UM2043

User manual

Getting started with the FP-CLD-AZURE1 software for IoT node with Wi-Fi or Ethernet, NFC, sensors and motor control, connected to Microsoft Azure IoT

Introduction

FP-CLD-AZURE1 is an STM32Cube function pack which lets you safely connect your IoT node to Microsoft Azure IoT, transmit sensor data and receive commands from Azure cloud applications.

It fully supports Azure device management primitives and includes a sample implementation for firmware update over the air (FOTA) and one for motor control. By using a mobile device with NFC, Wi-Fi and Ethernet connectivity links are easily configured.

This software, together with the suggested combination of STM32 and ST devices, can be used, for example, to develop sensor-to-cloud applications for a broad range of use cases, such as smart home or smart industry.

The software runs on the STM32 microcontroller and includes drivers for the Wi-Fi and Ethernet connectivity, dynamic NFC/ RFID tag, motion and environmental sensors, as well as two axis stepper motors.

1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
AP	Access point
BSP	Base support package
FOTA	Firmware update over-the-air
GPIO	General purpose input/output
HAL	Hardware abstraction layer
HTML	Hypertext markup language
HTTP	Hypertext transfer protocol
IDE	Integrated development environment
IoT	Internet of things
I^2C	Inter-integrated circuit
MCU	Microcontroller unit
MEMS	Micro electro-mechanical systems
NDEF	NFC data exchange format
ODE	Open development environment
REST API	Representational state tranfer apis
SDK	Software development kit
SMD	Surface mount device
SSID	Service set identifier
UART	Universal asynchronous receiver/transmitter
URL	Uniform resource locator
Wi-Fi	Wireless LAN based on IEEE 802.11
WLAN	Wireless local area network

2 FP-CLD-AZURE1 software description

2.1 Overview

The package features:

- Complete firmware to safely connect an IoT node with sensors and actuators to Microsoft Azure IoT, using Wi-Fi or Ethernet communication technology
- Middleware libraries featuring the Microsoft Azure IoT software development kit, Wi-Fi and NFC connectivity, transport-level security (mbedTLS), Real-time Operating System (FreeRTOS) and meta-data management
- Ready-to-use binaries to connect the IoT node to a web dashboard running on Microsoft Azure, for sensor data visualization, two axis stepper motor control and device management (FOTA)
- Sample implementations available for STM32L4 Discovery Kit for IoT node (B-L475E-IOT01A) with and without X-NUCLEO-IHM02A1, or for X-NUCLEO-IKS01A2, X-NUCLEO-IDW01M1, X-NUCLEO-IHM02A1 and X-NUCLEO-NFC04A1, when connected in different combinations to a NUCLEO-F401RE, a NUCLEO-L476RG or a NUCLEO-F429ZI development board
- Easy portability across different MCU families, thanks to STM32Cube
- Free, user-friendly license terms
- STM32 Nucleo boards are Microsoft Azure certified for IoT

How does this STM32Cube function pack complement STM32Cube?

This software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller. The package extends STM32Cube by providing a board support package (BSP) for Wi-Fi and sensor expansion boards.

The package integrates the Azure IoT device SDK middleware with APIs to simplify interaction between STM32 Nucleo or Discovery Kit for IoT node, and the Microsoft Azure IoT services. You can use it to prototype end-to-end sensors-to-cloud IoT applications, by registering your board to Microsoft Azure IoT and begin exchanging real-time sensor data and commands. A web dashboard based on Microsoft Azure is also provided free of charge to facilitate the evaluation of the function pack.

For Azure license terms, visit https://azure.microsoft.com.

For Microsoft Azure IoT certification information, visit: http://azure.com/certifiedforiot.

2.2 Architecture

What can you do with STM32Cube function packs?

The STM32Cube function packs leverage the modularity and interoperability of STM32 Nucleo and X-NUCLEO boards, and STM32Cube and X-CUBE software, to create function examples, embodying some of the most common use cases, for each application area.

These software function packs are designed to exploit as much as possible the underlying STM32 ODE hardware and software components to best fit the requirements of final users' applications.

Moreover, function packs may include additional libraries and frameworks which do not present the original X-CUBE packages, thus enabling new functionalities and creating a real and usable system for developers.

■

What is STM32Cube?

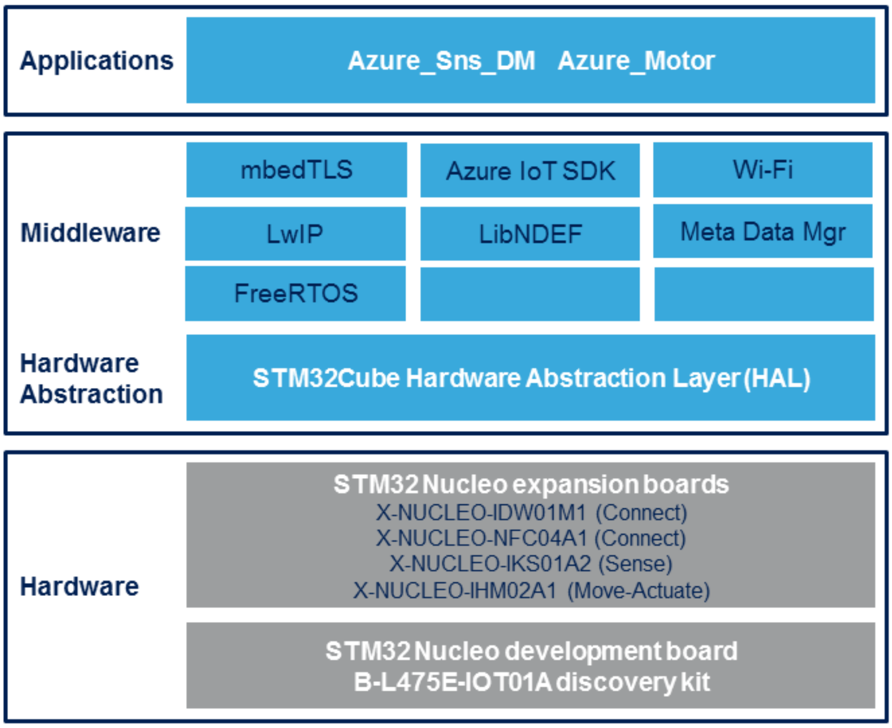
STMCube™ is designed by STMicroelectronics to reduce development effort, time and cost across the entire STM32 portfolio.

STM32Cube version 1.x includes:

- STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive embedded software platform specific to each series (such as the STM32CubeF4 for the STM32F4 series), which includes:
  - the STM32Cube HAL embedded abstraction-layer software, ensuring maximized portability across the STM32 portfolio
  - a consistent set of middleware components such as RTOS, USB, TCP/IP and graphics
  - all embedded software utilities with a full set of examples

2.2.1 FP-CLD-AZURE1 architecture

Figure 1. Software architecture


Applications
Azure_Sns_DM
Azure_Motor
Middleware
mbedTLS
Azure IoT SDK
Wi-Fi
LwIP
LibNDEF
Meta Data Mgr
FreeRTOS
Hardware Abstraction
STM32Cube Hardware Abstraction Layer (HAL)
Hardware
STM32 Nucleo expansion boards
X-NUCLEO-IDW01M1 (Connect)
X-NUCLEO-NFC04A1 (Connect)
X-NUCLEO-IKS01A2 (Sense)
X-NUCLEO-IHM02A1 (Move-Actuate)
STM32 Nucleo development board
B-L475E-IOT01A discovery kit

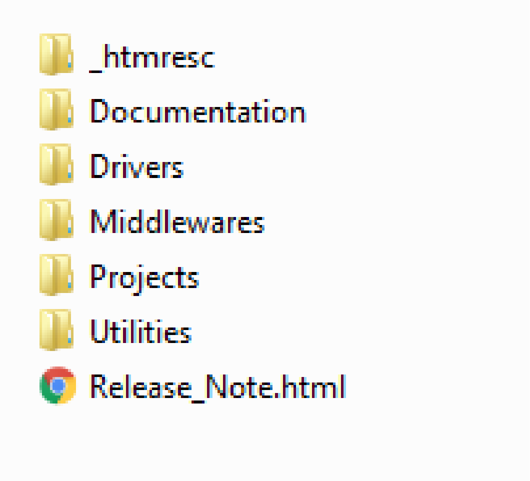
The software layers used by the application to access and use the STM32 microcontroller and the Wi-Fi (for NUCLEO-F401RE or NUCLEO-L476RG) or Ethernet (NUCLEO-F429ZI), sensors, NFC tag and motor control are:

- **STM32Cube HAL driver layer**: a simple, generic, multi-instance set of APIs (application programming interfaces) to interact with the upper layer applications, libraries and stacks. The APIs are based on a common framework so that overlying software like middleware can implement functions and routines without specific microcontroller unit (MCU) hardware configurations. This structure improves library code reusability and guarantees easy portability across other devices.

- **Board support package (BSP) layer**: drives the STM32 Nucleo board peripherals like the LED, user button, etc. (not the MCU), with a specific set of APIs. This interface also helps in identifying the specific board version.
- **Middleware layer**: contains the FreeRTOS operating system, MetaDataManager to save Meta Data in the STM32 Flash memory, mbedTLS, LwIP, NDEF library to read information from the NFC and the Microsoft Azure IoT device SDK (https://github.com/Azure/azure-iot-sdks) to facilitate the connection of STM32 Nucleo with Azure IoT services.

2.3 Folder structure

Figure 2. **Package folder structure**



_htmresc
Documentation
Drivers
Middlewares
Projects
Utilities
Release_Note.html

The following folders are included in the software package:

- **Documentation**: with a compiled HTML file generated from the source code detailing the software components and APIs (one for each project).
- **Drivers**: the HAL drivers and the board-specific drivers for each supported board or hardware platform, including those for the on-board components, and the CMSIS vendor-independent hardware abstraction layer for the ARM Cortex-M processor series.
- **Middlewares**: the middleware interface for Wi-Fi, NDEF library, MetaDataManager, FreeRTOS real-time operating system, mbedTLS, LwIP and the porting of Microsoft Azure IoT device SDK.
- **Utilities**: contains BootLoader for B-L475E-IOT01A, NUCLEO-L476RG and for NUCLEO-F429ZI
- **Projects** contains two sample applications, Azure_Sns_DM and and Azure_Motor. Azure_Sns_DM application (B-L475E-IOT01A, NUCLEO-L476RG, NUCLEO-F429ZI and NUCLEO-F401RE) reads sensor data and transmit them to Micorosft Azure IoT Hub via Wi-Fi or Etherent and enables remote control of the device thanks to full support for Microsoft Azure primitives for device management; for B-L475E-IOT01A, NUCLEO-L476RG and NUCLEO-F429ZI it also contains an example of remote firmware update procedure. Azure_Motor sample application (B-L475E-IOT01A, NUCLEO-L476RG) contains drivers and functions to remotely control two axis stepper motors through the X-NUCLEO-IHM02A1 expansion board. Sample applications can be compiled under the IAR Embedded Workbench for ARM, RealView Microcontroller Development Kit (MDK-ARM-STM32) or System Workbench for STM32 development environments. For each sample application, the folder also contains a configurable pre-compiled binary to connect devices with a custom Azure account, and another pre-configured binary which can be used together with a companion Azure-based web dashboard for easy sensor data visualization and device control.

2.4 Flash memory management

The sample application uses the Flash memory to:

1. save the Wi-Fi credentials or Ethernet configuration and the Azure device connection string in the Meta Data Manager (for all platforms);

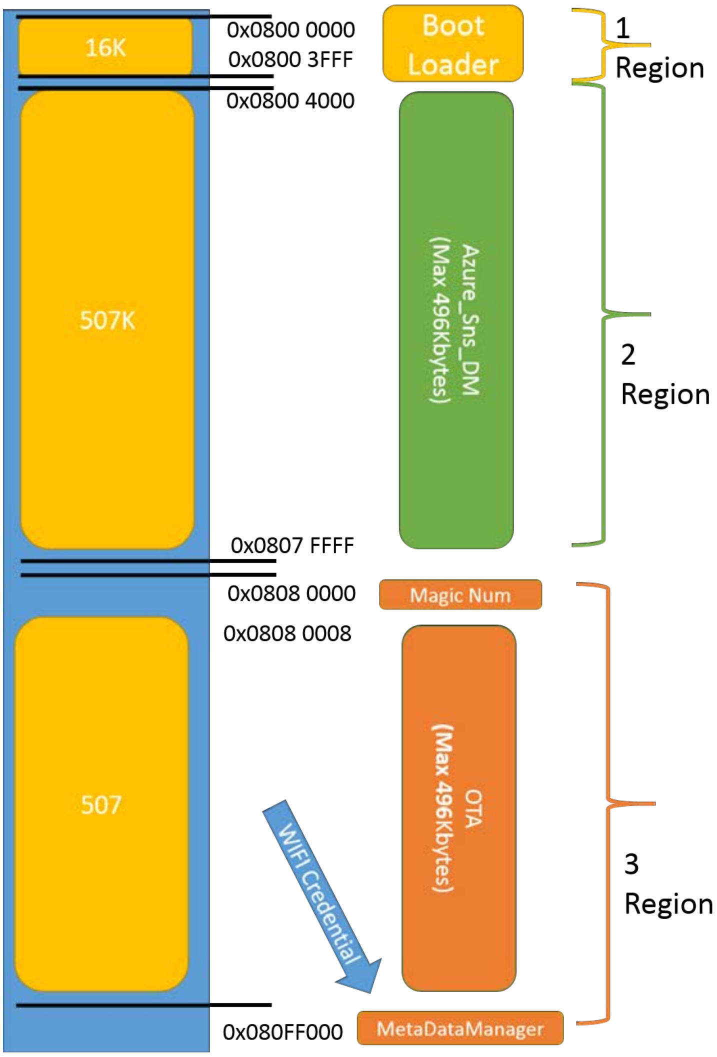
2. allow the Firmware-Over-The-Air update (NUCLEO-L476RG or NUCLEO-F429ZI, B-L475E-IOT01A).

To enable these features the Flash is divided in different regions (see Figure 3. Azure_Sns_DM Flash structure for NUCLEO-STM32L476RG):

1. the first region contains a custom boot loader (required for firmware update);
2. the second region contains the application firmware;
3. the third region is used in a firmware update procedure to store the new downloaded firmware before updating it, and to save the Wi-Fi or Ethernet credentials inside the Meta Data Manager.

The Meta Data Manager is placed at the end of the Flash (0x080FF000 for STM32L476RG).(For more information on the Flash memory management, refer to Reference manual STM32L4x6 advanced ARM®-based 32-bit MCUs (RM0351). For STM32F29ZI refer to Reference Manual STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM®-based 32-bit MCUs (RM0090)).

Figure 3. **Azure_Sns_DM Flash structure for NUCLEO-STM32L476RG**



2.5 The boot process for the firmware update over-the-air (FOTA) application

In order to enable the firmware update procedure (NUCLEO-L476RG or NUCLEO-F429ZI, B-L475E-IOT01A), the Azure_Sns_DM application binary cannot be flashed at the beginning of the Flash memory (address 0x08000000), and is therefore compiled to run from the beginning of the second Flash region (at 0x08004000).

To enable this procedure, a vector table offset is set in Src/system_stm32l4xx.c (for STM32L476): `#define VECT_TAB_OFFSET 0x4000`.

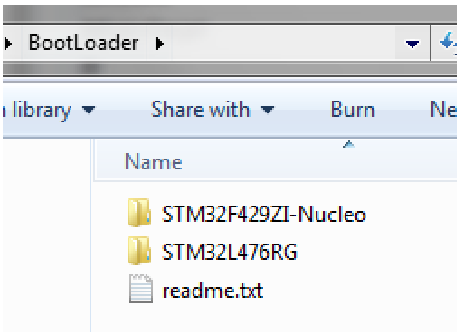
The linker script also requires changes; for example, the Linker script for Azure_Sns_DM running on STM32L476RG and compiled using IAR Embedded Workbench for ARM is modified as follows:

```
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08004000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08004000;
define symbol __ICFEDIT_region_ROM_end__ = 0x0807FFFF;
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;
define symbol __ICFEDIT_region_RAM_end__ = 0x20017FFF;
define symbol __ICFEDIT_region_SRAM2_start__ = 0x10000000;
define symbol __ICFEDIT_region_SRAM2_end__ = 0x10007FFF;
/*-Sizes-*/ define symbol __ICFEDIT_size_cstack__ = 0x6000;
define symbol __ICFEDIT_size_heap__ = 0x11000;
```

Using the above linker script, the maximum usable code size is fixed at 496 KB.

Before flashing the compiled Azure_Sns_DM firmware, you must flash the appropriate bootloader binary for NUCLEO-L476RG/F429ZI or B-L475E-IOT01Ax , available in the Utilities\BootLoader folder, in the first Flash region (address 0x08000000).

