {desired port}:8080*. The exposed ports are explained below:

Port 8080 - *SiteWhere Admin Console and REST services*

Port 1883 - *HiveMQ MQTT broker*

Port 61623 - *HiveMQ MQTT broker over web socket*

Supported protocols

The communication can be developed through different protocols like HTTP, WebSockets, direct sockets, MQTT and AMQP (Advanced Message Queuing Protocol).

Databases

- name: MongoDB

  repository: https://kubernetes-charts.storage.googleapis.com

  version: 5.2.0

- name: Cassandra

  repository: https://kubernetes-charts-incubator.storage.googleapis.com

  version: 0.10.1

- name: InfluxDB

  repository: https://kubernetes-charts.storage.googleapis.com

version: 1.4.1

- name: Warp10

repository: https://sitewhere.io/helm-charts

version: 0.1.4

- name: PostgreSQL

repository: https://kubernetes-charts.storage.googleapis.com

version: 6.5.0

Test tools

The test tools are downloadable here: https://github.com/sitewhere/sitewhere-load-test.

DeviceHive

License

Apache License, Version 2.0, January 2004, http://www.apache.org/licenses/:

https://github.com/devicehive/devicehive-java-server/blob/master/LICENSE

Software availability

All software source are downloadable here:

https://github.com/devicehive/

Host System requirements

The following requirements are for Docker-compose installation.

4 CPU cores

- 8 GB of RAM
- At least 16 GB of disk space for OS and Docker data (images, volumes, etc). On CentOS 7 it better to have additional 10 GB disk for Docker data.
- Linux distribution that fully support containers. Like CentOS 7, Fedora 24 and newer, Ubuntu 14.04 and later
- Docker version 1.13 and later
- Docker-compose version 1.12 and later

Installation was tested on machine with CentOS 7 distribution.

Installation

DeviceHive stack can runs with single instance of each component, then scale up by adding additional Front-end, Back-end, Kafka and ZooKeeper instances. And finally attach Apache Spark analytics to Apache Kafka.

The easiest way to try DeviceHive locally or in your development datacenter is to deploy it using Docker Compose.

This will start complete DeviceHive service stack running:

- DeviceHive Front-end
- DeviceHive Back-end
- Hazelcast IMDG
- Zookeeper
- Kafka
- PostreSQL
- Admin console

More details in the rdbms-image subdirectory.

https://github.com/devicehive/devicehive-docker

Supported protocols

DeviceHive provides REST, Websocket APIs by default + MQTT API as an additional plugin.
All communication is performed via JSON messages.

Databases

PostgreSQL 9.1 or above.

Test tools

The test tools are downloadable from here: https://github.com/devicehive

Some of interesting tools are:

- DeviceHive Python Device Simulator: a device activity simulating tool.
- Devicehive-tests: the integration tests

ThingsBoard Community Edition

License

Apache License, Version 2.0, January 2004, http://www.apache.org/licenses/

https://github.com/thingsboard/thingsboard/blob/master/LICENSE

Software availability

All software sources are downloadable here:

https://github.com/thingsboard/thingsboard

Host System requirements

Hardware requirements depend on chosen database and amount of devices connected to the system. To run ThingsBoard and PostgreSQL on a single machine you will need at least 1Gb of RAM. To run ThingsBoard and Cassandra on a single machine you will need at least 8Gb of RAM.

Installation

Starting ThingsBoard v2.2, platform was refactored to support microservices architecture, but also to be able to run the platform as a monolithic application in a standalone mode. Supporting both options requires some additional programming efforts, however, is critical due to back-ward compatibility with variety of existing installations.

ThingsBoard has installation instructions for standalone and cloud-based systems, with binary or by sources, here the options:

Installation in standalone mode:

Ubuntu Server: https://thingsboard.io/docs/user-guide/install/ubuntu/

- CentOS/RHEL Server: https://thingsboard.io/docs/user-guide/install/rhel/
- Windows: https://thingsboard.io/docs/user-guide/install/windows/
- Raspberry Pi 3: https://thingsboard.io/docs/user-guide/install/rpi/
- Docker (Windows): https://thingsboard.io/docs/user-guide/install/docker-windows/
- Docker (Linux or Mac OS): https://thingsboard.io/docs/user-guide/install/docker/

Installation on cloud-based mode:

- AWS: https://thingsboard.io/docs/user-guide/install/aws/
- Azure: https://thingsboard.io/docs/user-guide/install/azure/
- DigitalOcean: https://thingsboard.io/docs/user-guide/install/digital-ocean/
- GCP: https://thingsboard.io/docs/user-guide/install/gcp/
- IBM Cloud: https://thingsboard.io/docs/user-guide/install/ibm-cloud/
- Alibaba Cloud: https://thingsboard.io/docs/user-guide/install/alibaba-cloud/

We followed the Centos/RHEL Server installation with Hybrid database PostgreSQL + Cassandra database engines.

Supported protocols

ThingsBoard supports following protocols for device connectivity:

- MQTT
- CoAP
- HTTP

Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

Databases

ThingsBard uses database to store entities (devices, assets, customers, dashboards, etc) and telemetry data (attributes, timeseries sensor readings, statistics, events). Platform supports three database options at the moment:

- **SQL** - Stores all entities and telemetry in SQL database. ThingsBoard authors recommend to use PostgreSQL and this is the main SQL database that ThingsBoard supports. It is possible to use HSQLDB for local development purposes. We do not recommend to use HSQLDB for anything except running tests and launching dev instance that has minimum possible load.
- **NoSQL** - Stores all entities and telemetry in NoSQL database. ThingsBoard authors recommend to use Cassandra and this is the only NoSQL database that ThingsBoard supports at the moment. However, due to a lot of interest to deployments with managed databases, we plan to introduce support on AWS DynamoDB in v2.3.
- **Hybrid** - Stores all entities in SQL database and all telemetry in NoSQL database.

The supported databases are:

PostgreSQL for SQL DB and PostgreSQL+Cassandra for Hybrid solution.

Test tools

ThingsBoard has created a Gatling MQTT plugin and its performance test. Can be followed at the link: https://thingsboard.io/docs/reference/performance-tools/

WSo2 IoT

License

Apache License, Version 2.0, January 2004, http://www.apache.org/licenses/:

https://github.com/wso2/product-is/commit/de44cda12aaec9d348963fcf0c7d4f943938d51d

Software availability

All software sources are downloadable here:

https://github.com/wso2/product-is

Host System requirements

- Minimum memory - 2 GB
  - ~ 2 GB minimum
  - ~ 512 MB heap size. This is generally sufficient on DEV/QA with a low number of users.
  - ~ 1 GB heap size for PROD environment, supporting the most common use cases (OAuth/SAML, Federation, etc.) with moderate traffic.

- Processor - 2 Core/vCPU 1.1GHz or higher
- Disk - ~ 1 GB, excluding space allocated for log files and databases.
- Java SE Development Kit 1.8
- The Management Console requires full Javascript enablement of the Web browser.
- To build WSO2 Identity Server from the Source distribution, it is also necessary that you have Maven 3 or later

Installation

The installation can be performed by their Identity Server: https://wso2.com/identity-and-access-management/#. For Downloading the installation files have to leave an email, the options are:

Cloud	Standalone
• AWS Cloud Formation • Kubernetes • Docker • Docker Compose • Vagrant • Helm	• CentOS • Windows X64 • Ubuntu • MacOs • Apt • YUM • Puppet • Brew • Ansible • Binary • Source

About the source compilation, not always succeded:

```
 org.wso2.identity.integration.test.rest.api.user.approval.v1.UserMeApprovalTest.testListTasksWhenAvailable
 Error Details

1 expectation failed.
JSON path findAll{ it.presentationName == 'TestWorkflowAddUserForRestTask' }.size() doesn't match.
Expected: is <3>
  Actual: 4

 Stack Trace

java.lang.AssertionError:
1 expectation failed.
JSON path findAll{ it.presentationName == 'TestWorkflowAddUserForRestTask' }.size() doesn't match.
Expected: is <3>
  Actual: 4

         at
org.wso2.identity.integration.test.rest.api.user.approval.v1.UserMeApprovalTest.testListTasksWhenAvailable(UserMeApprovalTest.java:14
0)

 Standard Output

INFO  [org.wso2.carbon.automation.engine.testlisteners.TestManagerListener] - ================== On test success
org.wso2.identity.integration.test.rest.api.user.approval.v1.UserMeApprovalTest.testListTasksWhenEmpty ==================
INFO  [org.wso2.carbon.automation.engine.testlisteners.TestManagerListener] - ================== Running the test method
org.wso2.identity.integration.test.rest.api.user.approval.v1.UserMeApprovalTest.testListTasksWhenAvailable ==================
INFO  [org.wso2.identity.integration.test.rest.api.user.approval.common.UserApprovalTestBase] - Adding users matching the workflow
engagement TestWorkflowAddUserForRest to tenant carbon.super
WARN  [org.apache.commons.httpclient.HttpMethodBase] - Going to buffer response body of large or unknown size. Using
getResponseBodyAsStream instead is recommended.
WARN  [org.apache.commons.httpclient.HttpMethodBase] - Going to buffer response body of large or unknown size. Using
getResponseBodyAsStream instead is recommended.
WARN  [org.apache.commons.httpclient.HttpMethodBase] - Going to buffer response body of large or unknown size. Using
getResponseBodyAsStream instead is recommended.
INFO  [org.wso2.identity.integration.test.rest.api.user.approval.v1.UserMeApprovalTest] - Waiting 5 seconds till the workflow is
applied for association, iteration 1 of 3
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] - [2019-11-28 19:35:40,186] [30aa3f38-4e05-4826-b355-
be7df10f09dd] ERROR {org.apache.ode.bpel.engine.BpelEngineImpl} - Scheduled job failed; jobDetail=JobDetails( instanceId: null mexId:
```

```
hqejbhcnphresx2ndwe6g0 processId: {http://bpel.mgt.workflow.carbon.wso2.org/approvalProcess}TestWorkflowAddUserForRest-1 type:
INVOKE_INTERNAL channel: null correlatorId: null correlationKeySet: null retryCount: null inMem: false detailsExt: {enqueue=false})
java.lang.NullPointerException: Null operation
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.wso2.carbon.bpel.b4p.utils.SOAPHelper.createSoapRequest(SOAPHelper.java:220)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.wso2.carbon.bpel.b4p.utils.SOAPHelper.createSoapRequest(SOAPHelper.java:152)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.wso2.carbon.bpel.b4p.extension.PeopleActivity.invoke(PeopleActivity.java:649)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.wso2.carbon.bpel.b4p.extension.BPEL4PeopleExtensionOperation.runAsync(BPEL4PeopleExtensionOperation.java:66)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.runtime.extension.AbstractLongRunningExtensionOperation.run(AbstractLongRunningExtensionOperation.java:95)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
sun.reflect.GeneratedMethodAccessor654.invoke(Unknown Source)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at java.lang.reflect.Method.invoke(Method.java:498)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.jacob.vpu.JacobVPU$JacobThreadImpl.run(JacobVPU.java:451)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.jacob.vpu.JacobVPU.execute(JacobVPU.java:139)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelRuntimeContextImpl.execute(BpelRuntimeContextImpl.java:1002)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.PartnerLinkMyRoleImpl.invokeNewInstance(PartnerLinkMyRoleImpl.java:208)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelProcess$1.invoke(BpelProcess.java:283)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelProcess.invokeProcess(BpelProcess.java:224)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelProcess.invokeProcess(BpelProcess.java:279)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelProcess.handleJobDetails(BpelProcess.java:434)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelEngineImpl.onScheduledJob(BpelEngineImpl.java:558)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelServerImpl.onScheduledJob(BpelServerImpl.java:467)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler$RunJob$1.call(SimpleScheduler.java:687)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler$RunJob$1.call(SimpleScheduler.java:681)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler.execTransaction(SimpleScheduler.java:319)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler.execTransaction(SimpleScheduler.java:274)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler$RunJob.call(SimpleScheduler.java:681)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler$RunJob.call(SimpleScheduler.java:665)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
java.util.concurrent.FutureTask.run(FutureTask.java:266)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at java.lang.Thread.run(Thread.java:748)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] - [2019-11-28 19:35:40,432] [306b4167-6323-49c6-8fa4-
445de4bf698a] ERROR {org.apache.ode.bpel.engine.BpelEngineImpl} - Scheduled job failed; jobDetail=JobDetails( instanceId: null mexId:
hqejbhcnphresx2ndwe6g3 processId: {http://bpel.mgt.workflow.carbon.wso2.org/approvalProcess}TestWorkflowAddUserForRest-1 type:
INVOKE_INTERNAL channel: null correlatorId: null correlationKeySet: null retryCount: null inMem: false detailsExt: {enqueue=false})
java.lang.NullPointerException
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelRuntimeContextImpl.registerForExtensionNotification(BpelRuntimeContextImpl.java:1774)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.runtime.extension.AbstractLongRunningExtensionOperation.registerForNotification(AbstractLongRunningExtensionOpera
tion.java:77)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.runtime.extension.AbstractLongRunningExtensionOperation$WAITING.<init>(AbstractLongRunningExtensionOperation.java
:115)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.runtime.extension.AbstractLongRunningExtensionOperation$WAITING.<init>(AbstractLongRunningExtensionOperation.java
:104)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.runtime.extension.AbstractLongRunningExtensionOperation.run(AbstractLongRunningExtensionOperation.java:101)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
sun.reflect.GeneratedMethodAccessor654.invoke(Unknown Source)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at java.lang.reflect.Method.invoke(Method.java:498)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.jacob.vpu.JacobVPU$JacobThreadImpl.run(JacobVPU.java:451)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.jacob.vpu.JacobVPU.execute(JacobVPU.java:139)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelRuntimeContextImpl.execute(BpelRuntimeContextImpl.java:1002)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.PartnerLinkMyRoleImpl.invokeNewInstance(PartnerLinkMyRoleImpl.java:208)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelProcess$1.invoke(BpelProcess.java:283)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelProcess.invokeProcess(BpelProcess.java:224)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelProcess.invokeProcess(BpelProcess.java:279)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelProcess.handleJobDetails(BpelProcess.java:434)
```

```
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelEngineImpl.onScheduledJob(BpelEngineImpl.java:558)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.bpel.engine.BpelServerImpl.onScheduledJob(BpelServerImpl.java:467)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler$RunJob$1.call(SimpleScheduler.java:687)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler$RunJob$1.call(SimpleScheduler.java:681)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler.execTransaction(SimpleScheduler.java:319)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler.execTransaction(SimpleScheduler.java:274)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler$RunJob.call(SimpleScheduler.java:681)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
org.apache.ode.scheduler.simple.SimpleScheduler$RunJob.call(SimpleScheduler.java:665)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
java.util.concurrent.FutureTask.run(FutureTask.java:266)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -  at java.lang.Thread.run(Thread.java:748)
INFO  [org.wso2.carbon.automation.extensions.servers.utils.ServerLogReader] -
INFO  [org.wso2.identity.integration.test.rest.api.user.approval.v1.UserMeApprovalTest] - Waiting 5 seconds till the workflow is
applied for association, iteration 2 of 3
INFO  [org.wso2.identity.integration.test.rest.api.user.approval.v1.UserMeApprovalTest] - Waiting 5 seconds till the workflow is
applied for association, iteration 3 of 3
Request method:     GET
Request URI:        https://localhost:9853/t/carbon.super/api/users/v1/me/approval-tasks
Proxy:                      <none>
Request params:     <none>
Query params:       <none>
Form params:        <none>
Path params:        <none>
Headers:            Authorization=Basic YWRtaW46YWRtaW4=
                                    Accept=application/json
                                    Content-Type=application/json; charset=UTF-8
Cookies:            <none>
Multiparts:                 <none>
Body:                       <none>

HTTP/1.1 200
Date: Thu, 28 Nov 2019 19:35:55 GMT
Content-Type: application/json
Transfer-Encoding: chunked
Server: WSO2 Carbon Server

[
    {
        "id": "454",
        "name": "{http://ht.bpel.mgt.workflow.identity.carbon.wso2.org}TestWorkflowAddUserForRestTask",
        "presentationSubject": "sample approval task",
        "presentationName": "TestWorkflowAddUserForRestTask",
        "taskType": "TASK",
        "status": "READY",
        "priority": 0,
        "createdTimeInMillis": "1574969741529"
    },
    {
        "id": "453",
        "name": "{http://ht.bpel.mgt.workflow.identity.carbon.wso2.org}TestWorkflowAddUserForRestTask",
        "presentationSubject": "sample approval task",
        "presentationName": "TestWorkflowAddUserForRestTask",
        "taskType": "TASK",
        "status": "READY",
        "priority": 0,
        "createdTimeInMillis": "1574969741335"
    },
    {
        "id": "451",
        "name": "{http://ht.bpel.mgt.workflow.identity.carbon.wso2.org}TestWorkflowAddUserForRestTask",
        "presentationSubject": "sample approval task",
        "presentationName": "TestWorkflowAddUserForRestTask",
        "taskType": "TASK",
        "status": "READY",
        "priority": 0,
        "createdTimeInMillis": "1574969740310"
    },
    {
        "id": "452",
        "name": "{http://ht.bpel.mgt.workflow.identity.carbon.wso2.org}TestWorkflowAddUserForRestTask",
        "presentationSubject": "sample approval task",
        "presentationName": "TestWorkflowAddUserForRestTask",
        "taskType": "TASK",
        "status": "READY",
        "priority": 0,
        "createdTimeInMillis": "1574969740310"
    }
]
```