Figure 4. **BootLoader folder content**



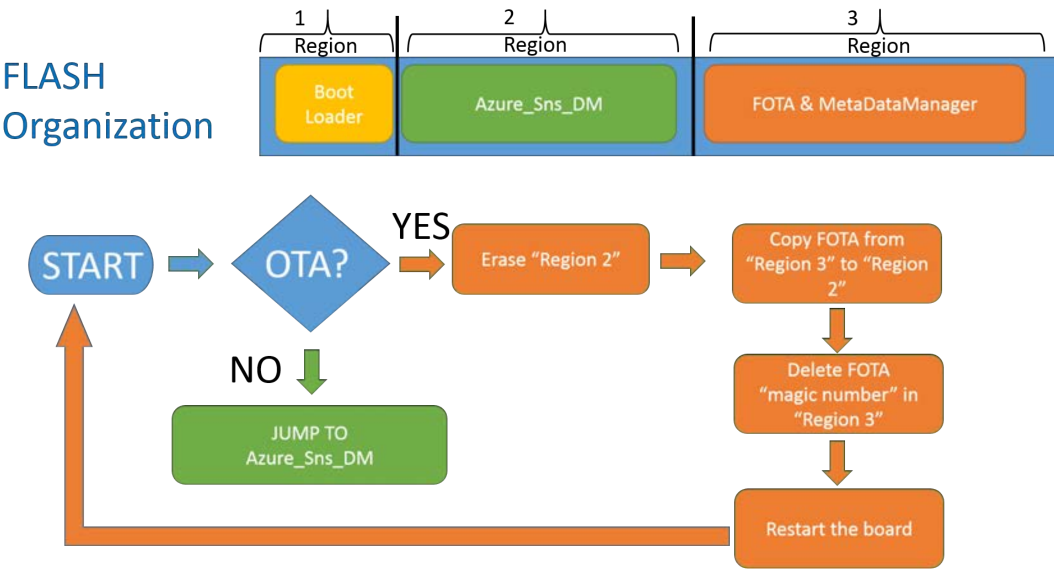
STM32F429ZI-Nucleo
STM32L476RG
readme.txt

When the firmware update procedure is activated, the new firmware is downloaded and copied in the third Flash region.

After board reset, the following procedure applies:

- if there is a new firmware downloaded in the third Flash region, the BootLoader overwrites the second Flash region (containing the current firmware), replaces its content with the new firmware and restarts the board;
- if there is no new firmware downloaded, the BootLoader jumps to the firmware stored in region 2.

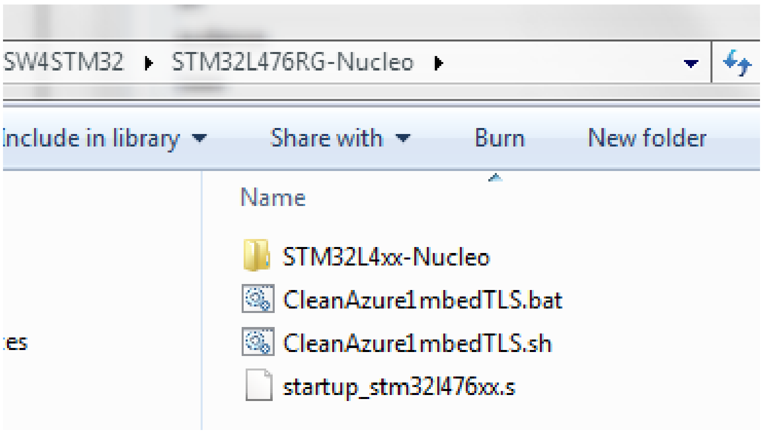
Figure 5. **Azure_Sns_DM boot sequence**



2.6 The installation process for the firmware update over-the-air (FOTA) application

For NUCLEO-L476RG or NUCLEO-F429ZI and B-L475E-IOT01A platforms running the Azure_Sns_DM application, the flashing procedure is simplified by a script available for each IDE (IAR/RealView/System Workbench). The script writes the compiled firmware and BootLoader in the right memory position.

Figure 6. **Project folder content example**

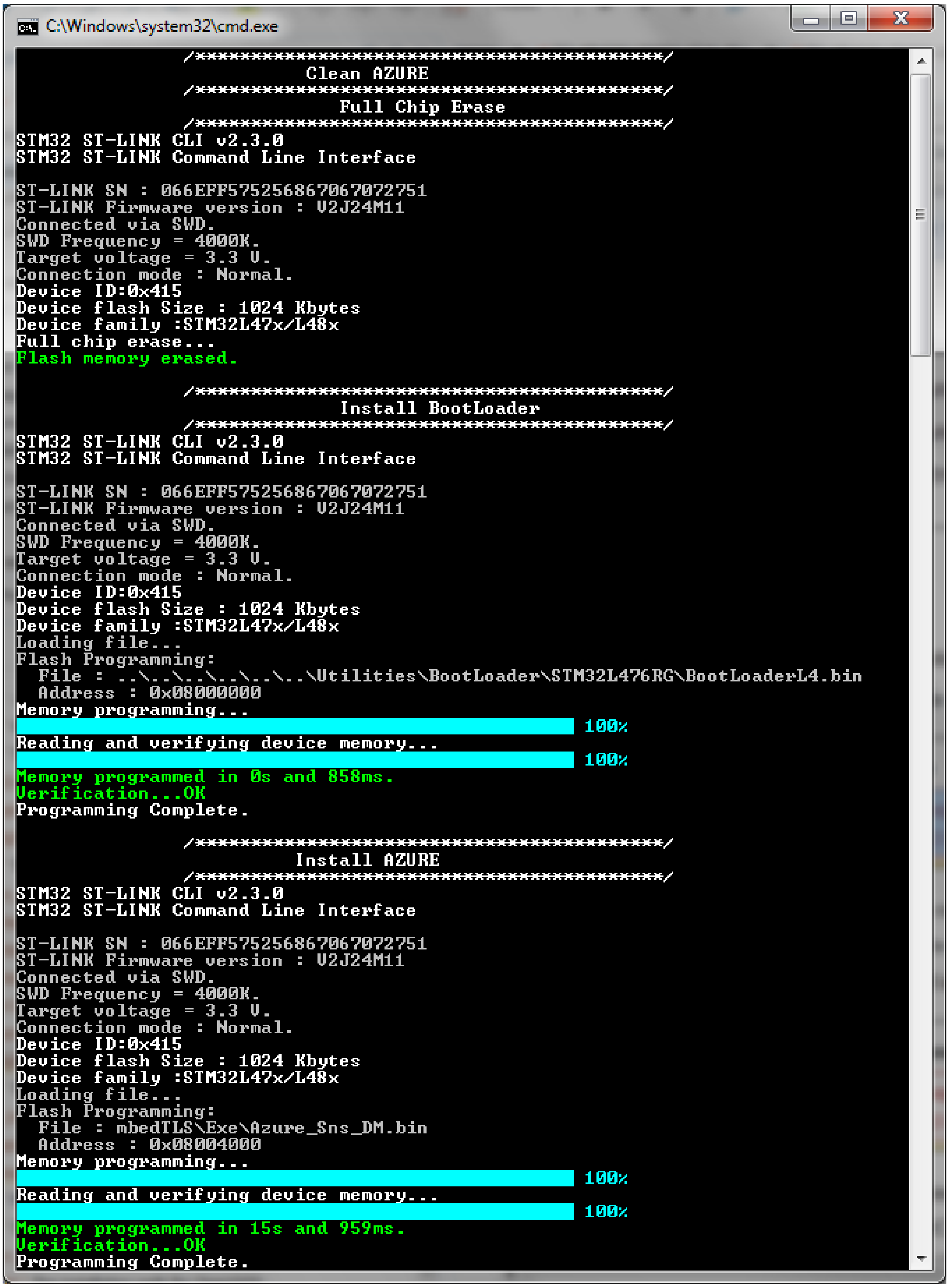


STM32L4xx-Nucleo
CleanAzure1mbedTLS.bat
CleanAzure1mbedTLS.sh
startup_stm32l476xx.s

In particular, the script:

- erases the full Flash;
- flashes the BootLoader at the correct position (0x08000000);
- flashes the Azure_Sns_DM firmware at the correct position (0x08004000).

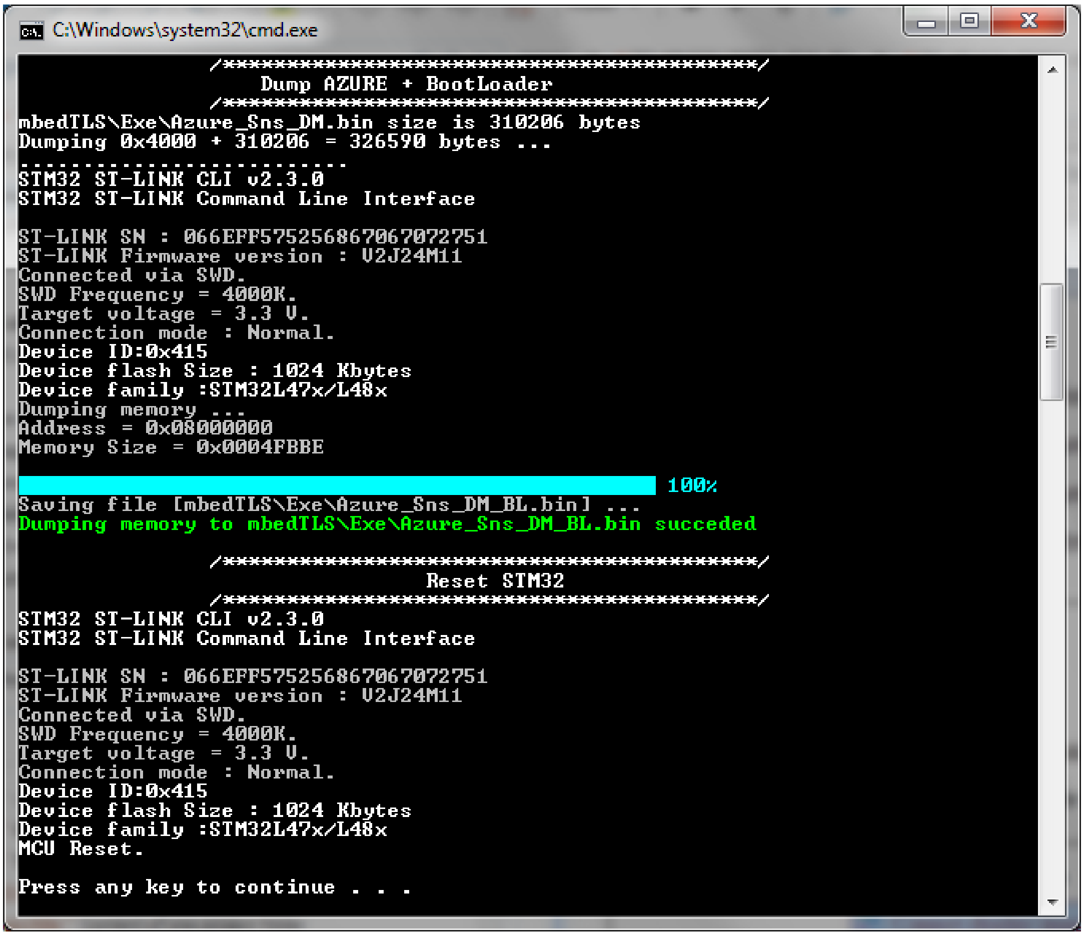
Figure 7. BootLoader and Azure_Sns_DM installation



```
               /******************************************/
                          Clean AZURE
               /******************************************/
                             Full Chip Erase
               /******************************************/
STM32 ST-LINK CLI v2.3.0
STM32 ST-LINK Command Line Interface

ST-LINK SN : 066EFF575256867067072751
ST-LINK Firmware version : V2J24M11
Connected via SWD.
SWD Frequency = 4000K.
Target voltage = 3.3 V.
Connection mode : Normal.
Device ID:0x415
Device flash Size : 1024 Kbytes
Device family :STM32L47x/L48x
Full chip erase...
Flash memory erased.

               /******************************************/
                             Install BootLoader
               /******************************************/
STM32 ST-LINK CLI v2.3.0
STM32 ST-LINK Command Line Interface

ST-LINK SN : 066EFF575256867067072751
ST-LINK Firmware version : V2J24M11
Connected via SWD.
SWD Frequency = 4000K.
Target voltage = 3.3 V.
Connection mode : Normal.
Device ID:0x415
Device flash Size : 1024 Kbytes
Device family :STM32L47x/L48x
Loading file...
Flash Programming:
  File : ..\..\..\..\..\..\Utilities\BootLoader\STM32L476RG\BootLoaderL4.bin
  Address : 0x08000000
Memory programming...
██████████████████████████████████████████████████ 100%
Reading and verifying device memory...
██████████████████████████████████████████████████ 100%
Memory programmed in 0s and 858ms.
Verification...OK
Programming Complete.

               /******************************************/
                         Install AZURE
               /******************************************/
STM32 ST-LINK CLI v2.3.0
STM32 ST-LINK Command Line Interface

ST-LINK SN : 066EFF575256867067072751
ST-LINK Firmware version : V2J24M11
Connected via SWD.
SWD Frequency = 4000K.
Target voltage = 3.3 V.
Connection mode : Normal.
Device ID:0x415
Device flash Size : 1024 Kbytes
Device family :STM32L47x/L48x
Loading file...
Flash Programming:
  File : mbedTLS\Exe\Azure_Sns_DM.bin
  Address : 0x08004000
Memory programming...
██████████████████████████████████████████████████ 100%
Reading and verifying device memory...
██████████████████████████████████████████████████ 100%
Memory programmed in 15s and 959ms.
Verification...OK
Programming Complete.
```

The same script also dumps a unique image file (containing the BootLoader and the Azure_Sns_DM firmware) that can be directly flashed to the beginning of the Flash memory.

Figure 8. BootLoader and Azure_Sns_DM Dump process



```
                    /*********************************************/
                        Dump AZURE + BootLoader
                    /*********************************************/
mbedTLS\Exe\Azure_Sns_DM.bin size is 310206 bytes
Dumping 0x4000 + 310206 = 326590 bytes ...
..........................
STM32 ST-LINK CLI v2.3.0
STM32 ST-LINK Command Line Interface

ST-LINK SN : 066EFF575256867067072751
ST-LINK Firmware version : V2J24M11
Connected via SWD.
SWD Frequency = 4000K.
Target voltage = 3.3 V.
Connection mode : Normal.
Device ID:0x415
Device flash Size : 1024 Kbytes
Device family :STM32L47x/L48x
Dumping memory ...
Address = 0x08000000
Memory Size = 0x0004FBBE

                                                   100%
Saving file [mbedTLS\Exe\Azure_Sns_DM_BL.bin] ...
Dumping memory to mbedTLS\Exe\Azure_Sns_DM_BL.bin succeded

                    /*********************************************/
                                Reset STM32
                    /*********************************************/
STM32 ST-LINK CLI v2.3.0
STM32 ST-LINK Command Line Interface

ST-LINK SN : 066EFF575256867067072751
ST-LINK Firmware version : V2J24M11
Connected via SWD.
SWD Frequency = 4000K.
Target voltage = 3.3 V.
Connection mode : Normal.
Device ID:0x415
Device flash Size : 1024 Kbytes
Device family :STM32L47x/L48x
MCU Reset.

Press any key to continue . . .
```

For the Linux/OSx operating system there is a similar script, CleanAzure1mbedTLS.sh, that uses OpenOCD instead of the ST-LINK command line

The script is included only in the System Workbench IDE and requires:

- the installation path for OpenOCD according to the local configuration;
- the installation path for STM32 OpenOCD scripts according to the local configuration;
- the library path for OpenOCD according to the variable used for Linux/OSx environments.

```
# 1) Set the Installation path for OpenOCD
# example:
#OpenOCD_DIR="C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.externaltools.openocd.win3
2_1.10.0.201607261143/tools/openocd/"
OpenOCD_DIR=""

# 2) Set the installation path for stm32 OpenOCD scripts
# example:
#OpenOCD_CFC="C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.debug_1.10.0.201607251855/
resources/openocd/scripts"
OpenOCD_CFC=""

# 3) Only for Linux/iOS add openocd library path to _LIBRARY_PATH:
# For iOS example:
#export DYLD_LIBRARY_PATH=${DYLD_LIBRARY_PATH}:${OpenOCD_DIR}"lib/"
```

```
# For Linux example:
#export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${OpenOCD_DIR}"lib/"
```

2.7 Azure IoT sample application description

Azure_Sns_DM sample application for X-NUCLEO-IKS01A2, X-NUCLEO-NFC04A1 expansion boards together with the NUCLEO-F401RE/NUCLEO-L476RG/NUCLEO-F429ZI, and for B-L475E-IOT01A boards is provided in the Projects directory; NUCLEO-F401RE/NUCLEO-L476RG rely on X-NUCLEO-IDW01M1 for Wi-Fi connectivity, NUCLEO-F429ZI uses the integrated Ethernet whereas B-L475E-IOT01A uses an integrated Wi-Fi module. This sample application reads data values from the temperature, humidity, accelerometer and gyroscope sensors and transmits them to the Microsoft Azure IoT Hub via Wi-Fi (NUCLEO-F401RE/NUCLEO-L476RG, B-L475E-IOT01A) or Ethernet (NUCLEO-F429ZI). The sample application also fully supports Azure device management primitives to remotely control the device, and includes, for NUCLEO-L476RG/NUCLEO-F429ZI and B-L475E-IOT01A, a sample to trigger firmware update over-the-air procedure.

Azure_Motor sample application for X-NUCLEO-NFC04A1, X-NUCLEO-IHM02A1 expansion boards together with NUCLEO-L476RG, and for X-NUCLEO-IHM02A1 expansion board together with B-L475E-IOT01Ax, is provided in the Projects directory. This sample application enables to remotely control two axis stepper motors driver through X-NUCLEO-IHM02A1 expansion board.

Note: *You must create a Microsoft Azure account in order to create Azure IoT Hub and test the sample application.*

Detailed information on how to register a free sample account in Azure can be found at azure.microsoft.com/en-us/pricing/free-trial/.

The NFC expansion board can be used by the embedded application to read Wi-Fi access point parameters and, for Ethernet, to configure MAC address and DHCP or static IP/Gateway address; the NFC board can also be used to set the personal credentials to connect with Azure IoT Hub. If the X-NUCLEO-NFC04A1 expansion board is not used, device configuration is always possible via serial terminal, or by entering parameters in the source code and rebuilding the solution.

If the X-NUCLEO-IKS01A2 is not connected to NUCLEO-F401RE/NUCLEO-L476RG/NUCLEO-F429ZI, sensor data are simulated.