Protocols

The platform includes a variety of connectors across many categories such as payments, CRM, ERP, social networks and legacy systems.

It also supports HTTP, HTTPS, WebSocket, gRPC, POP, IMAP, SMTP and many more transports. Formats and protocols: JSON, XML, SOAP and more.

Ships with adapters to COTS systems, such as SAP BAPI and IDoc, IBM WebSphere MQ, Oracle AQ, and MSMQ.

By the connectors can be encapsulated third-party API calls. There are three different type of connectors:

- SOAP-based connectors
- REST-based connectors
- Java API-based connectors

Databases

The database engine is RDBMS, the default database engine is Mysql server but you can set up the following database types for the API-M-specific databases:

- MySQL database
- MS SQL database
- Oracle database
- IBM DB2 database
- PostgreSQL database

At this link https://docs.wso2.com/display/AM210/Changing+the+Default+API-M+Databases, you can find the configuration metods.

Test tools

WSO2 team has developed a WSO2 Test Automation Framework (TAF): is an automation framework that performs equally in all stages of the deployment lifecycle.

It performs:

- Integration tests - Verify the product at the build level. Tests get executed on user mode/tenant module in standalone product execution mode.
- Platform tests - Verify the scenarios that integrate multiple WSO2 products or different platforms.

TAF simplifies the process by providing the following functionality:

- Simple automation context API
- Platform wide test execution support

- Annotation based test execution management support
- Test reports and coverage report generation
- Third party test tooling integration support (Jmeter)
- Selenium webdriver integration and ability to run UI tests to cross browser environments
- Framework extensibility capability through pluggable modules
- Configured admin services as test-oriented API
- Easy retrieval of environment variables and third party configurations

The sources are available at this link:

https://svn.wso2.org/repos/wso2/carbon/platform/trunk/platform-integration/

Zetta

License

MIT License:

https://github.com/zettajs/zetta

Software availability

All software sources are downloadable here:

https://github.com/zettajs/zetta

Host System requirements

The Zetta server is the highest level of abstraction in Zetta. The Zetta server contains Drivers, Scouts, Apps, and Server Extensions. A Zetta server will typically run on a hardware hub such as a BeagleBone Black, Intel Edison, or Raspberry Pi. The server itself coordinates interactions between all of the contained components to communicate with devices, and generate HTTP APIs that an API consumer can interact with.

Installation

Zetta is an open source platform built on Node.js for creating Internet of Things servers that run across geo-distributed computers and the cloud. Zetta combines REST APIs, WebSockets and reactive programming – perfect for assembling many devices into data-intensive, real-time applications.

The typical Zetta deployment.

1. One Zetta server runs on a hardware hub. This hub is typically something like a BeagleBone Black, Intel Edison, or Raspberry Pi.

- The Zetta hub connects to devices. Zetta mediates from HTTP to the particular protocols used in a deployment.

2. Another Zetta server runs in the Cloud. This server uses the exact same Node.js packages as Zetta running on the hub.
    - The Zetta server on the hardware hub connects to the server in the Cloud.
3. Zetta then exposes an API at the Cloud endpoint for developers to consume.

Zetta has the same code in the cloud and on the device, it is a peer to peer system. So, the Zetta servers are the same wherever they run.

The classic Zetta installation command is: npm install zetta.

Supported protocols

Zetta processes establish a secure link between each other device that use Z2Z protocol.