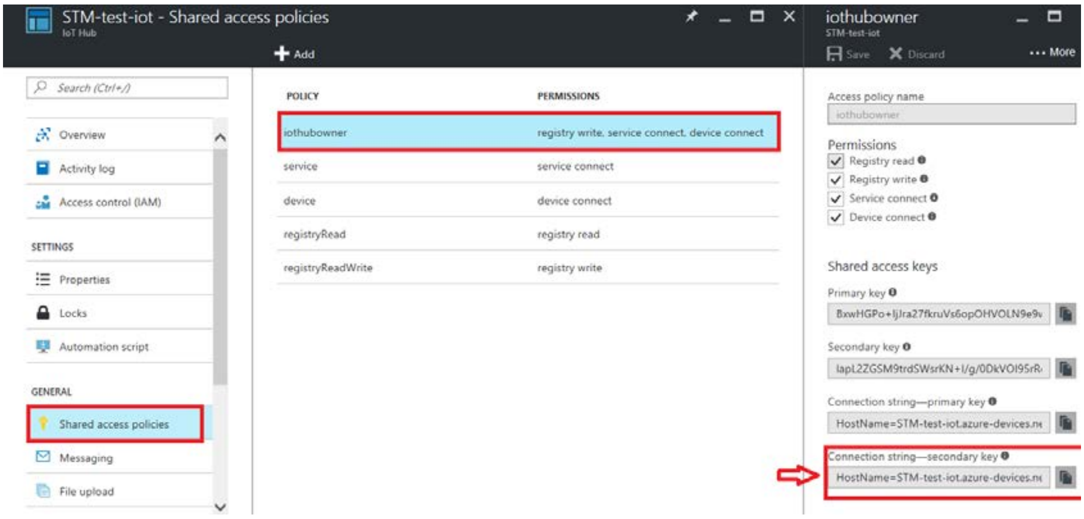
2.7.1 Create Azure IoT Hub instance and generate connection string

Before you can run the supplied sample application, you must first create an account on Microsoft Azure (see https://azure.microsoft.com/en-us/pricing/free-trial/) and then create an instance of the Azure IoT Hub (see).

Afterwards, you must follow the procedure below to register your STM32 Nucleo device.

Step 1. In the Azure Portal, select the IoT Hub service and note down your *IoT Hub connection string*:

Figure 9. IoT Hub connection string in Azure Portal



STM-test-iot - Shared access policies
IoT Hub

POLICY	PERMISSIONS
iothubowner	registry write, service connect, device connect
service	service connect
device	device connect
registryRead	registry read
registryReadWrite	registry write

iothubowner
STM-test-iot

Access policy name: iothubowner

Permissions
☑ Registry read
☑ Registry write
☑ Service connect
☑ Device connect

Shared access keys

Primary key: BxwHGPo+IjJra27fkruVs6opOHVOLN9e9v

Secondary key: lapL2ZGSM9trdSWsrKN+I/g/0DkVOI95rR

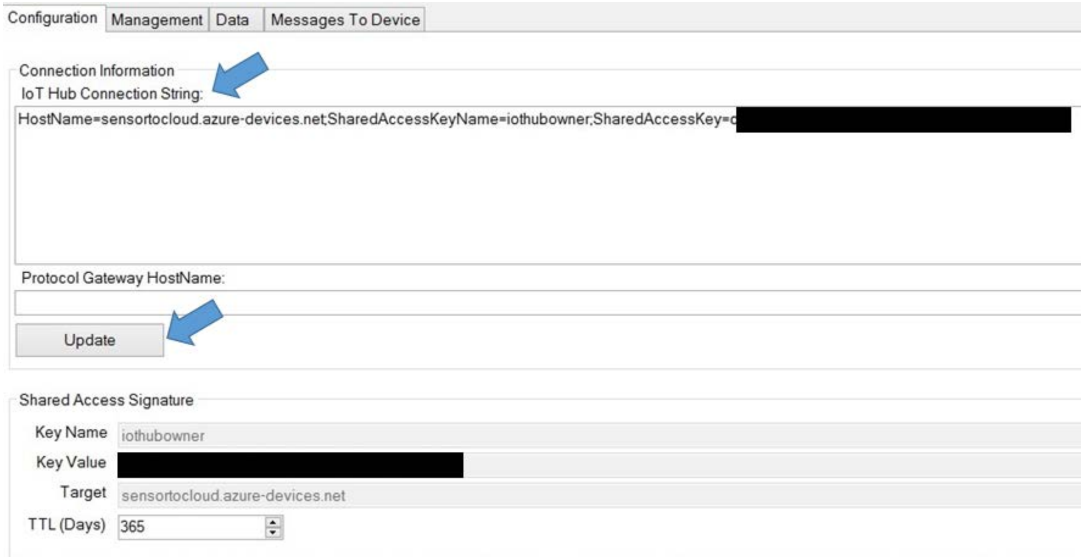
Connection string—primary key: HostName=STM-test-iot.azure-devices.ne

Connection string—secondary key: HostName=STM-test-iot.azure-devices.ne

Step 2. Download and install the Microsoft Device Explorer utility for Windows available at .(Refer to Microsoft documentation to learn how to register and handle devices using other cross-platform tools for OSx and Linux operating systems.)

Step 3. Launch Device Explorer, paste the IoT Hub connection string and then press Update.

Figure 10. **Paste IoT Hub connection string in Device Explorer**



Connection Information

IoT Hub Connection String:

HostName=sensortocloud.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=[redacted]

Protocol Gateway HostName:

Shared Access Signature

Key Name: iothubowner

Key Value: [redacted]

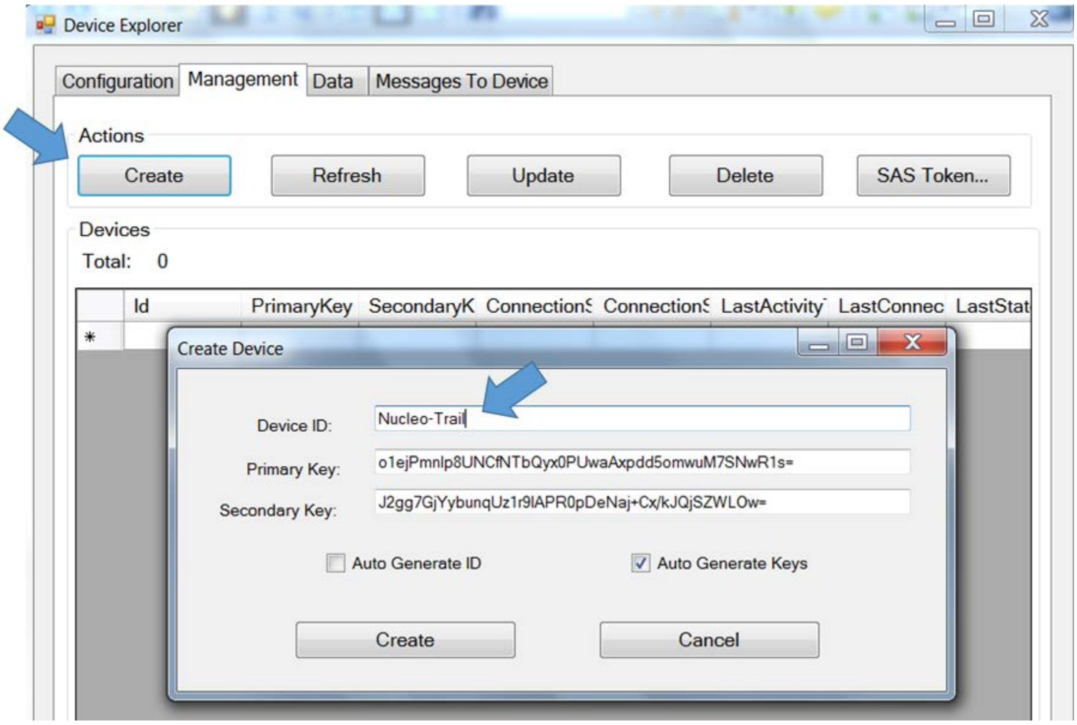
Target: sensortocloud.azure-devices.net

TTL (Days): 365

Step 4. From the Management tab, click Create and insert a unique device ID for your device.

A Device Created window will confirm that the device has been successfully registered with your Azure IoT Hub

Figure 11. **Create a new device for your STM32 Nucleo**



Devices

Total: 0

Device ID: Nucleo-Trail

Primary Key: o1ejPmnlp8UNCfNTbQyx0PUwaAxpdd5omwuM7SNwR1s=

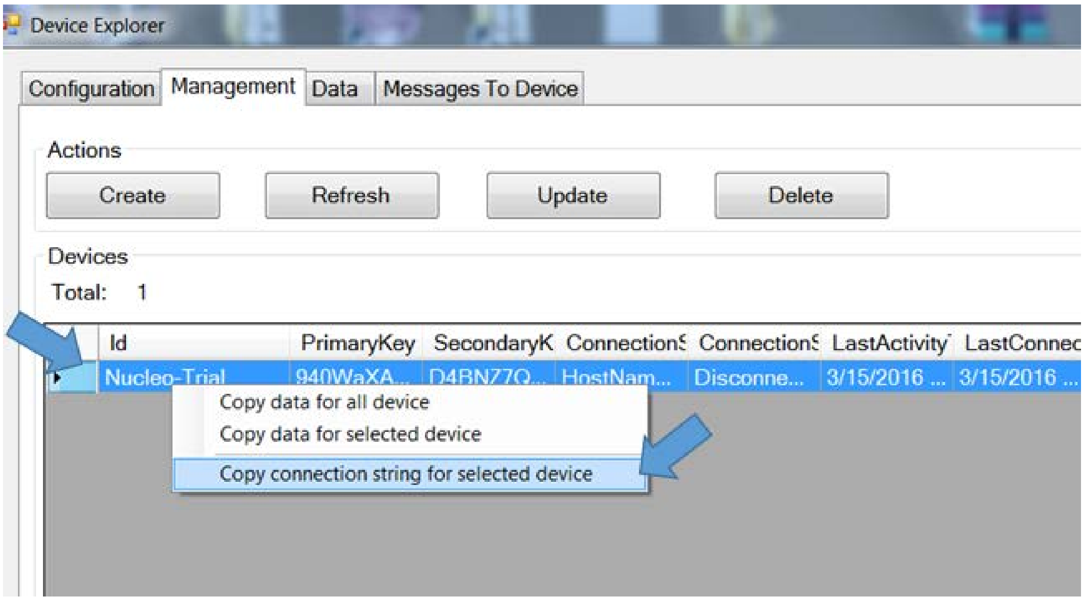
Secondary Key: J2gg7GjYybunqUz1r9lAPR0pDeNaj+Cx/kJQjSZWLOw=

□ Auto Generate ID

☑ Auto Generate Keys

Step 5. Right-click your device in the Devices list and select Copy connection string for selected device.

Figure 12. **Copy connection string for your STM32 Nucleo device**



Id	PrimaryKey	SecondaryK	Connection$	Connection$	LastActivity	LastConnec
Nucleo-Trial	940WaXA...	D4BN77Q...	HostNam...	Disconne...	3/15/2016 ...	3/15/2016 ...

Step 6. Paste the device connection string in a text editor for later usage. The connection string can be entered directly in the sample application source code before recompiling it or can be configured via serial interface and NFC as described in the following sections.

2.7.2 Launch sample application

Once you have completed the procedure in Section 2.7.1 Create Azure IoT Hub instance and generate connection string and set up your system as per Section 3.3 Hardware and software setup, you can proceed to launch the sample application provided with FP-CLD-AZURE1.

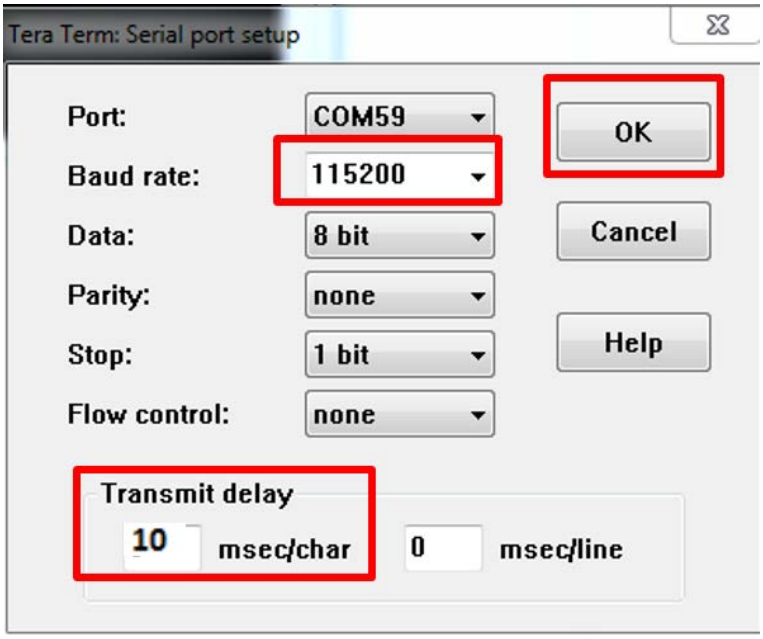
Pre-compiled binaries are available for each platform, together with the project files to rebuild the solution.

A serial terminal interface is necessary to monitor the log of the sample application and can be used for device configuration.

2.7.2.1 *Serial terminal interface setup*

Set up a serial terminal (i.e. TeraTerm) with the parameters reported in the figure below. In particular, set the proper baudrate (115200) and a **Transmit delay** (10 msec/char).

Figure 13. Serial port configuration



Tera Term: Serial port setup

Port: COM59

Baud rate: 115200

Data: 8 bit

Parity: none

Stop: 1 bit

Flow control: none

Transmit delay

10 msec/char 0 msec/line

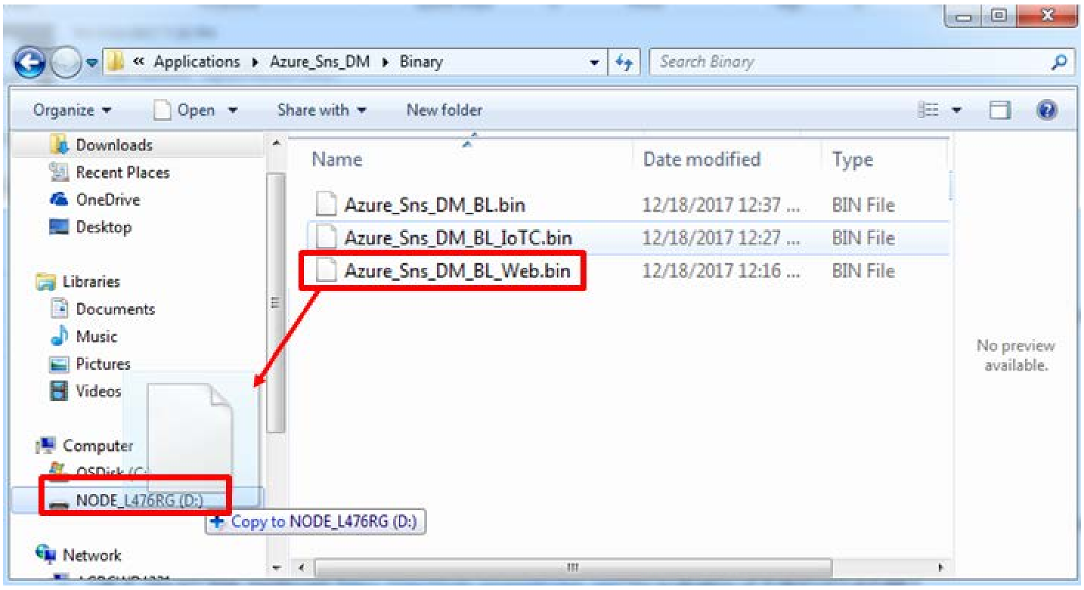
2.7.2.2 *Use pre-compiled binaries*

Pre-compiled binaries for each platform are contained in *Projects\Multi\Applications\Azure_Sns_DM\Binaries* for **Azure_Sns_DM** application, and in *Projects\Multi\Applications\Azure_Motor\Binaries* for **Azure_Motor** application.

Step 1. **Azure_Sns_DM**: The following pre-compiled binaries can be found for each platform: *Azure_Sns.bin* (NUCLEO-F401RE) and *Azure_Sns_DM_BL.bin* (NUCLEO-L476RG/NUCLEO-F429ZI, B-L475-IOT01A) can be configured with custom parameters for Wi-Fi/Ethernet using serial terminal or NFC and require a personal device connection string to connect with a custom Azure IoT Hub; *Azure_Sns_DM_IoTC.bin* (NUCLEO-L476RG, B-L475-IOT01A) also requires a connection string to be used with Microsoft IoT Central (https://www.microsoft.com/en-us/iot-central) rather than a common Azure IoT Hub. *Azure_Sns_DM_BL_Web.bin* (NUCLEO-L476RG/NUCLEO-F429ZI, B-L475-IOT01A) and *Azure_Sns.bin* (NUCLEO-F401RE) must be used jointly with IoT Web Dashboard described in section Section 2.8 STM32 ODE web dashboard for FP-CLD-AZURE1, and does not require a custom IoT Hub device connection string.

Step 2. **Azure_Motor**. The following pre-compiled binaries can be found for each platform: *Azure_Sns_DM_BL.bin* (NUCLEO-L476RG, B-L475-IOT01A) can be configured with custom parameters for Wi-Fi using serial terminal or NFC and require a personal device connection string to connect with a custom Azure IoT Hub; *Azure_Sns_DM_BL_Web.bin* (NUCLEO-L476RG, B-L475-IOT01A) must be used jointly with IoT Web Dashboard described in section Section 2.8 STM32 ODE web dashboard for FP-CLD-AZURE1, and does not require a custom IoT Hub device connection.

Step 3. Connect your board to the laptop using a Mini-B USB or Micro-B USB cable depending on the platform used; when the board appears as a mass storage device, drag the binary as shown in the picture below.



Name	Date modified	Type
Azure_Sns_DM_BL.bin	12/18/2017 12:37 ...	BIN File
Azure_Sns_DM_BL_IoTC.bin	12/18/2017 12:27 ...	BIN File
Azure_Sns_DM_BL_Web.bin	12/18/2017 12:16 ...	BIN File

Figure 14. **Drag the binary to the connected board**

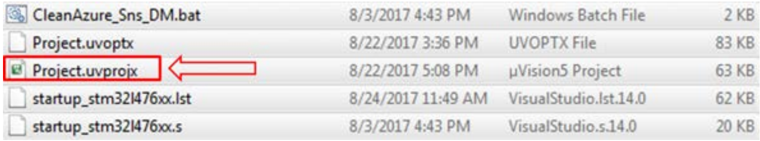
Step 4. After copying the binary, open the serial terminal to view the device message log (see Section 2.7.3 Device configuration to learn how to enter credentials according to the platform used).

2.7.2.3 *Modify and rebuild solution files*

Solution files for the two sample applications can be found in *Projects\Multi\Application\Azure_Sns_DM* and *Projects\Multi\Application\Azure_Motor*. For both applications, solution files can be opened and modified using the following procedure.

Step 1. Open the solution file according to the platform used; i.e. for MDK-ARM and B-L475E-IOT01A, the project file can be found in the folder *Projects\Multi\Applications\Azure_Sns_DM\MDK-ARM\B-L475E-IOT01A*.

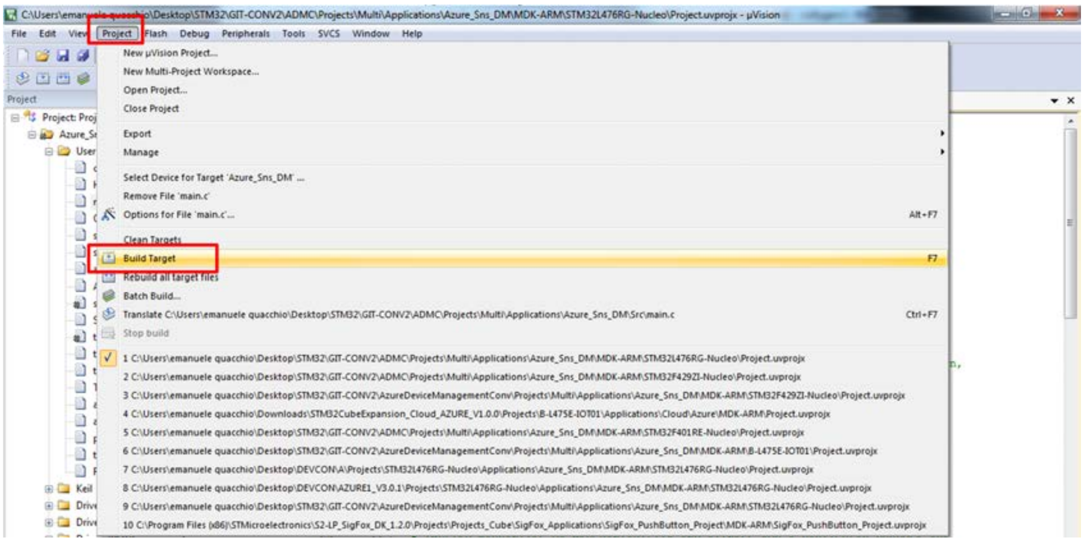
Figure 15. **Select and open a solution file**



CleanAzure_Sns_DM.bat	8/3/2017 4:43 PM	Windows Batch File	2 KB
Project.uvoptx	8/22/2017 3:36 PM	UVOPTX File	83 KB
Project.uvprojx	8/22/2017 5:08 PM	µVision5 Project	63 KB
startup_stm32l476xx.lst	8/24/2017 11:49 AM	VisualStudio.lst.14.0	62 KB
startup_stm32l476xx.s	8/3/2017 4:43 PM	VisualStudio.s.14.0	20 KB

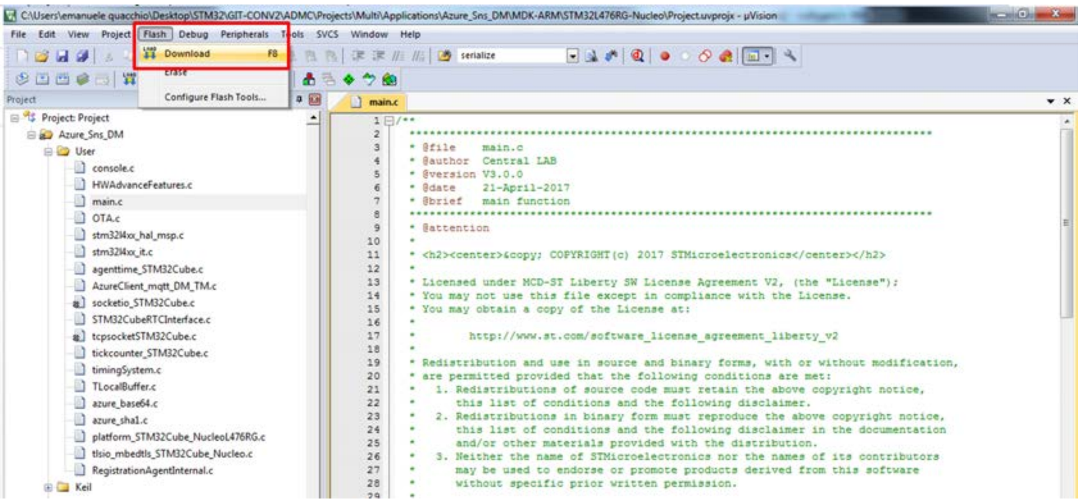
Step 2. Recompile the solution whenever the code is modified according to the selected IDE (i.e. for MDK-ARM):

Figure 16. Recompile a solution file in MDK-ARM


C:\Users\emanuele quacchio\Desktop\STM32\GIT-CONV2\ADMC\Projects\Multi\Applications\Azure_Sns_DM\MDK-ARM\STM32L476RG-Nucleo\Project.uvprojx - µVision
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target 'Azure_Sns_DM' ...
Remove File 'main.c'
Options for File 'main.c'...
Alt+F7
Clean Targets
Build Target
F7
Rebuild all target files
Batch Build...
Translate C:\Users\emanuele quacchio\Desktop\STM32\GIT-CONV2\ADMC\Projects\Multi\Applications\Azure_Sns_DM\Src\main.c
Ctrl+F7
Stop build
1 C:\Users\emanuele quacchio\Desktop\STM32\GIT-CONV2\ADMC\Projects\Multi\Applications\Azure_Sns_DM\MDK-ARM\STM32L476RG-Nucleo\Project.uvprojx
2 C:\Users\emanuele quacchio\Desktop\STM32\GIT-CONV2\ADMC\Projects\Multi\Applications\Azure_Sns_DM\MDK-ARM\STM32F429ZI-Nucleo\Project.uvprojx
3 C:\Users\emanuele quacchio\Desktop\STM32\GIT-CONV2\AzureDeviceManagementConv\Projects\Multi\Applications\Azure_Sns_DM\MDK-ARM\STM32F429ZI-Nucleo\Project.uvprojx
4 C:\Users\emanuele quacchio\Downloads\STM32CubeExpansion_Cloud_AZURE_V1.0.0\Projects\B-L475E-IOT01\Applications\Cloud\Azure\MDK-ARM\Project.uvprojx
5 C:\Users\emanuele quacchio\Desktop\STM32\GIT-CONV2\ADMC\Projects\Multi\Applications\Azure_Sns_DM\MDK-ARM\STM32F401RE-Nucleo\Project.uvprojx
6 C:\Users\emanuele quacchio\Desktop\STM32\GIT-CONV2\AzureDeviceManagementConv\Projects\Multi\Applications\Azure_Sns_DM\MDK-ARM\B-L475E-IOT01\Project.uvprojx
7 C:\Users\emanuele quacchio\Desktop\DEVCON\A\Projects\STM32L476RG-Nucleo\Applications\Azure_Sns_DM\MDK-ARM\STM32L476RG-Nucleo\Project.uvprojx
8 C:\Users\emanuele quacchio\Desktop\DEVCON\AZURE1_V3.0.1\Projects\STM32L476RG-Nucleo\Applications\Azure_Sns_DM\MDK-ARM\STM32L476RG-Nucleo\Project.uvprojx
9 C:\Users\emanuele quacchio\Desktop\STM32\GIT-CONV2\AzureDeviceManagementConv\Projects\Multi\Applications\Azure_Sns_DM\MDK-ARM\STM32L476RG-Nucleo\Project.uvprojx
10 C:\Program Files (x86)\STMicroelectronics\S2-LP_SigFox_DK_1.2.0\Projects\Projects_Cube\SigFox_Applications\SigFox_PushButton_Project\MDK-ARM\SigFox_PushButton_Project.uvprojx
Project
Project: Proj
Azure_S
User
Keil
Drive
Drive

Step 3. For the NUCLEO-F401RE platform, which does not support the firmware update-over-the-air procedure and does not require BootLoader, the firmware can be written to the microcontroller from the IDE.(For further details refer to the selected IDE documentation.)

Figure 17. Flash the recompiled solution to the STM32 microcontroller with MDK-ARM



```
/**
  ******************************************************************************
  * @file    main.c
  * @author  Central LAB
  * @version V3.0.0
  * @date    21-April-2017
  * @brief   main function
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; COPYRIGHT(c) 2017 STMicroelectronics</center></h2>
  *
  * Licensed under MCD-ST Liberty SW License Agreement V2, (the "License");
  * You may not use this file except in compliance with the License.
  * You may obtain a copy of the License at:
  *
  *        http://www.st.com/software_license_agreement_liberty_v2
  *
  * Redistribution and use in source and binary forms, with or without modification,
  * are permitted provided that the following conditions are met:
  *   1. Redistributions of source code must retain the above copyright notice,
  *      this list of conditions and the following disclaimer.
  *   2. Redistributions in binary form must reproduce the above copyright notice,
  *      this list of conditions and the following disclaimer in the documentation
  *      and/or other materials provided with the distribution.
  *   3. Neither the name of STMicroelectronics nor the names of its contributors
  *      may be used to endorse or promote products derived from this software
  *      without specific prior written permission.
  *
```

Step 4. For NUCLEO-F476RG, NUCLEO-F429ZI, B-L475E-IOT0A platforms, once the code has been recompiled, it is necessary to use provided scripts to flash the binary, as described in Section 2.6 The installation process for the firmware update over-the-air (FOTA) application.

2.7.3 Device configuration

It is possible to configure the sample application via serial terminal interface or NFC expansion board, or by modifying the source code.

The parameters to be configured are:

1. Wi-Fi access point credentials (NUCLEO-F401RE/NUCLEO-L476RG plus X-NUCLEO-IDW01A1 or B-L475E-IOT01A): SSID, password and security (WEP,WPA,WPA2)
2. Ethernet configuration (NUCLEO-F429ZI) : DHCP or static IP and gateway
3. Azure IoT Hub device connection string

After configuration, parameters are permanently stored in the Flash memory and can be overwritten or re-used as default configuration after board reset.

2.7.3.1 *Configuration via serial terminal*

After launching the application as described in the previous section, you can use a serial terminal to configure Wi-Fi/Ethernet and to enter the IoT Hub device connection string.

2.7.3.1.1 Wi-Fi access point configuration (NUCLEO-L476RG/F401RE, B-L475E-IOT01A)

Step 1. Open the serial terminal and press **Reset**.

The very first time the board is flashed and the application is launched, the default SSID and password are used as written in the code.

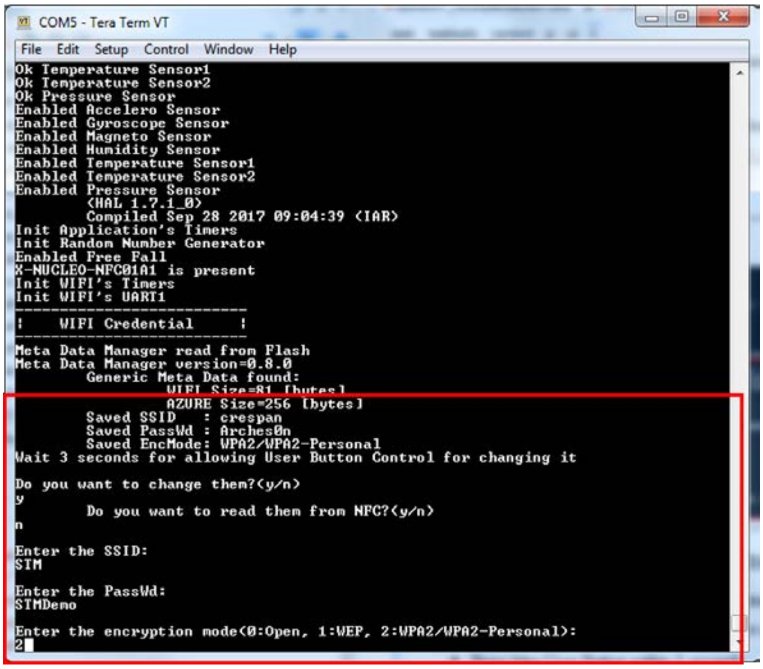
Step 2. Press the blue **User Button** within 3 seconds.

Step 3. Press **n** when asked to read from NFC

Step 4. Enter the requested parameters.

Wi-Fi credentials inserted are stored in the Flash memory and used after each board reset, unless the **User Button** is pressed.

Figure 18. **Configure Wi-Fi credentials**



```
Ok Temperature Sensor1
Ok Temperature Sensor2
Ok Pressure Sensor
Enabled Accelero Sensor
Enabled Gyroscope Sensor
Enabled Magneto Sensor
Enabled Humidity Sensor
Enabled Temperature Sensor1
Enabled Temperature Sensor2
Enabled Pressure Sensor
        <HAL 1.7.1_0>
        Compiled Sep 28 2017 09:04:39 <IAR>
Init Application's Timers
Init Random Number Generator
Enabled Free Fall
X-NUCLEO-NFC01A1 is present
Init WIFI's Timers
Init WIFI's UART1
--------------------------
!    WIFI Credential     !
--------------------------
Meta Data Manager read from Flash
Meta Data Manager version=0.8.0
        Generic Meta Data found:
                 WIFI Size=81 [bytes]
                 AZURE Size=256 [bytes]
        Saved SSID   : crespan
        Saved PassWd : Arches0n
        Saved EncMode: WPA2/WPA2-Personal
Wait 3 seconds for allowing User Button Control for changing it

Do you want to change them?(y/n)
y
        Do you want to read them from NFC?(y/n)
n

Enter the SSID:
STM

Enter the PassWd:
STMDemo

Enter the encryption mode(0:Open, 1:WEP, 2:WPA2/WPA2-Personal):
2
```

2.7.3.1.2 Ethernet configuration (NUCLEO-F429ZI)

Step 1. Open the serial terminal and press **Reset**. The very first time the board is flashed and the application is launched, the default Ethernet configuration is used (DHCP enabled)

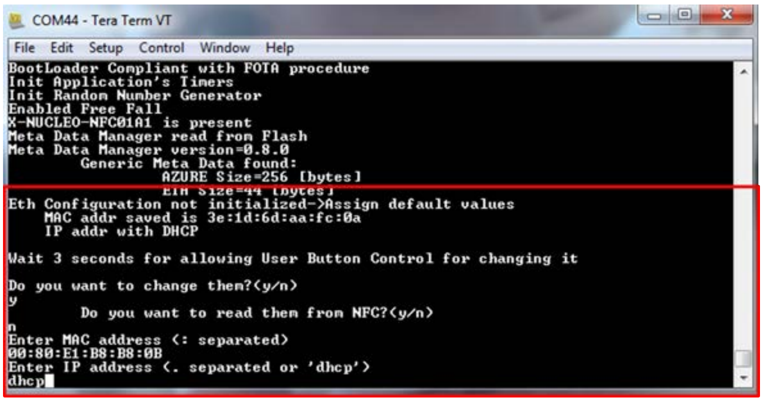
Step 2. Press the blue **User Button** within 3 seconds

Step 3. Press **n** when asked to read from NFC

Step 4. Enter the MAC address and then **dhcp** for automatic IP configuration or static IP and gateway address

Step 5. The Ethernet configuration is stored in the Flash memory and used after each board reset, unless the **User Button** is pressed.