In Zetta FAQ page is reported that :

Protocol agnostic - Zetta can support almost all device protocols, and mediate them to HTTP.

Databases

The persistent layer is called Registry.

The Zetta Registry is a persistence layer for Zetta. It's a small database that lives in the server context, and holds information about devices connected to the server itself.

Test tools

From the https://github.com/zettajs you can find all example test files that you need.

Development kits conclusions table

	SiteWhere	Device Hive	ThigsBoard	WSo2 IoT	Zetta
License	Apache 1	Apache 2	Apache 2	Apache 2	MIT
Software repos	https://github.com/sitewhere/sitewhere	https://github.com/devicehive/	https://github.com/thingsboard/thingsboard	https://github.com/wso2/product-is	https://github.com/zettajs/zetta
HW Reaquirements	Memory 16GB RAM CPU 4 CPUs Hard Disk/SSD 100GB	8 GB of RAM At least 16 GB of disk space for OS and Docker data (images, volumes, etc). On CentOS 7 it better to have additional 10 GB disk for Docker data.	Minimum memory – 1 GB	Minimum memory - 2 GB Processor - 2 Core/vCPU 1.1GHz or higher Disk - ~ 1 GB, excluding space allocated for log files and databases.	Zetta server will typically run on a hardware hub such as a BeagleBone Black, Intel Edison, or Raspberry Pi
Installation	Intermediate	Intermediate	Yes	Intermediate	Yes
Supported protocols	HTTP, WebSockets, direct sockets, MQTT and AMQP	All communication is performed via JSON messages.. REST, Websocket APIs by default + MQTT API as an additional plugin	• MQTT • CoAP • HTTP	A lot by connectors: • SOAP-based connectors • REST-based connectors • Java API-based connectors	Z2Z, HTTP
Databases	Mongodb Cassandra Influxdb Warp10 Postgresql	PostgreSQL 9.1 or above.	PostgreSQL / +Cassandra	• MySQL database • MS SQL database • Oracle database • IBM DB2 database • PostgreSQL DB	ZETTA Registy
Test tools	Yes	Yes	Yes	Yes	NA

SOB benchmark tests

We choose as benchmark tests the ones provided by Phoronix Test Suite[6], a free open-source benchmark platform available for many configurations and operating systems. At first, we used a single board to analyze what could be the best tests to benchmark the board. After discarding the too heavy ones for the boards, the ones with difficult installation of dependencies needed and the network ones (since every board has its way to connect, from Ethernet to gsm to WiFi to WiFi only on a specific hot-spot), final benchmark tests selected are:

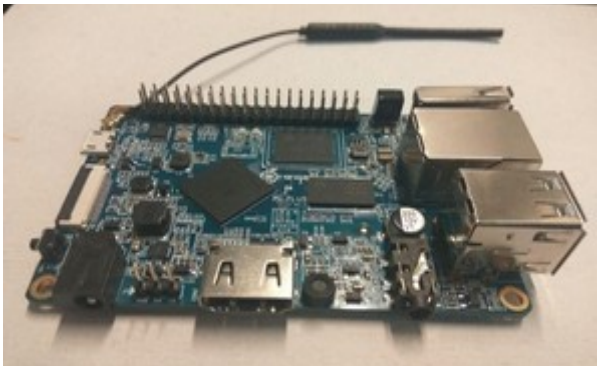
- John the ripper (https://openbenchmarking.org/test/pts/john-the-ripper), a password cracking test useful to benchmark CPU performance.
- Cachebench (https://openbenchmarking.org/test/pts/cachebench), specifically designed to test the cache bandwidth performance.
- C-Ray (https://openbenchmarking.org/test/pts/c-ray), a raytracer designed to test the floating-point CPU performance.
- RAMspeed (https://openbenchmarking.org/test/pts/ramspeed), the benchmark name speaks for itself

The devices selected are:

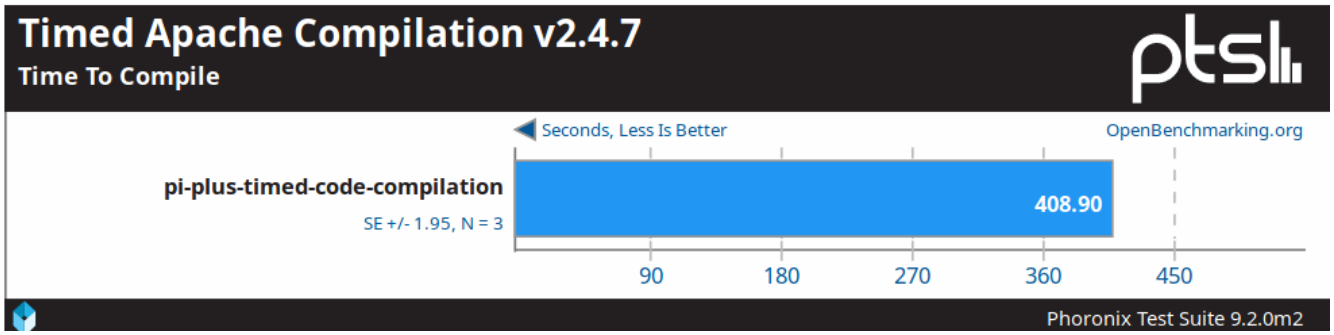
6 https://www.phoronix-test-suite.com/

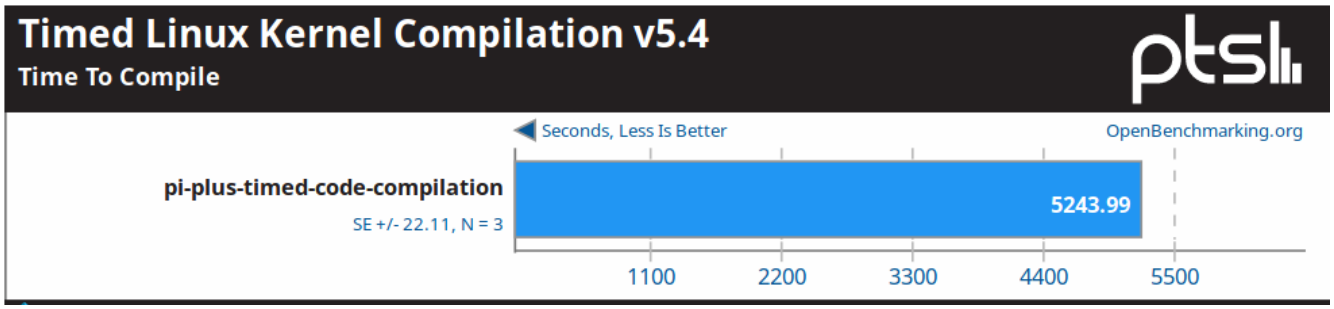
Orange Pi Plus by Xunlong

ARMv7 Cortex-A7 @ 1.30GHz (4 Cores)	Processor
sun8i	Motherboard
1024MB	Memory
16GB N/A + 8GB 8GTF4R	Disk
Ubuntu 16.04	OS
3.4.113+ (armv7l)	Kernel
LXDE 0.8.2	Desktop
X Server 1.18.4	Display Server
GCC 5.4.0 20160609	Compiler
ext4	File-System

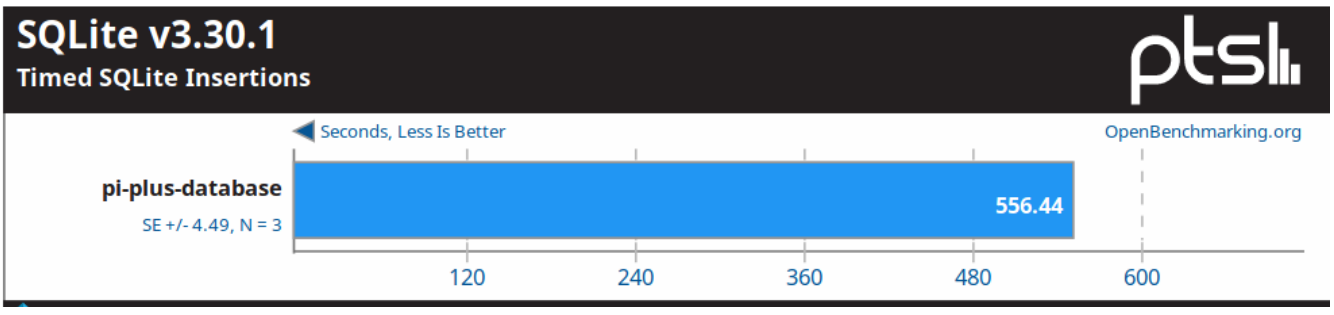


This is the board selected for the first benchmark tests in order to check time needed and compatibility, as it is the most powerful board of the manufacturer which we have more different model. At the beginning we tried the compilation profile (pts/compilation) in order to get an overview of the board but, out of various tests of 6 different benchmarks, only 2 provided a result, due to dependencies and/or execution errors:

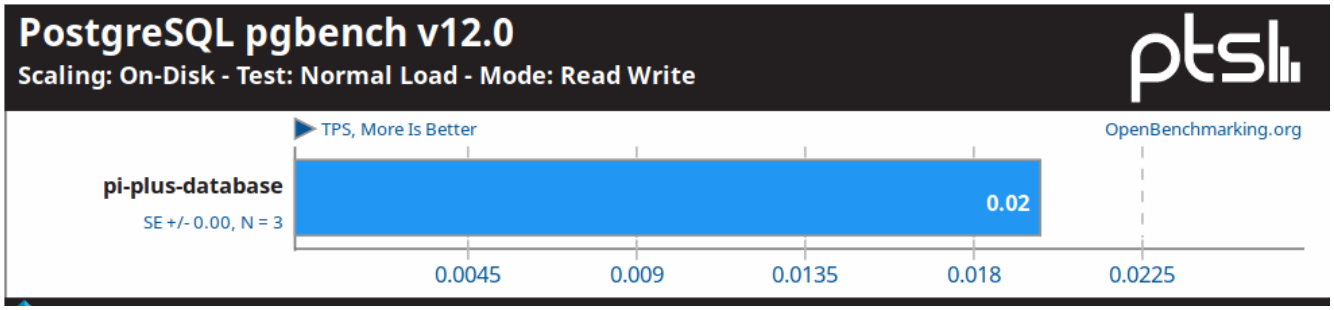


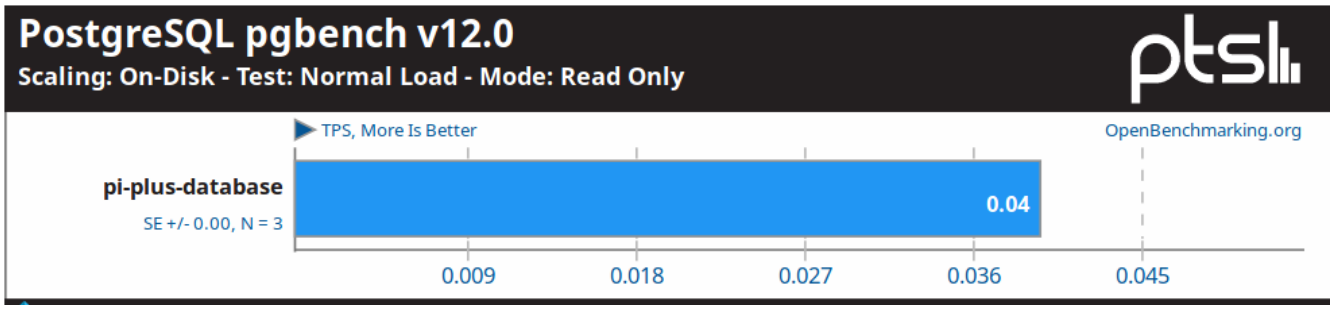


After discarding this Phoronix profile, the database one has been tried with same result (3 tests completed out of 22 scheduled and many tests running for hours without giving a meaningful result):



Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001





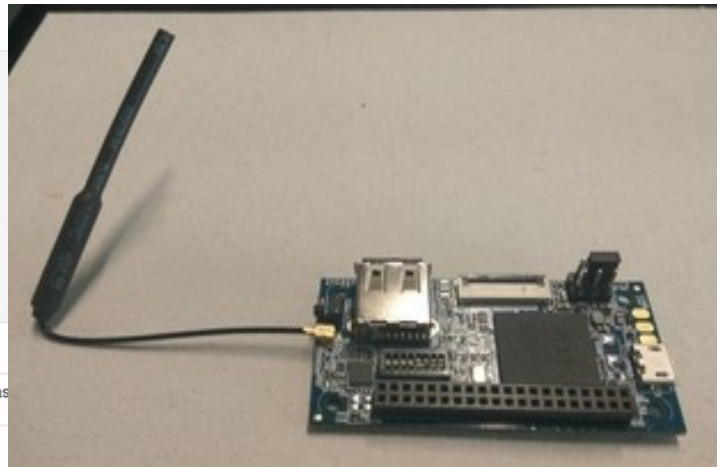
Based on this results, we made an analysis of simpler but still meaningful tests that would reduce compatibility problem. After finalizing the list, we started to try them on this board but it didn't work correctly anymore, even after trying different officials OS images on different SD cards, probably due to the extensive use.

Notes on Orange Pi PC Plus:

- The board goes in stand-by mode after some time. On wake-up, there is no way to put the password in the input box prompted. This is a known bug but the solution proposed in FAQ (http://www.orangepi.org/Docs/FAQ.html) didn't work. The workaround found was to use the "other" tab and log in there with the same credentials (username orangepi, password orangepi).
- Debian images boots correctly but there is no way to use browser for check Phoronix results. The already installed Firefox and the Chromium installed from APT both raise input/output errors trying to launch them.