Figure 19. Configure Ethernet credentials



```
BootLoader Compliant with FOTA procedure
Init Application's Timers
Init Random Number Generator
Enabled Free Fall
X-NUCLEO-NFC01A1 is present
Meta Data Manager read from Flash
Meta Data Manager version=0.8.0
        Generic Meta Data found:
                 AZURE Size=256 [bytes]
                 ETH Size=44 [bytes]
Eth Configuration not initialized->Assign default values
    MAC addr saved is 3e:1d:6d:aa:fc:0a
    IP addr with DHCP

Wait 3 seconds for allowing User Button Control for changing it

Do you want to change them?(y/n)
y
        Do you want to read them from NFC?(y/n)
n
Enter MAC address (: separated)
00:80:E1:B8:B8:0B
Enter IP address (. separated or 'dhcp')
dhcp
```

2.7.3.1.3 IoT Hub device connection string

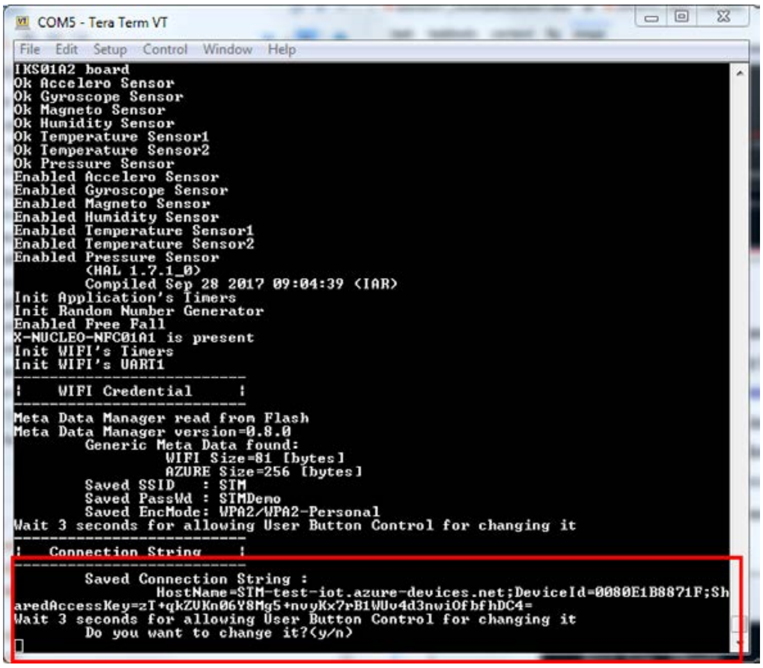
Step 1. Once completed Wi-Fi or Ethernet configuration, enter the Azure IoT Hub device connection string obtained following the procedure described in Section 2.7.1 Create Azure IoT Hub instance and generate connection string

The sample application comes with a **NULL** default value for the device connection string which has to be overwritten with a valid one the first time the binary is used.

Step 2. After board reset, press the blue **User Button** within 3 seconds to update the connection string.

The connection string is stored in the Flash memory and used after each board reset, unless the **User Button** is pressed.

Figure 20. Enter IoTHub device connection string



```
IKS01A2 board
Ok Accelero Sensor
Ok Gyroscope Sensor
Ok Magneto Sensor
Ok Humidity Sensor
Ok Temperature Sensor1
Ok Temperature Sensor2
Ok Pressure Sensor
Enabled Accelero Sensor
Enabled Gyroscope Sensor
Enabled Magneto Sensor
Enabled Humidity Sensor
Enabled Temperature Sensor1
Enabled Temperature Sensor2
Enabled Pressure Sensor
        (HAL 1.7.1_0)
        Compiled Sep 28 2017 09:04:39 (IAR)
Init Application's Timers
Init Random Number Generator
Enabled Free Fall
X-NUCLEO-NFC01A1 is present
Init WIFI's Timers
Init WIFI's UART1
--------------------------
|    WIFI Credential     |
--------------------------
Meta Data Manager read from Flash
Meta Data Manager version=0.8.0
        Generic Meta Data found:
                 WIFI Size=81 [bytes]
                 AZURE Size=256 [bytes]
        Saved SSID    : STM
        Saved PassWd  : STMDemo
        Saved EncMode: WPA2/WPA2-Personal
Wait 3 seconds for allowing User Button Control for changing it
--------------------------
|   Connection String    |
--------------------------
        Saved Connection String :
                HostName=STM-test-iot.azure-devices.net;DeviceId=0080E1B8871F;Sh
aredAccessKey=zI+qkZVKn06Y8Mg5+nvyKx7rB1WUv4d3nwiOfbfhDC4=
Wait 3 seconds for allowing User Button Control for changing it
        Do you want to change it?(y/n)
```

2.7.3.2 *Configuration via NFC*

If you are using B-L475E-IOT01A or NUCLEO-F476RG/F401RE/F429ZI with an XNUCLEO-NFC04A1 expansion board and you have an NFC-enabled mobile phone, you can use near field communication technology to configure Wi-Fi or Ethernet and enter the Azure IoT Hub connection string.

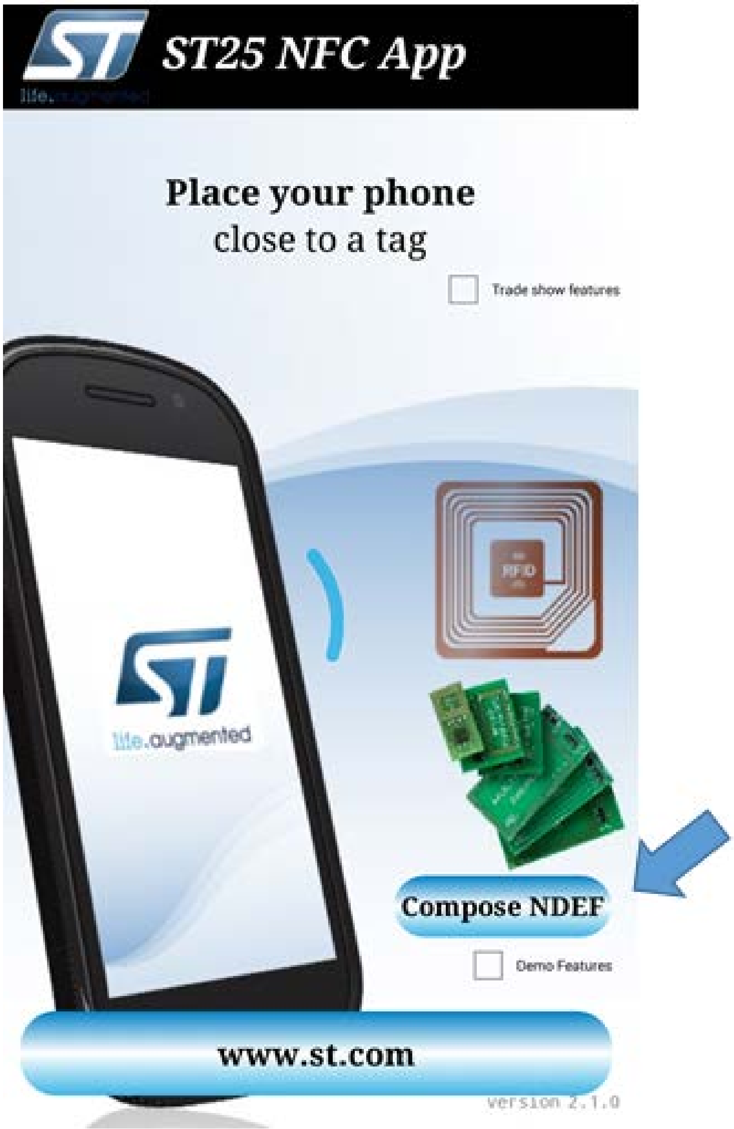
Rebuild solution files before flashing the board, or use the pre-compiled binaries, then follow instructions below according to your board configuration.

2.7.3.2.1 Wi-Fi access point configuration (NUCLEO-F401RE/NUCLEO-L476RG plus X-NUCLEO-IDW01A1, or B-L475E-IOT01A)

Step 1. Download the ST25 NFC mobile application on your NFC-enabled mobile phone: (any other mobile application able to write NDEF parameters to NFC can also be used).

Step 2. Launch your mobile application to write/read NDEF tags (i.e. ST25 NFC tag)

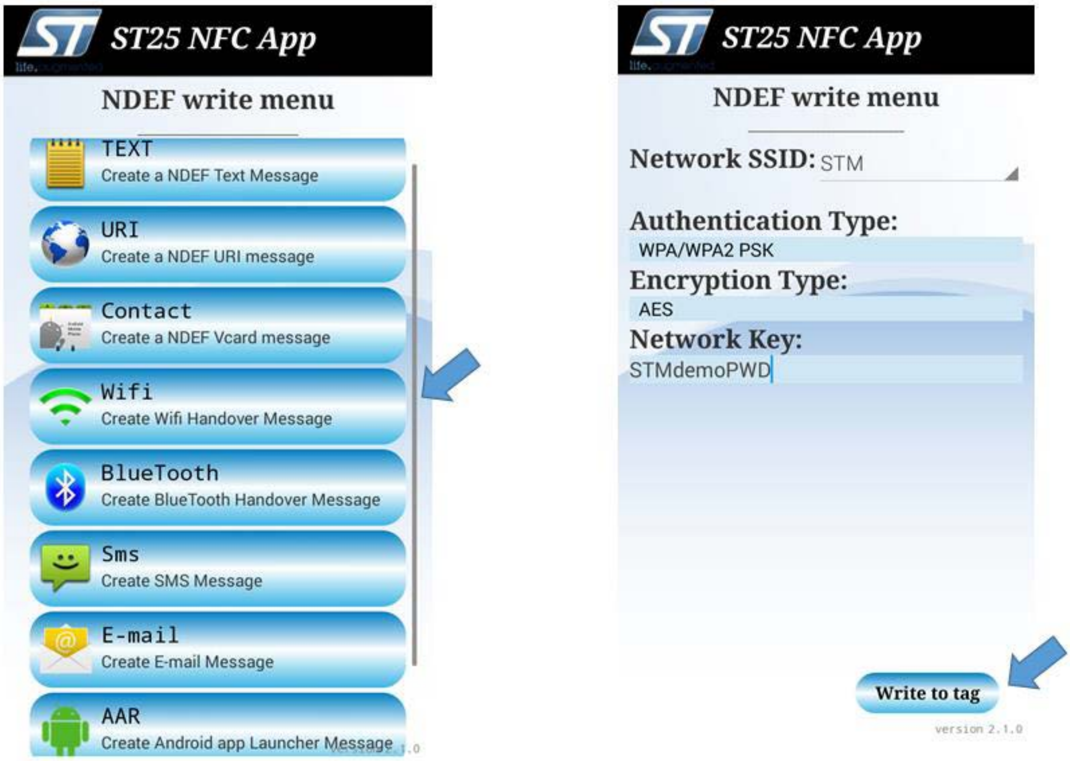
Figure 21. **ST25 NFC app**



Step 3. Click on the Wi-Fi button and enter the SSID and password for the access point.

Step 4. Then click on the **Write to tag** button after placing your mobile phone near to the NFC expansion board.

Figure 22. **AP parameter setting on the ST25 NFC app**


ST25 NFC App
NDEF write menu
TEXT
Create a NDEF Text Message
URI
Create a NDEF URI message
Contact
Create a NDEF Vcard message
Wifi
Create Wifi Handover Message
BlueTooth
Create BlueTooth Handover Message
Sms
Create SMS Message
E-mail
Create E-mail Message
AAR
Create Android app Launcher Message
ST25 NFC App
NDEF write menu
Write to tag
version 2.1.0

Network SSID: STM

Authentication Type: WPA/WPA2 PSK

Encryption Type: AES

Network Key: STMdemoPWD

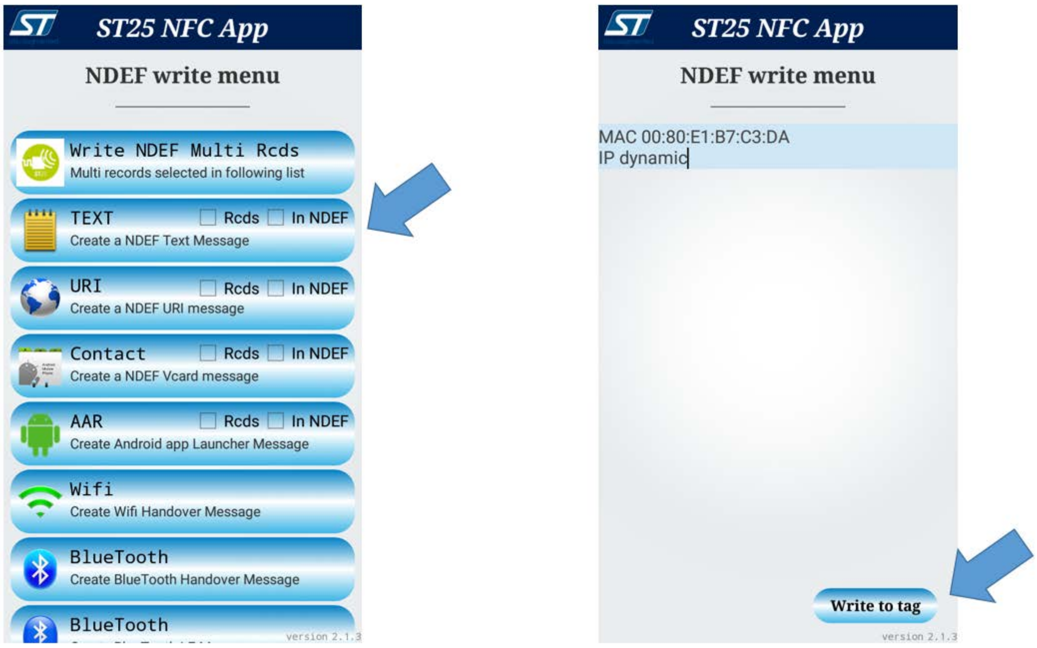
2.7.3.2.2 Ethernet configuration (NUCLEO-F429ZI)

Step 1. Click on the **Text** button and enter the MAC address as **MAC** followed by a sequence of six colon-separated hexadecimal numbers, and the IP address as **IP** followed by either 4 dot-separated decimal numbers (in case of static pre-assigned address) or **dynamic** or **dhcp** (in case of DHCP-assigned address).

In case of static IP address you have to add a further line containing **Gateway** followed by 4 dot-separated decimal numbers.

Step 2. Click on the **Write to tag** button with your mobile phone placed near the NFC expansion board.

Figure 23. AP parameters setting on ST25 NFC app



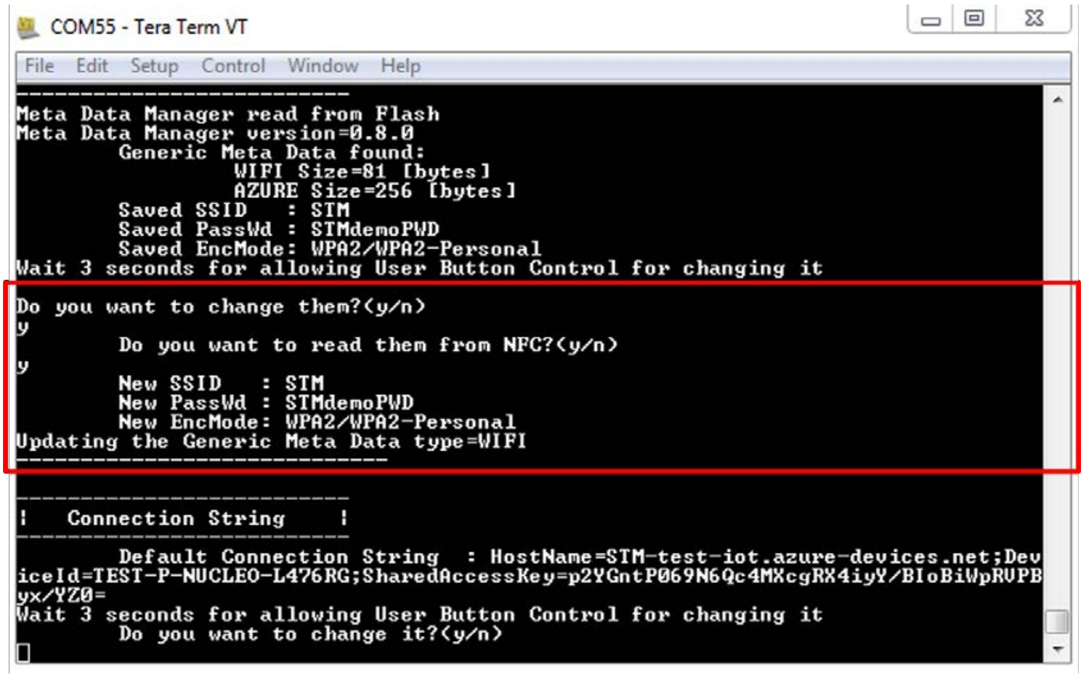
Step 3. Reset you board

Step 4. Press the blue user button within 3 seconds.

Step 5. Press y when requested to read from NFC.

The application reads the NDEF parameters from NFC and uses them for the Wi-Fi or Ethernet configuration.

Figure 24. Read Wi-Fi configuration from NFC



```
--------------------------
Meta Data Manager read from Flash
Meta Data Manager version=0.8.0
        Generic Meta Data found:
                 WIFI Size=81 [bytes]
                 AZURE Size=256 [bytes]
        Saved SSID    : STM
        Saved PassWd : STMdemoPWD
        Saved EncMode: WPA2/WPA2-Personal
Wait 3 seconds for allowing User Button Control for changing it

Do you want to change them?(y/n)
y
        Do you want to read them from NFC?(y/n)
y
        New SSID    : STM
        New PassWd : STMdemoPWD
        New EncMode: WPA2/WPA2-Personal
Updating the Generic Meta Data type=WIFI
-----------------------------

--------------------------
|   Connection String    |
--------------------------
        Default Connection String   : HostName=STM-test-iot.azure-devices.net;Dev
iceId=TEST-P-NUCLEO-L476RG;SharedAccessKey=p2YGntP069N6Qc4MXcgRX4iyY/BIoBiWpRVPB
yx/YZ0=
Wait 3 seconds for allowing User Button Control for changing it
        Do you want to change it?(y/n)
```

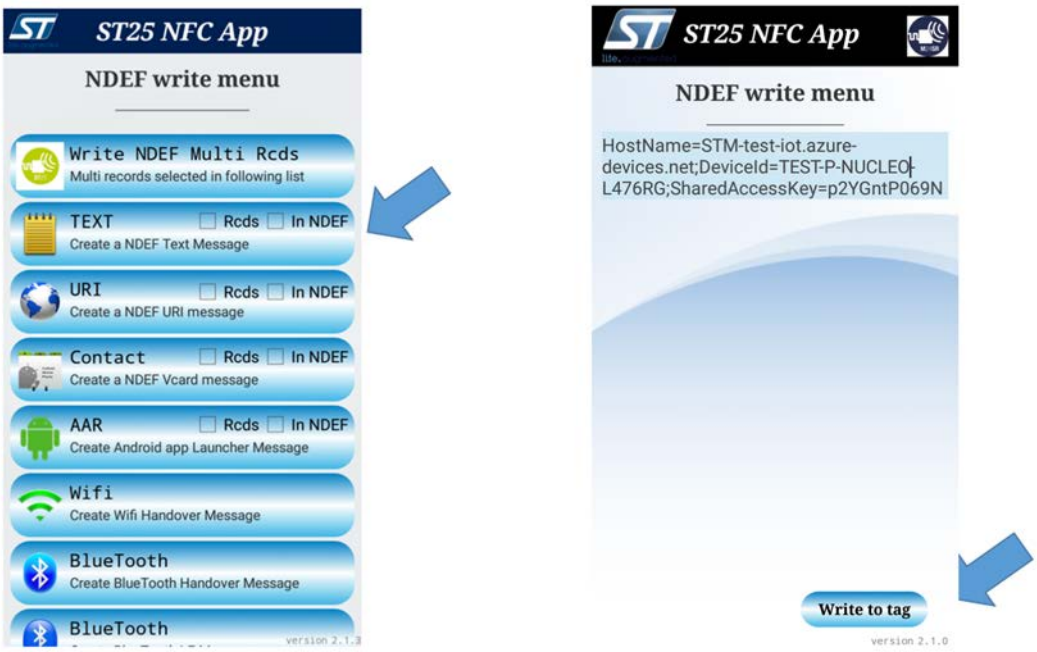
2.7.3.2.3 IoT Hub connection string update

Step 1. After Wi-Fi/Ethernet configuration, press again the blue user button within 3 seconds to update the IoT Hub device connection string.