Orange Pi i96 by Xunlong

CPU	ARM Cortex-A5 32bit
GPU	Separate graphic processor, Vivante's GC860 support OpenGLES1.1/2.0 support OpenVG1.4 support DirectFB support GDI/DirecShow 30M Triangle/s, 250M Pixel/s
Memory (SDRAM)	Integrated 256MB LPDDR2 SDRAM
Onboard Storage	TF card / Integrated 512MB 8Bit 1.8V 4K SLC Nand Flas
Onboard WIFI+BT	RDA5991, WIFI+BT



Since Orange Pi i96 has no display connector, the way you connect to the board is described in the manual: the Debian image provided by Orange automatically connects to any WiFi with SSID “orangepi” and password “orangepi” (tested with WPA2-Personal protocol). Once connected to the network, the board has an SSH port to connect to it, it is sufficient to find the IP address of the board (for example, using fping in the Linux environment providing the WiFi network).

There wasn’t possible to test Phoronix tests as the RAM available (256M) is not sufficient to run the software, even creating a 1GB swap partition. Since it is the only one board where the suite can’t run, we didn’t evaluate this further.

Notes on Orange Pi i96:

- There was also an option to use microUSB port for serial connection. It wasn’t possible to use OTG feature as the connector is used as power supply and the board requires at least a 2A electric current, while a pc USB2.0 port provides 0,5A and a USB3.0 port provides 0,9A. We didn’t test a USB3.1 port, which as specification provides 2A.
- We tested changes in firmware to try different connection methods, for example direct connection to the available WiFi network without creating a specific OrangePi hot-spot, discovering that shutting down the board cutting off the power supply has an high chance to damage the SD integrity. During these tests, 2 SD needed complete reformatting and 1 SD was not usable anymore.
- To check the OS functionality, it’s sufficient to check LED luminosity: a bright green light means that the OS is running, a dim light that the board it’s connected to power supply but OS isn’t working (it happened when SD got broken and the board was connected to a power supply with less than 2A.

Orange Pi 4G-IoT by Xunlong

Processor	MT6737
CPU	Quad core ARM® Cortex-A53, Main frequency up to 1.25
GPU	• ARM Mali-T720 MP1
Memory	1GB DDR3
Emmc	8GB EMMC Flash
Wireless	WIFI / BT / FM / GPS Four in one

It wasn't possible to communicate with this device. The operating systems provided by manufacturer require a display for first settings. Using the guide provided and the SP Flash Tool v5.1916 (Linux) suggested, all version of Android (6.0, 8.1 and the one with HDMI support) don't be recognized as a working device by any monitor and television used. Modifying the eMMC internal memory as suggested by guide to boot an SD card with Linux, the board has been recognized but its refresh rate frequency is not supported by any monitor we tested.

We also tried to replicate the code for the Orange Pi i96 automatic hot-spot connection but the 4G-IoT didn't connect to it.

Searching online (Orange provide an official community forum), the board uses a 48Hz refresh rate not supported by many monitors and the only solutions found would be to buy a specified type of display or rewrite the specific kernel module by our-self, as no one in the community already did it.

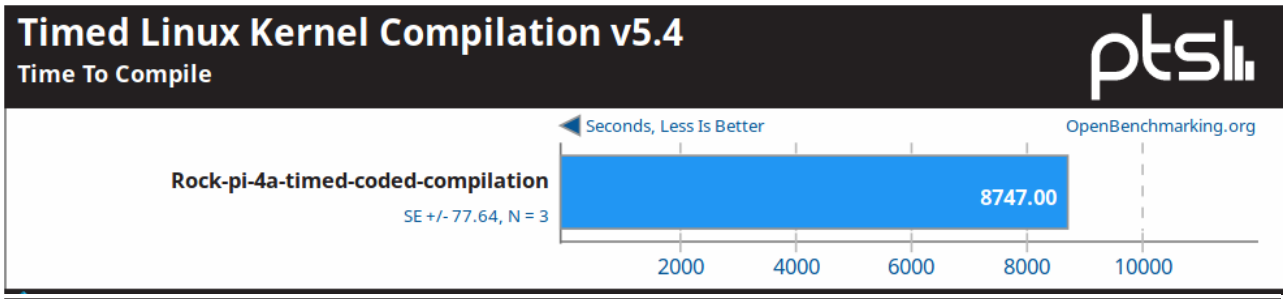
Since the main goal of this work is testing and evaluating a SOB that could be easily used for IoT application without relying on any additional limited hardware/software, we didn't continue the analysis on this board.

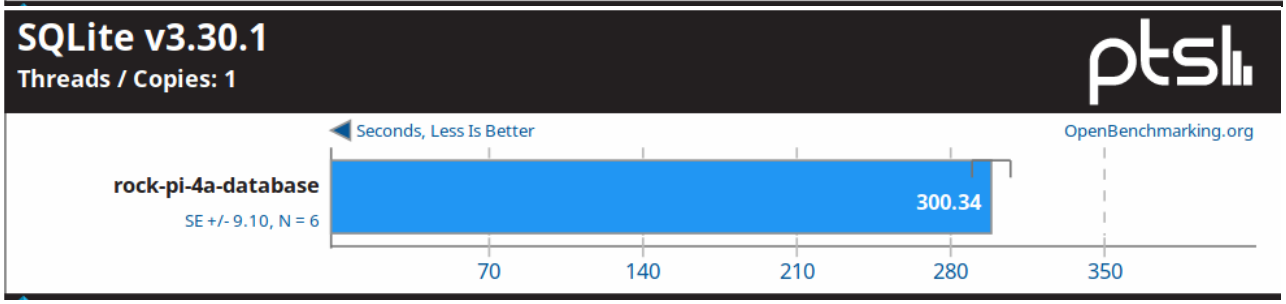
For the one which would like to continue the evaluation of the board, Orange provided a 5,5inch display for around 20$ plus shipping costs: http://www.orangepi.org/orangepibbsen/forum.php?mod=viewthread&tid=3716