Step 2. Go back to your mobile application, click on the **Text** button and paste the IoTHub connection string.

Step 3. Click on the **Write to tag** button after placing your mobile phone near your device NFC tag.

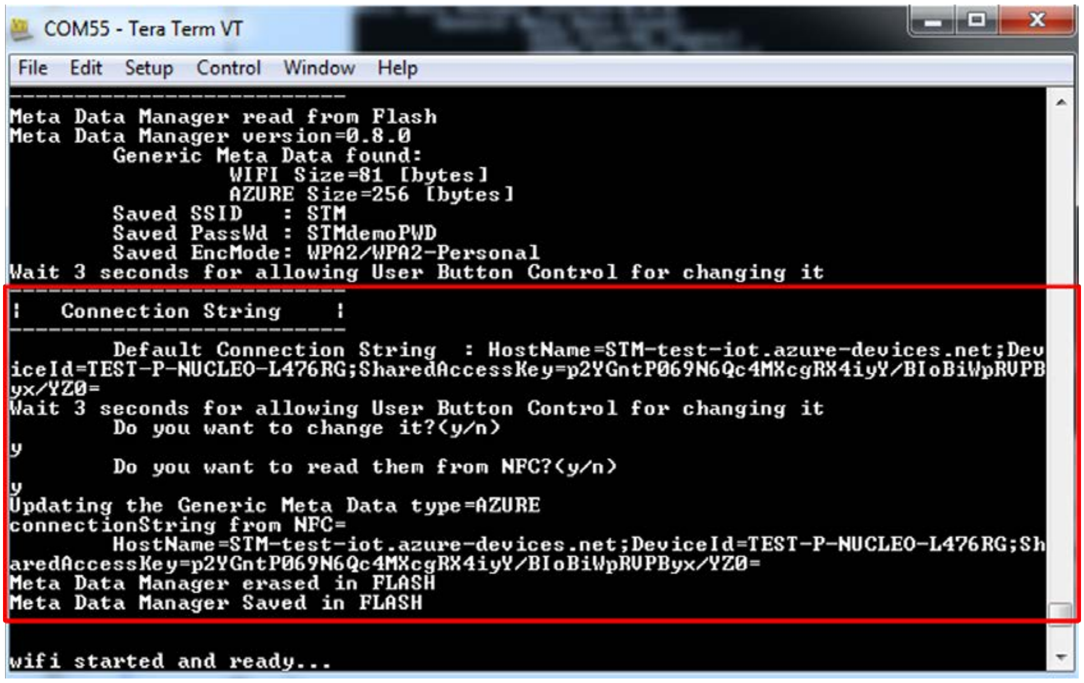
Figure 25. **Paste IoT Hub connection string in ST25 NFC App**



HostName=STM-test-iot.azure-devices.net;DeviceId=TEST-P-NUCLEO-L476RG;SharedAccessKey=p2YGntP069N

Step 4. In the serial terminal, press **y** when asked to read connection string from NFC.

Figure 26. **Read connection string from NFC**



```
--------------------------
Meta Data Manager read from Flash
Meta Data Manager version=0.8.0
        Generic Meta Data found:
                 WIFI Size=81 [bytes]
                 AZURE Size=256 [bytes]
        Saved SSID    : STM
        Saved PassWd : STMdemoPWD
        Saved EncMode: WPA2/WPA2-Personal
Wait 3 seconds for allowing User Button Control for changing it
--------------------------
|   Connection String     |
--------------------------
        Default Connection String   : HostName=STM-test-iot.azure-devices.net;Dev
iceId=TEST-P-NUCLEO-L476RG;SharedAccessKey=p2YGntP069N6Qc4MXcgRX4iyY/BIoBiWpRVPB
yx/YZ0=
Wait 3 seconds for allowing User Button Control for changing it
        Do you want to change it?(y/n)
y
        Do you want to read them from NFC?(y/n)
y
Updating the Generic Meta Data type=AZURE
connectionString from NFC=
        HostName=STM-test-iot.azure-devices.net;DeviceId=TEST-P-NUCLEO-L476RG;Sh
aredAccessKey=p2YGntP069N6Qc4MXcgRX4iyY/BIoBiWpRVPByx/YZ0=
Meta Data Manager erased in FLASH
Meta Data Manager Saved in FLASH

wifi started and ready...
```

2.7.3.3 *Configuration in source code*

Wi-Fi or Ethernet configuration and Azure IoT Hub connection string can be directly entered in the source code before recompiling the solution file, as described below.

Step 1. Open the solution file according to the selected IDE and platform used.

Step 2. For **Wi-Fi configuration** (NUCLEO-F01RE/NUCLEO-L476RG + X-NUCLEO-IDW01A1, or B-L475E-IOT01Ax), open the file *azure1_config.h* and add a custom value for *AZURE_DEFAULT_SSID*, *AZURE_DEFAULT_SECKEY*, *AZURE_DEFAULT_PRIV_MODE*.

Step 3. For **Ethernet configuration** (NUCLEO-F429ZI), open the file *platform_STM32Cube_NucleoF429ZI.c* at line 307, and set a custom IP and Gateway configuration (*IP_ADDR0*,*IP_ADDR1*,*IP_ADDR2*,*IP_ADDR3* for IP address, *GW_ADDR0*, *GW_ADDR1*, *GW_ADDR2*, *GW_ADDR3* for Gateway address), or set *EthConfiguration.use_dhcp* flag to 1 to use DHCP.

Step 4. To enter **IoTHub device connection string** open *azure1_config.h* and add a valid connection string for *AZUREDEVICECONNECTIONSTRING*

```
#define AZUREDEVICECONNECTIONSTRING
"HostName=trial.azurewebsites.net;DeviceId=Nucleo-
Trial;SharedAccessKey=XXXXXXX"
```

Step 5. Rebuild the solution file according to the selected IDE and flash the microcontroller.

For NUCLEO-L476RG, NUCLEO-F429ZI, B-L475E-IOT01Ax, follow the procedure described in Section 2.6 The installation process for the firmware update over-the-air (FOTA) application.

2.7.4 Sensor data visualization

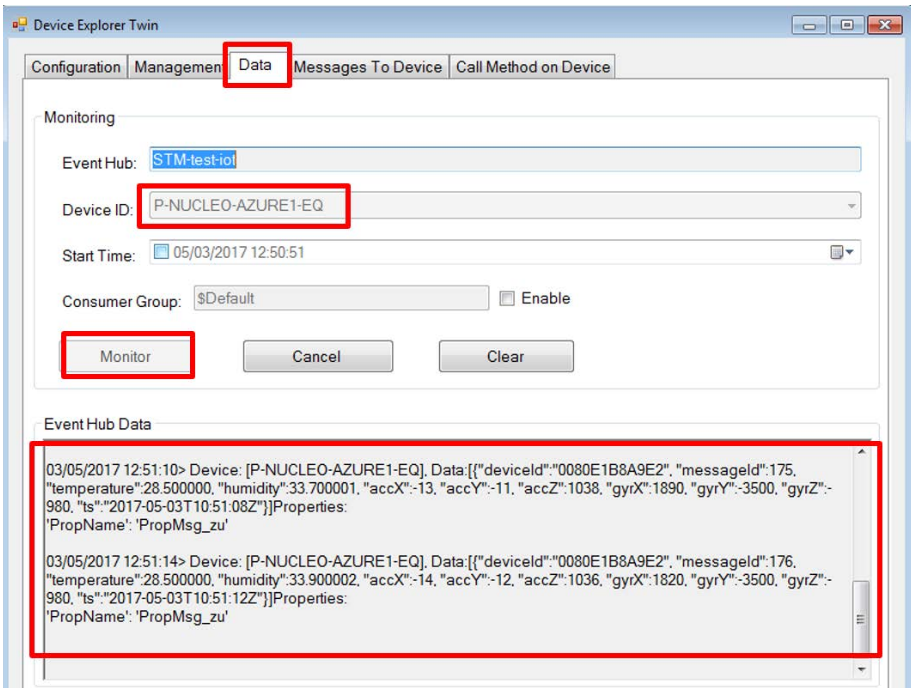
Once your board is properly configured and you have entered a valid connection string, the sample application connects the provisioned IoT Hub and starts to transmit messages containing sensor data.

To view the log of the messages received by the IoT Hub, you can use the Device Explorer.

Step 1. Verify that Device Explorer is properly configured with your IoT Hub connection string as described in Section 2.7.1 Create Azure IoT Hub instance and generate connection string.

Step 2. In the Device Explorer data tab, select your **Device ID** and press **Monitor** to view the sensor data log.

Figure 27. Sensor data in Device Explorer



Event Hub: STM-test-iot
Device ID: P-NUCLEO-AZURE1-EQ
Start Time: □ 05/03/2017 12:50:51
Consumer Group: $Default □ Enable

```
03/05/2017 12:51:10> Device: [P-NUCLEO-AZURE1-EQ], Data:[{"deviceId":"0080E1B8A9E2", "messageId":175,
"temperature":28.500000, "humidity":33.700001, "accX":-13, "accY":-11, "accZ":1038, "gyrX":1890, "gyrY":-3500, "gyrZ":-
980, "ts":"2017-05-03T10:51:08Z"}]Properties:
'PropName': 'PropMsg_zu'

03/05/2017 12:51:14> Device: [P-NUCLEO-AZURE1-EQ], Data:[{"deviceId":"0080E1B8A9E2", "messageId":176,
"temperature":28.500000, "humidity":33.900002, "accX":-14, "accY":-12, "accZ":1036, "gyrX":1820, "gyrY":-3500, "gyrZ":-
980, "ts":"2017-05-03T10:51:12Z"}]Properties:
'PropName': 'PropMsg_zu'
```

2.7.5 Control the device with Azure device management primitives

The sample application supports Azure device management primitives, which enable the device full remote control.(For further information on Azure device management primitives, refer to .)

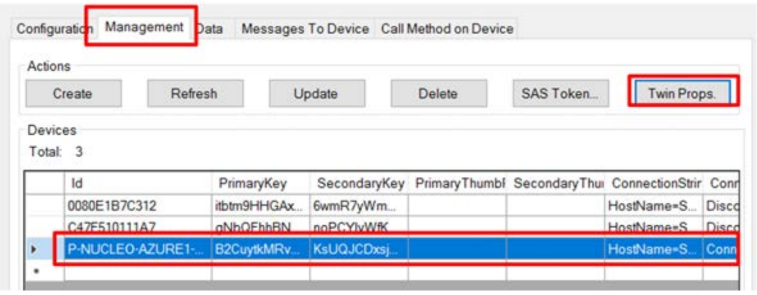
Change telemetry interval by setting the desired properties

It is possible to change the telemetry frequency (from 1 to 30 seconds) by adding one desired property in the Device Twin associated with the registered board. Desired Properties should be written in the following format:

```
{ "properties": { "desired": { "DesiredTelemetryInterval":8, }}}
```

You can modify the desired properties in the Device Explorer from the **Management** tab:

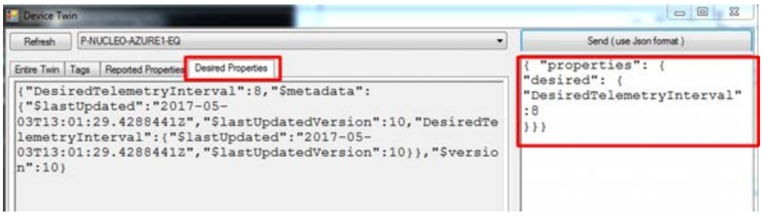
Figure 28. Device Explorer: Management tab Twin Props



Total: 3

	Id	PrimaryKey	SecondaryKey	PrimaryThumbl	SecondaryThu	ConnectionStrir	Conr
	0080E1B7C312	itbtm9HHGAx...	6wmR7yWm...			HostName=S...	Disco
	C47F510111A7	gNbQFhhBN...	noPCYlvWfK...			HostName=S...	Disco
▸	P-NUCLEO-AZURE1-...	B2CuytkMRv...	KsUQJCDxsj...			HostName=S...	Conn
*							

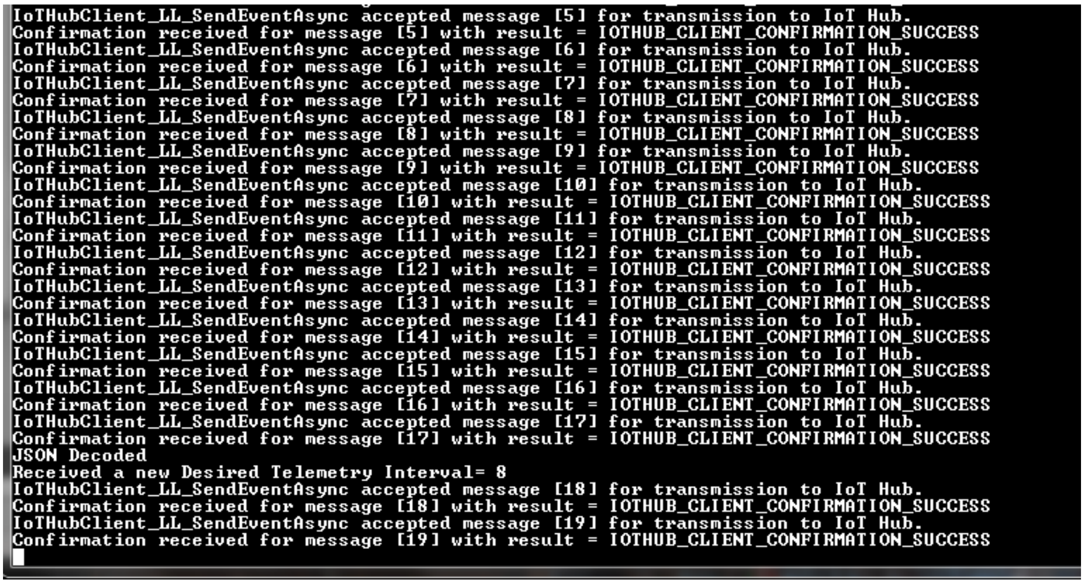
Figure 29. Device Explorer: Desired Properties tab



```
{"DesiredTelemetryInterval":8,"$metadata":
{"$lastUpdated":"2017-05-
03T13:01:29.4288441Z","$lastUpdatedVersion":10,"DesiredTe
lemetryInterval":{"$lastUpdated":"2017-05-
03T13:01:29.4288441Z","$lastUpdatedVersion":10}},"$versio
n":10}
```

```
{ "properties": {
"desired": {
"DesiredTelemetryInterval"
:8
}}}
```

Figure 30. Desired telemetry interval property change reported to device



```
IoTHubClient_LL_SendEventAsync accepted message [5] for transmission to IoT Hub.
Confirmation received for message [5] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [6] for transmission to IoT Hub.
Confirmation received for message [6] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [7] for transmission to IoT Hub.
Confirmation received for message [7] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [8] for transmission to IoT Hub.
Confirmation received for message [8] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [9] for transmission to IoT Hub.
Confirmation received for message [9] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [10] for transmission to IoT Hub.
Confirmation received for message [10] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [11] for transmission to IoT Hub.
Confirmation received for message [11] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [12] for transmission to IoT Hub.
Confirmation received for message [12] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [13] for transmission to IoT Hub.
Confirmation received for message [13] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [14] for transmission to IoT Hub.
Confirmation received for message [14] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [15] for transmission to IoT Hub.
Confirmation received for message [15] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [16] for transmission to IoT Hub.
Confirmation received for message [16] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [17] for transmission to IoT Hub.
Confirmation received for message [17] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
JSON Decoded
Received a new Desired Telemetry Interval= 8
IoTHubClient_LL_SendEventAsync accepted message [18] for transmission to IoT Hub.
Confirmation received for message [18] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [19] for transmission to IoT Hub.
Confirmation received for message [19] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
```

Send Cloud-To-Device messages

It is possible to send C2D (Cloud-To-Device) messages to the application from IoT Hub. Some C2D commands are interpreted by the embedded application:

- **Pause** pauses the application;

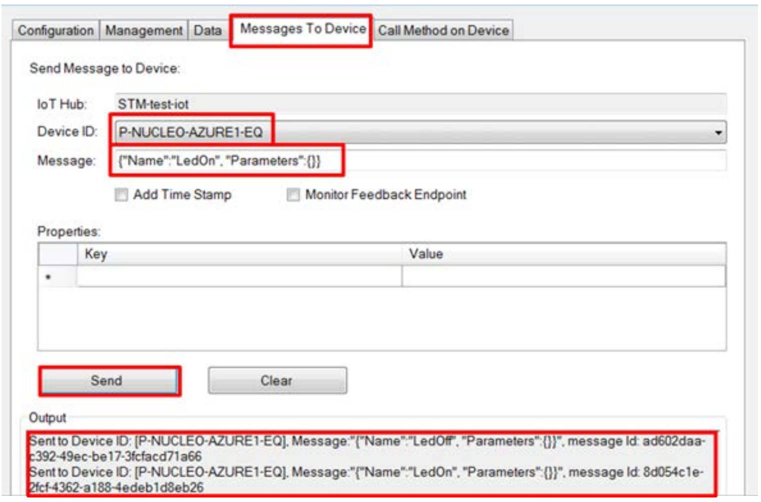
- **Play** restarts the application after the **Pause** command;
- **LedOn/LedOff** turns on/off the STM32 Nucleo on-board LED2;
- **LedBilink** STM32 Nucleo on-board LED2 blinks for each message transmitted.