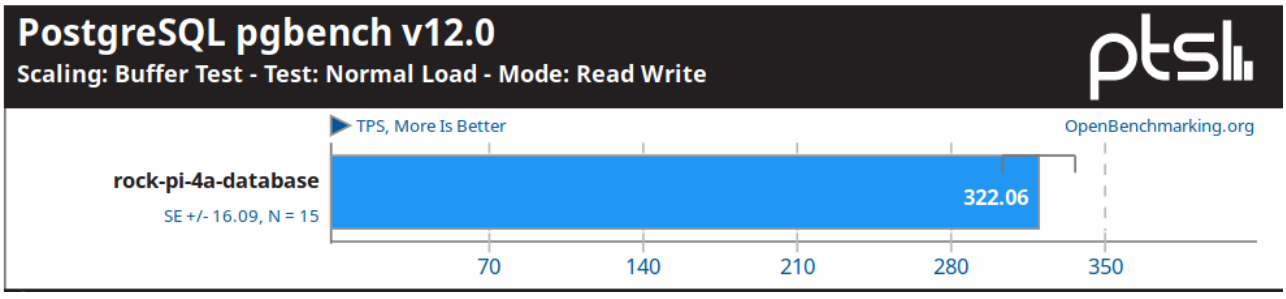
Rock Pi 4A by Radxa

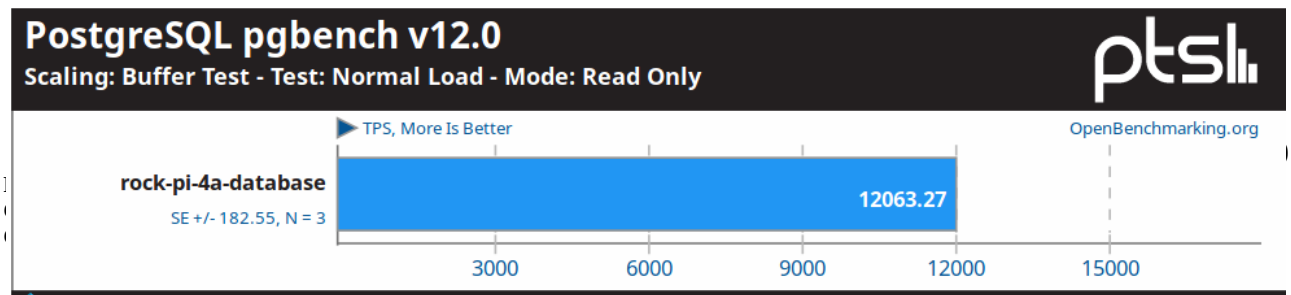
ARMv8 Cortex-A72 @ 1.42GHz (6 Cores)	Processor
ROCK PI 4B	Motherboard
2048MB	Memory
16GB SD16G	Disk
LLVMpipe	Graphics
Acer AL1917	Monitor
Debian 9.9	OS
4.4.154-90-rockchip-ga14f6502e045 (aarch64)	Kernel
LXDE 0.9.3	Desktop
X Server 1.19.3	Display Server
modesetting 1.19.3	Display Driver
3.3 Mesa 13.0.6 Gallium 0.4 (LLVM 3.9 128 bits)	OpenGL
GCC 6.3.0 20170516	Compiler
ext4	File-System

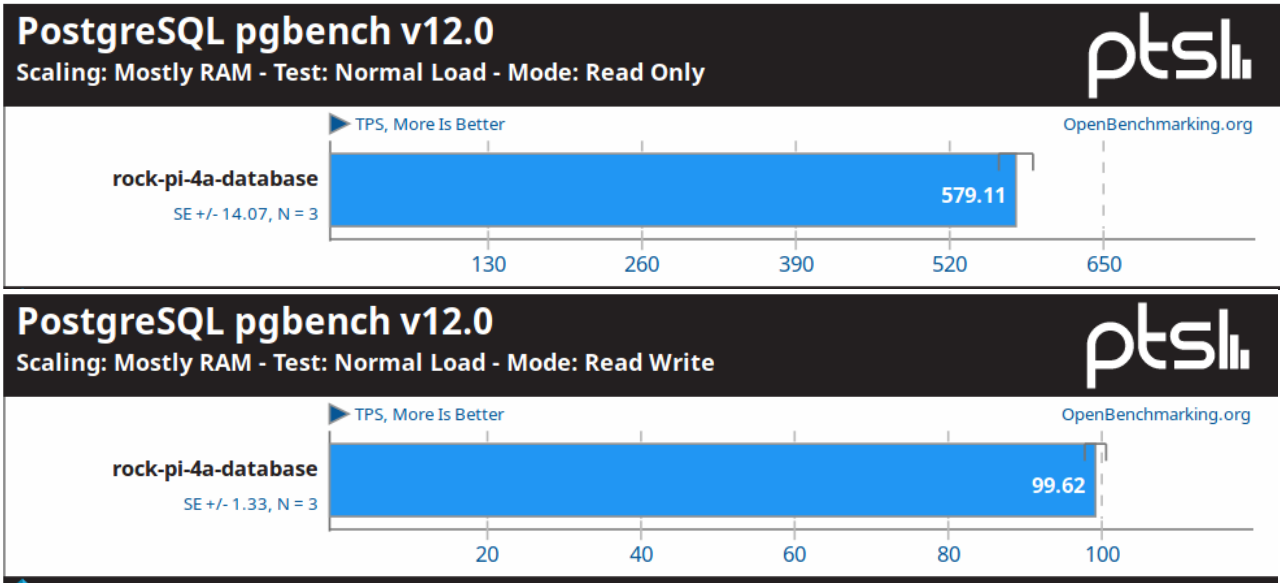
The Rock Pi 4A was the second board tested and the first one used for the list of test selected. As it's possible to see in the picture, Phoronix Test Suite The OS used was the Debian suggested from manufacturer. All test was successful, then we tried the ones tested on Pi Plus but we got some errors in this board too (for the time coded compilation one, only one of the two test succeeded):











Notes on Rock Pi 4A:

- The Debian OS was not able to install the Phoronix Test Suite .deb package. It was necessary to download the tarball, extract the binaries, install the dependencies and execute the software.
- Phoronix Test Suite doesn't recognize the difference between Rock Pi 4A (this one, with internet connection provided by Ethernet connector) and 4B (with internet provided by WiFi module), as it's possible to see in the specifications image at the beginning of the paragraph.

Nano Pi Duo2 by FriendlyArm

ARMv7 rev 5 @ 1.01GHz (4 Cores)	Processor
sun8i	Motherboard
491MB	Memory
16GB SD16G	Disk
Ubuntu 16.04	OS
4.14.111 (armv7l)	Kernel
GCC 5.4.0 20160609	Compiler
overlayfs	File-System
1920x2160	Screen Resolution

The Nano Pi Duo2 have only a microUSB port and a SD holder and the official manual suggest to buy the shield created for the board, which provides USB ports and a HDMI port. Since the idea for this board was to use as it is to have a small and cheap device, we try to communicate with it via the microUSB OTG port. As default there are no SSH or serial connections to the board via microUSB. Since

this board is equipped with a WiFi module, we implement the Orange Pi i96 network configuration in the official Ubuntu image, in order to make the board connect to an orangepi hot-spot with password “orangepi” and then connect to the board via SSH. Thanks to its 512Mb of RAM, this board was able to execute Phoronix Test Suite.

Notes on Nano Pi Duo2:

- 2 pin headers are included in the package
- As the Orange Pi i96, the response time of the SSH connection is significant.
- The official image is not updated: even if the date indicated an image 2 weeks old, hundreds of packages needed an update.
- The image doesn’t have an unzip utility, it was needed to install also zip and unzip to run the tests.

Banana Pi M2 Ultra by Sinovoip

ARMv7 Cortex-A7 @ 1.20GHz (4 Cores)	Processor
sun8iw11p1	Motherboard
2048MB	Memory
16GB SD16G + 8GB 8WPD3R	Disk
Debian 9.11	OS
3.10.108-BPI-M2U-Kernel (armv7l)	Kernel
GCC 6.3.0 20170516	Compiler
ext4	File-System
1280x1440	Screen Resolution

The Banana Pi M2 Ultra was the most responsive one during evaluation. It was very fast to boot and update the official Ubuntu server image using an SD and an ethernet connection. The Phoronix Test Suite was not present in the apt and it was necessary to use the deb package, fixing the missing dependencies following the commands suggested directly by apt commands in console.