The C2D messages must have the following format:

```
{"Name" : "Pause", "Parameters" : {}}
{"Name" : "Play", "Parameters" : {}}
{"Name" : "LedOn", "Parameters" : {}}
{"Name" : "LedOff", "Parameters" : {}}
{"Name" : "LedBlink", "Parameters" : {}}
```

You can send messages to your connected boards via the **Message to Device** tab in Device Explorer:

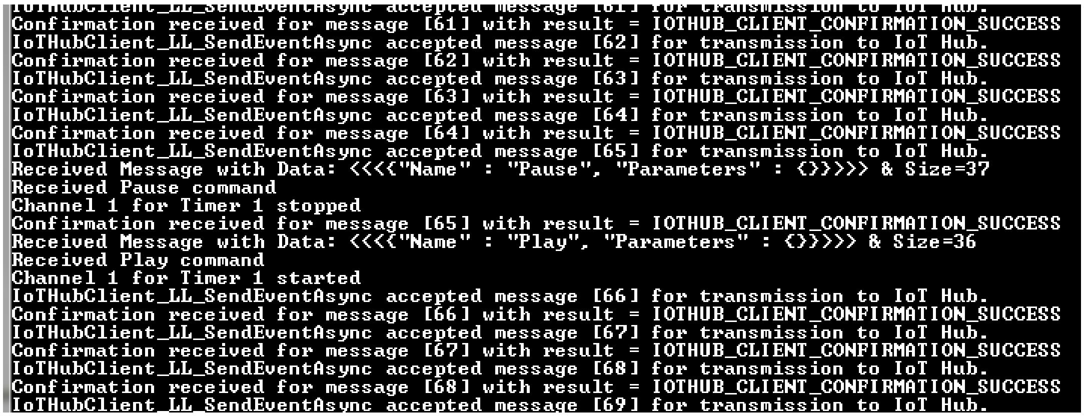
Figure 31. Sending messages to the connected board



IoT Hub: STM-test-iot
Device ID: P-NUCLEO-AZURE1-EQ
Message: {"Name":"LedOn", "Parameters":{}}
□ Add Time Stamp □ Monitor Feedback Endpoint

```
Sent to Device ID: [P-NUCLEO-AZURE1-EQ], Message:"{"Name":"LedOff", "Parameters":{}}", message Id: ad602daa-
c392-49ec-be17-3fcfacd71a66
Sent to Device ID: [P-NUCLEO-AZURE1-EQ], Message:"{"Name":"LedOn", "Parameters":{}}", message Id: 8d054c1e-
2fcf-4362-a188-4edeb1d8eb26
```

Figure 32. Cloud-To-Device messages as received by the embedded application



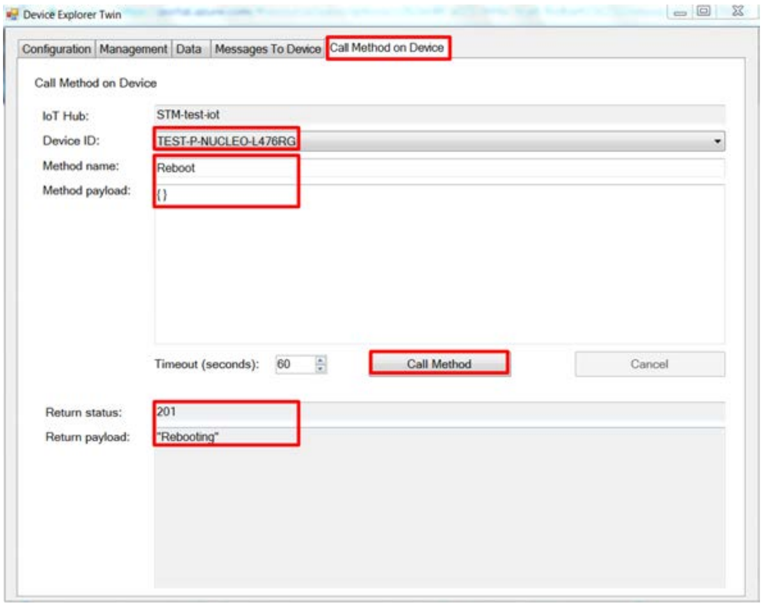
```
IoTHubClient_LL_SendEventAsync accepted message [61] for transmission to IoT Hub.
Confirmation received for message [61] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [62] for transmission to IoT Hub.
Confirmation received for message [62] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [63] for transmission to IoT Hub.
Confirmation received for message [63] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [64] for transmission to IoT Hub.
Confirmation received for message [64] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [65] for transmission to IoT Hub.
Received Message with Data: <<<{"Name" : "Pause", "Parameters" : {}}>>> & Size=37
Received Pause command
Channel 1 for Timer 1 stopped
Confirmation received for message [65] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
Received Message with Data: <<<{"Name" : "Play", "Parameters" : {}}>>> & Size=36
Received Play command
Channel 1 for Timer 1 started
IoTHubClient_LL_SendEventAsync accepted message [66] for transmission to IoT Hub.
Confirmation received for message [66] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [67] for transmission to IoT Hub.
Confirmation received for message [67] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [68] for transmission to IoT Hub.
Confirmation received for message [68] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [69] for transmission to IoT Hub.
```

Send Direct Methods. The embedded applicaton supports three different Direct Methods:

- **Reboot** reboots the system;
- **Quit** stops the application;
- **FirmwareUpdate** triggers the Firmware-Over-The-Air update (see Section 2.7.6 Firmware-over-the-air (FOTA) update sample application).

You can select **Direct Methods** using the **Call Method** on the Device tab in the Device Explorer:

Figure 33. Call Method on the device using Device Explorer



IoT Hub: STM-test-iot

Device ID: TEST-P-NUCLEO-L476RG

Method name: Reboot

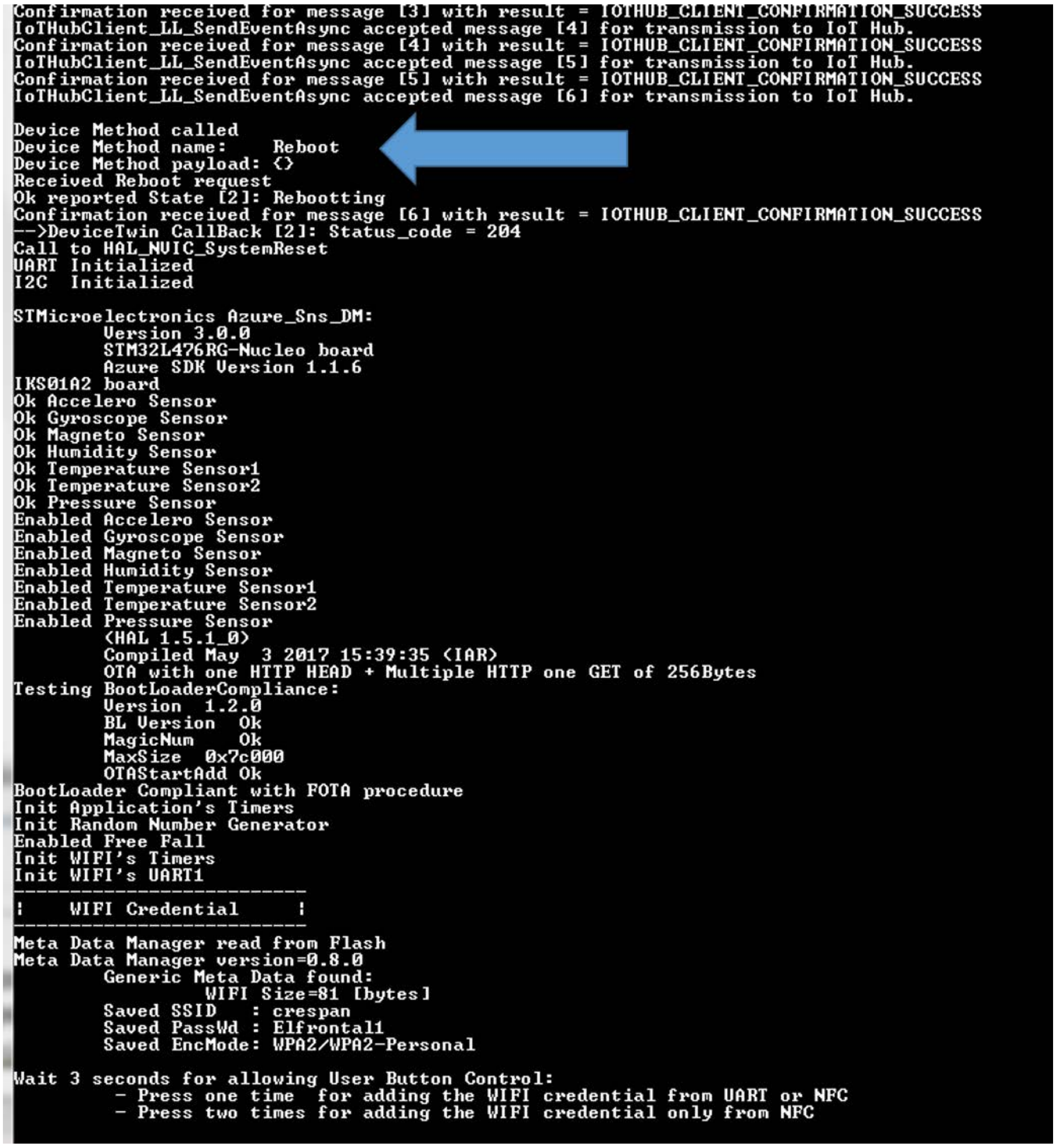
Method payload: { }

Timeout (seconds): 60

Return status: 201

Return payload: "Rebooting"

Figure 34. Direct Method executed by the device



```
Confirmation received for message [3] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [4] for transmission to IoT Hub.
Confirmation received for message [4] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [5] for transmission to IoT Hub.
Confirmation received for message [5] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [6] for transmission to IoT Hub.

Device Method called
Device Method name:    Reboot
Device Method payload: {}
Received Reboot request
Ok reported State [2]: Rebootting
Confirmation received for message [6] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
-->DeviceTwin CallBack [2]: Status_code = 204
Call to HAL_NVIC_SystemReset
UART Initialized
I2C  Initialized

STMicroelectronics Azure_Sns_DM:
        Version 3.0.0
        STM32L476RG-Nucleo board
        Azure SDK Version 1.1.6
IKS01A2 board
Ok Accelero Sensor
Ok Gyroscope Sensor
Ok Magneto Sensor
Ok Humidity Sensor
Ok Temperature Sensor1
Ok Temperature Sensor2
Ok Pressure Sensor
Enabled Accelero Sensor
Enabled Gyroscope Sensor
Enabled Magneto Sensor
Enabled Humidity Sensor
Enabled Temperature Sensor1
Enabled Temperature Sensor2
Enabled Pressure Sensor
        (HAL 1.5.1_0)
        Compiled May  3 2017 15:39:35 (IAR)
        OTA with one HTTP HEAD + Multiple HTTP one GET of 256Bytes
Testing BootLoaderCompliance:
        Version  1.2.0
        BL Version   Ok
        MagicNum     Ok
        MaxSize  0x7c000
        OTAStartAdd Ok
BootLoader Compliant with FOTA procedure
Init Application's Timers
Init Random Number Generator
Enabled Free Fall
Init WIFI's Timers
Init WIFI's UART1
--------------------------
|    WIFI Credential     |
--------------------------
Meta Data Manager read from Flash
Meta Data Manager version=0.8.0
        Generic Meta Data found:
                 WIFI Size=81 [bytes]
        Saved SSID    : crespan
        Saved PassWd  : Elfrontal1
        Saved EncMode: WPA2/WPA2-Personal

Wait 3 seconds for allowing User Button Control:
         - Press one time  for adding the WIFI credential from UART or NFC
         - Press two times for adding the WIFI credential only from NFC
```

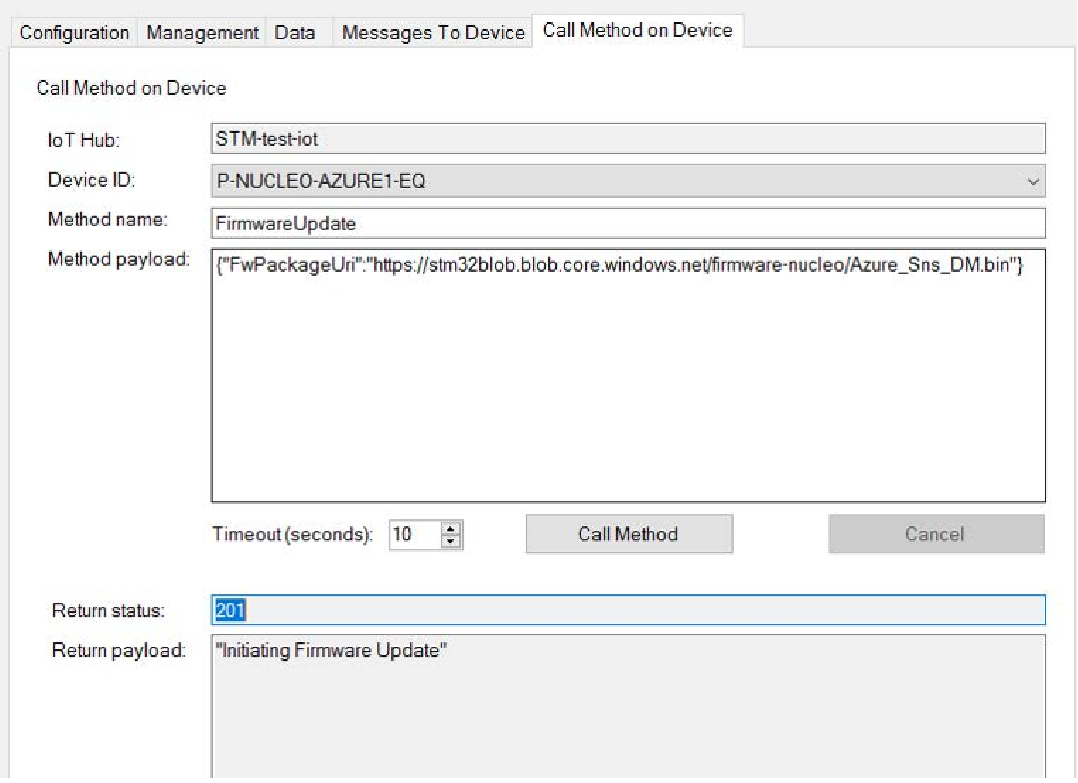
2.7.6 Firmware-over-the-air (FOTA) update sample application

For NUCLEO-L476RG/NUCLEO-F429ZI and B-L475E-IOT01A boards, Azure_Sns_DM application also includes an example of firmware update over-the-air (FOTA) procedure.

It is possible to trigger the FOTA procedure using the Device Explorer:

1. go to the **Call Method on Device** tab;
2. write the name of the Method **FirmwareUpdate**;
3. write the correct json payload containing the URI to download the new binary.

Figure 35. **How to call the FirmwareUpdate Direct Method**



IoT Hub: STM-test-iot

Device ID: P-NUCLEO-AZURE1-EQ

Method name: FirmwareUpdate

Method payload: {"FwPackageUri":"https://stm32blob.blob.core.windows.net/firmware-nucleo/Azure_Sns_DM.bin"}

Timeout (seconds): 10

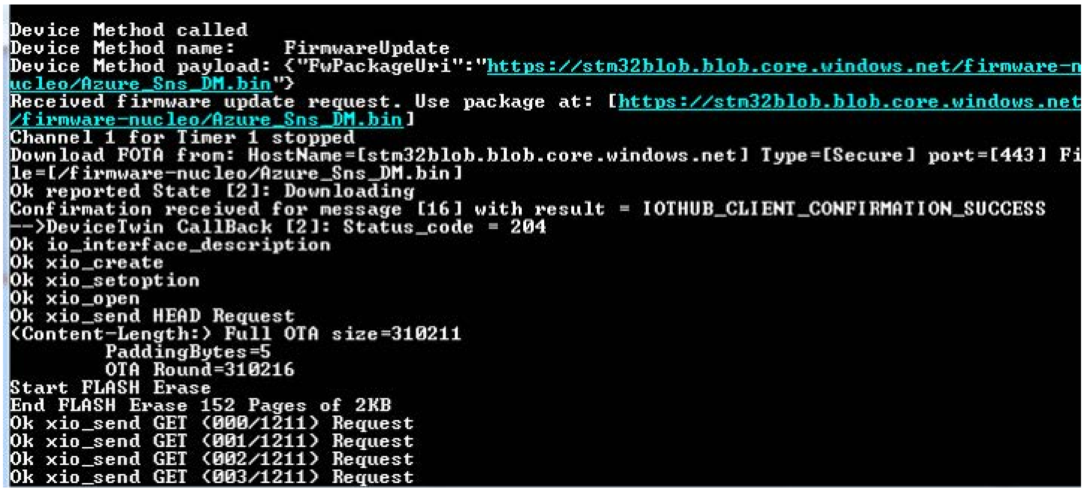
Return status: 201

Return payload: "Initiating Firmware Update"

Once the instruction from DeviceExplorer has been received, the application interrupts its current execution and download the new firmware.