Notes on Banana Pi M2 Ultra:

- The board has a eMMC integrated memory. We didn’t use it to test all boards in the same conditions
- The board, when powered with 2A current, doesn’t boot with devices connected with it. After removing the keyboard connected via USB and the monitor connected via HDMI, the board boots. When the LEDs starts to blink, it’s possible to connect.

- Some OS images are available only via Baidu host, which is completely in Chinese. Also, image archives hosted in Baidu are password-protected, making the download even more difficult.

- Unlike the Orange Pi images, the IMG file of Banana OS has included some free space, and it’s not required to resize the root partition of the OS.
- The John the ripper test profile requires the “libssl” library. Due to the configuration of the provided images (tested both Ubuntu server and Debian server current release), it wasn’t possible to resolve the dependency and execute the test.

Raspberry Pi

ARMv7 @ 1.00GHz (1 Core)	Processor
BCM2835 Raspberry Pi Zero W Rev 1.1	Motherboard
370MB	Memory
4GB SU04G	Disk
BCM2708	Graphics
Raspbian 9.11	OS
4.14.79+ (armv6l)	Kernel
GCC 6.3.0 20170516	Compiler
ext4	File-System
656x416	Screen Resolution

ARMv7 rev 4 @ 1.20GHz (4 Cores)	Processor
BCM2709 Raspberry Pi 3 Model B Rev 1.2	Motherboard
925MB	Memory
16GB SD16G	Disk
LLVMpipe	Graphics
Raspbian 8.0	OS
4.1.18-v7+ (armv7l)	Kernel
LXDE 0.7.2	Desktop
X Server 1.17.2	Display Server
3.0 Mesa 11.1.0 Gallium 0.4	OpenGL
GCC 4.9.2	Compiler
ext4	File-System

To have a comparison to well known boards, we made the tests also on some Raspberry Pi. We choose a Raspberry Pi 3B for a comparison with standard boards and a Raspberry Pi Zero W for a comparison with micro and less expensive boards. The dependencies of the test didn’t have installation issues, in contrast of some of the previous tested boards.

Note on Raspberry:

- The Phoronix Test Suite is present in the apt-get package list of Raspbian but its dependencies doesn’t match the current version of packages (php5 vs php7) and the installation fails, but following the instructions printed in console apt was able to fix them.

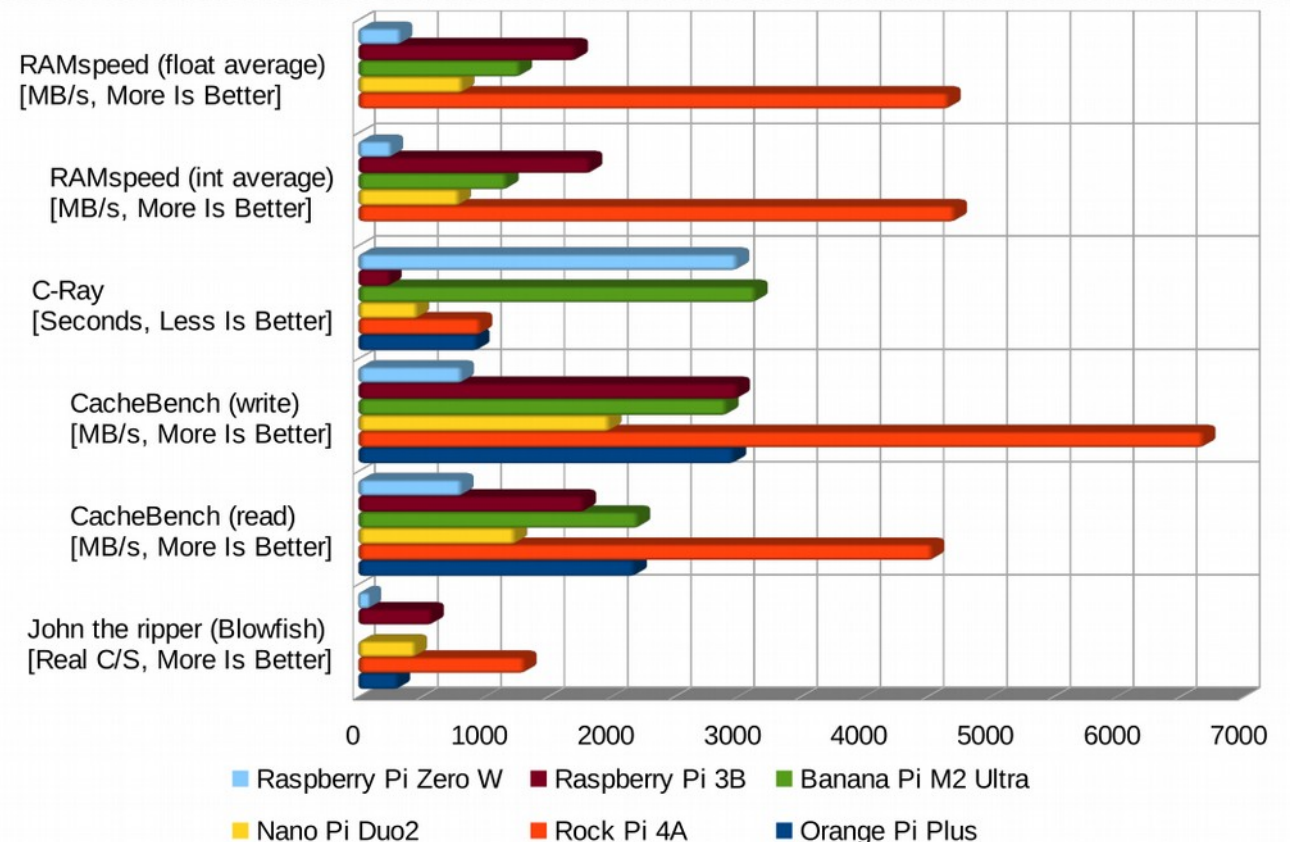
Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

Results tables

Here there are the tests results. Since the Pi Plus stopped working during the evaluations, its data had been integrated with the ones available from public benchmarks shared at

https://openbenchmarking.org

	Orange Pi Plus	Rock Pi 4A	Nano Pi Duo2	Banana Pi M2 Ultra	Raspberry Pi 3B	Raspberry Pi Zero W
John the ripper (Blowfish) [Real C/S, More Is Better]	322	1323	457	0	590	92,73
CacheBench (read) [MB/s, More Is Better]	2196,39	4550,74	1247,31	2222,81	1794,44	824,29
CacheBench (write) [MB/s, More Is Better]	2983,31	6696,74	1993,96	2922,43	3011,89	826,26
C-Ray [Seconds, Less Is Better]	951,1	972,53	472,52	3152,44	255,79	3005,63
RAMspeed (int average) [MB/s, More Is Better]	0	4729,03	804,36	1181,41	1843,88	265,99
RAMspeed (float average) [MB/s, More Is Better]	0	4679,99	830,44	1287,14	1729,99	338,95



Project: Beacon Südtirol - Alto Adige
Code: FESR-2023
CUP: B31H17000060001

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/.