When the download is completed, the board is reset and the new firmware is installed.

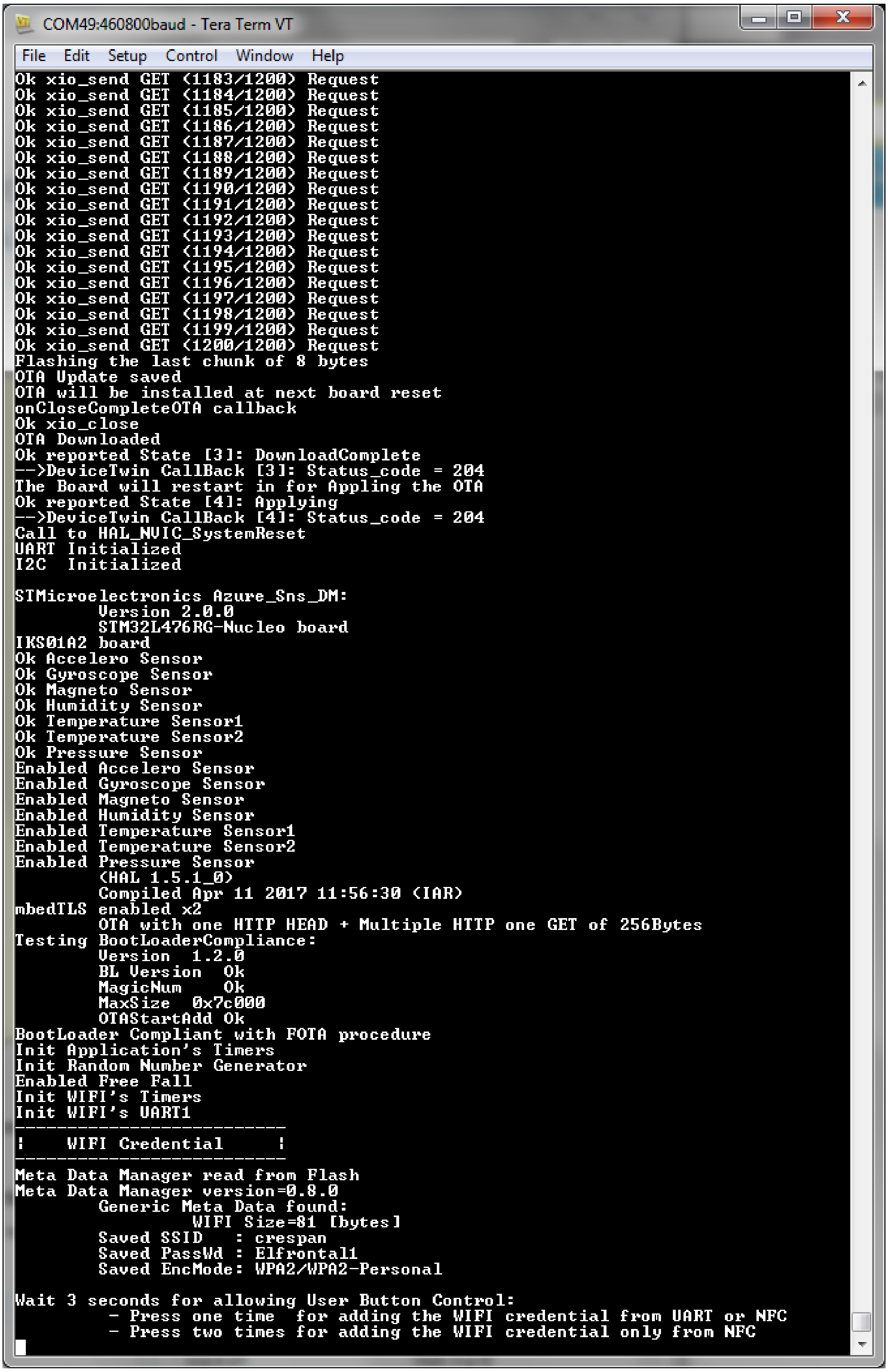
Figure 36. **Downloading the FOTA**



```
Device Method called
Device Method name:    FirmwareUpdate
Device Method payload: {"FwPackageUri":"https://stm32blob.blob.core.windows.net/firmware-n
ucleo/Azure_Sns_DM.bin"}
Received firmware update request. Use package at: [https://stm32blob.blob.core.windows.net
/firmware-nucleo/Azure_Sns_DM.bin]
Channel 1 for Timer 1 stopped
Download FOTA from: HostName=[stm32blob.blob.core.windows.net] Type=[Secure] port=[443] Fi
le=[/firmware-nucleo/Azure_Sns_DM.bin]
Ok reported State [2]: Downloading
Confirmation received for message [16] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
-->DeviceTwin CallBack [2]: Status_code = 204
Ok io_interface_description
Ok xio_create
Ok xio_setoption
Ok xio_open
Ok xio_send HEAD Request
<Content-Length:> Full OTA size=310211
        PaddingBytes=5
        OTA Round=310216
Start FLASH Erase
End FLASH Erase 152 Pages of 2KB
Ok xio_send GET (000/1211) Request
Ok xio_send GET (001/1211) Request
Ok xio_send GET (002/1211) Request
Ok xio_send GET (003/1211) Request
```

While downloading and executing the installation of the new firmware, the application constantly reports its status to the remote application. You can monitor the device status by checking **Twin** in the Device Explorer.

Figure 37. Board restart after the FOTA


COM49:460800baud - Tera Term VT
File Edit Setup Control Window Help

```
Ok xio_send GET (1183/1200) Request
Ok xio_send GET (1184/1200) Request
Ok xio_send GET (1185/1200) Request
Ok xio_send GET (1186/1200) Request
Ok xio_send GET (1187/1200) Request
Ok xio_send GET (1188/1200) Request
Ok xio_send GET (1189/1200) Request
Ok xio_send GET (1190/1200) Request
Ok xio_send GET (1191/1200) Request
Ok xio_send GET (1192/1200) Request
Ok xio_send GET (1193/1200) Request
Ok xio_send GET (1194/1200) Request
Ok xio_send GET (1195/1200) Request
Ok xio_send GET (1196/1200) Request
Ok xio_send GET (1197/1200) Request
Ok xio_send GET (1198/1200) Request
Ok xio_send GET (1199/1200) Request
Ok xio_send GET (1200/1200) Request
Flashing the last chunk of 8 bytes
OTA Update saved
OTA will be installed at next board reset
onCloseCompleteOTA callback
Ok xio_close
OTA Downloaded
Ok reported State [3]: DownloadComplete
-->DeviceTwin CallBack [3]: Status_code = 204
The Board will restart in for Appling the OTA
Ok reported State [4]: Applying
-->DeviceTwin CallBack [4]: Status_code = 204
Call to HAL_NVIC_SystemReset
UART Initialized
I2C  Initialized

STMicroelectronics Azure_Sns_DM:
        Version 2.0.0
        STM32L476RG-Nucleo board
IKS01A2 board
Ok Accelero Sensor
Ok Gyroscope Sensor
Ok Magneto Sensor
Ok Humidity Sensor
Ok Temperature Sensor1
Ok Temperature Sensor2
Ok Pressure Sensor
Enabled Accelero Sensor
Enabled Gyroscope Sensor
Enabled Magneto Sensor
Enabled Humidity Sensor
Enabled Temperature Sensor1
Enabled Temperature Sensor2
Enabled Pressure Sensor
        (HAL 1.5.1_0)
        Compiled Apr 11 2017 11:56:30 (IAR)
mbedTLS enabled x2
        OTA with one HTTP HEAD + Multiple HTTP one GET of 256Bytes
Testing BootLoaderCompliance:
        Version  1.2.0
        BL Version  Ok
        MagicNum    Ok
        MaxSize  0x7c000
        OTAStartAdd Ok
BootLoader Compliant with FOTA procedure
Init Application's Timers
Init Random Number Generator
Enabled Free Fall
Init WIFI's Timers
Init WIFI's UART1
--------------------------
|    WIFI Credential     |
--------------------------
Meta Data Manager read from Flash
Meta Data Manager version=0.8.0
        Generic Meta Data found:
                 WIFI Size=81 [bytes]
        Saved SSID   : crespan
        Saved PassWd : Elfrontal1
        Saved EncMode: WPA2/WPA2-Personal

Wait 3 seconds for allowing User Button Control:
         - Press one time  for adding the WIFI credential from UART or NFC
         - Press two times for adding the WIFI credential only from NFC
```

2.7.7 Motor control sample application

For NUCLEO-L476RG and B-L475E-IOT01A boards, **Azure_Motor** application enables to remotely control two axis stepper motor driver through X-NUCLEO-IHM02A1 expansion board.

To remotely control the motor, the following cloud-to-device commands are supported by the firmware:

- **MoveMotor [MotorNum][Angle]**: moves the motor number [0 or 1] of a specific angle [0°-360°];
- **RunMotor [MotorNum][Speed]**: starts the motor number [0 or 1] with a specific speed [1 - 10];
- **ResetMotor [MotorNum]**: resets the position for motor number [0 or 1];
- **GoHomeMotor [MotorNum]**: moves the motor number [0 or 1] to home position;
- **ComplexMove [ComplexProgramString]**: sends a string containing combination of consecutives commands to the motors.

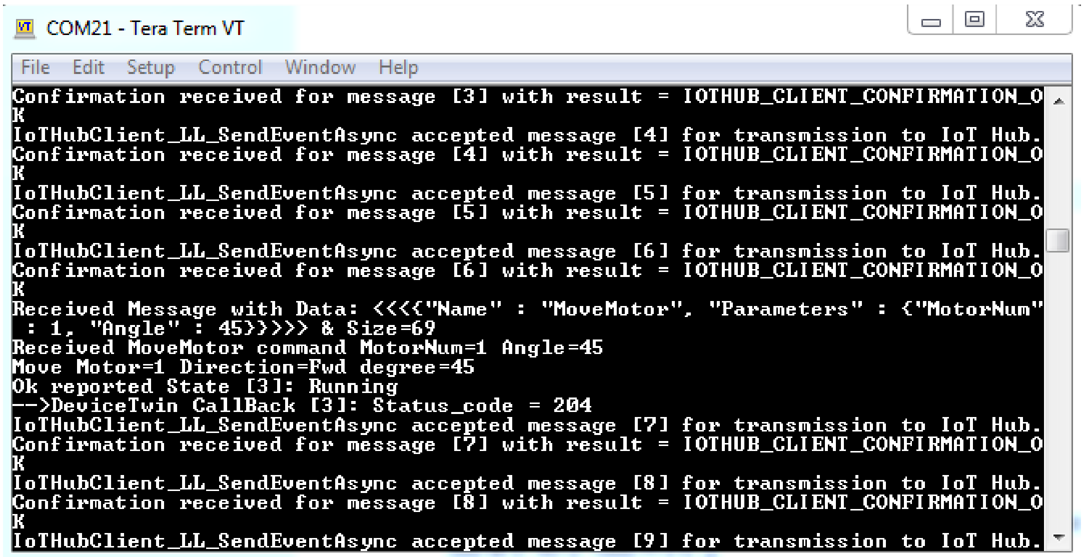
To learn more on the syntax for motor control commands, see the UM1963 at X-CUBE-SPN2.

The C2D messages for motor control can be invoked using DeviceExplorer or via Azure portal and must have the following format:

```
{"Name" : "MoveMotor", "Parameters" : {"MotorNum" : 1, "Angle" : 45}}
{"Name" : "RunMotor", "Parameters" : {"MotorNum" : 1, "Speed" : 1}}
{"Name" : "ResetMotor", "Parameters" : {"MotorNum" : 1}}
{"Name" : "GoHomeMotor", "Parameters" : {"MotorNum" : 1}}
{"Name" : "ComplexMove", "Parameters" : {"ComplexProgram" : "M 0 45 D 1000 H 0 E"}
}
```

Once received, the command is executed by the device, and a message appears in the serial terminal:

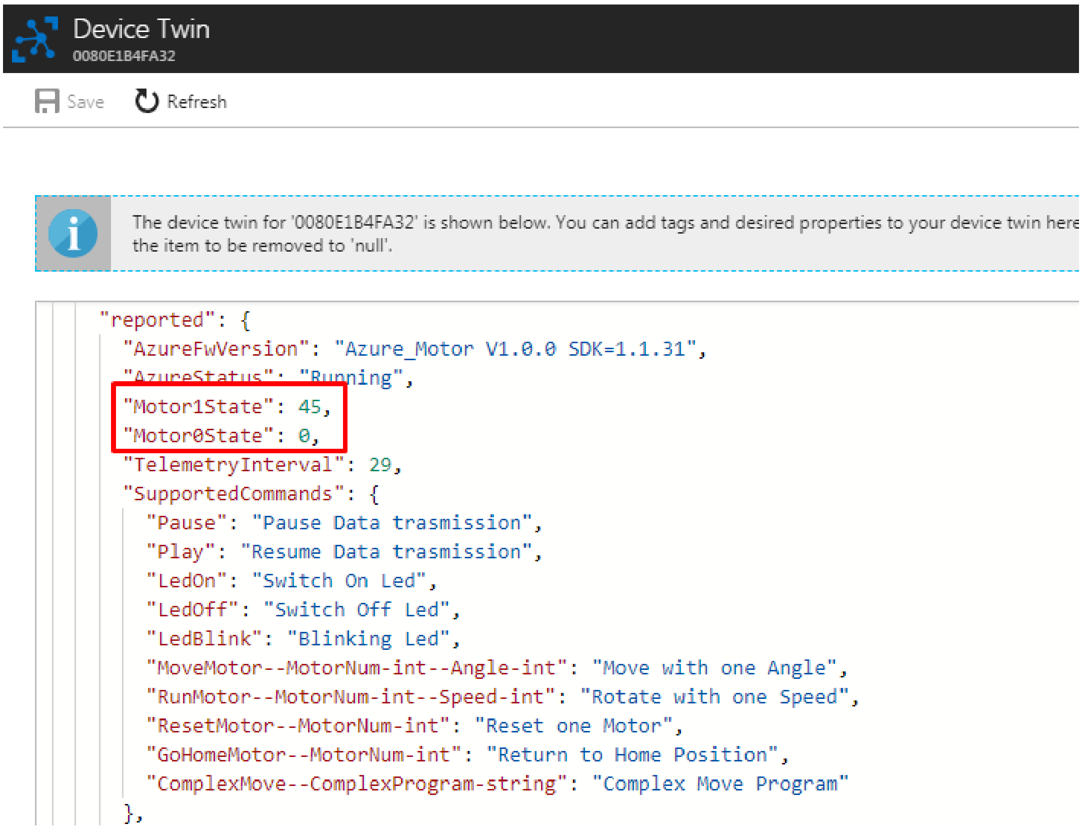
Figure 38. C2D message for motor control received by the device



```
Confirmation received for message [3] with result = IOTHUB_CLIENT_CONFIRMATION_O
K
IoTHubClient_LL_SendEventAsync accepted message [4] for transmission to IoT Hub.
Confirmation received for message [4] with result = IOTHUB_CLIENT_CONFIRMATION_O
K
IoTHubClient_LL_SendEventAsync accepted message [5] for transmission to IoT Hub.
Confirmation received for message [5] with result = IOTHUB_CLIENT_CONFIRMATION_O
K
IoTHubClient_LL_SendEventAsync accepted message [6] for transmission to IoT Hub.
Confirmation received for message [6] with result = IOTHUB_CLIENT_CONFIRMATION_O
K
Received Message with Data: <<<{"Name" : "MoveMotor", "Parameters" : {"MotorNum"
 : 1, "Angle" : 45}}>>> & Size=69
Received MoveMotor command MotorNum=1 Angle=45
Move Motor=1 Direction=Fwd degree=45
Ok reported State [3]: Running
-->DeviceTwin CallBack [3]: Status_code = 204
IoTHubClient_LL_SendEventAsync accepted message [7] for transmission to IoT Hub.
Confirmation received for message [7] with result = IOTHUB_CLIENT_CONFIRMATION_O
K
IoTHubClient_LL_SendEventAsync accepted message [8] for transmission to IoT Hub.
Confirmation received for message [8] with result = IOTHUB_CLIENT_CONFIRMATION_O
K
IoTHubClient_LL_SendEventAsync accepted message [9] for transmission to IoT Hub.
```

The firmware application updates the Device Twin with information about the new position of the motor.

Figure 39. New motor position reported in Device Twin



The device twin for '0080E1B4FA32' is shown below. You can add tags and desired properties to your device twin here
the item to be removed to 'null'.

```
"reported": {
  "AzureFwVersion": "Azure_Motor V1.0.0 SDK=1.1.31",
  "AzureStatus": "Running",
  "Motor1State": 45,
  "Motor0State": 0,
  "TelemetryInterval": 29,
  "SupportedCommands": {
    "Pause": "Pause Data trasmission",
    "Play": "Resume Data trasmission",
    "LedOn": "Switch On Led",
    "LedOff": "Switch Off Led",
    "LedBlink": "Blinking Led",
    "MoveMotor--MotorNum-int--Angle-int": "Move with one Angle",
    "RunMotor--MotorNum-int--Speed-int": "Rotate with one Speed",
    "ResetMotor--MotorNum-int": "Reset one Motor",
    "GoHomeMotor--MotorNum-int": "Return to Home Position",
    "ComplexMove--ComplexProgram-string": "Complex Move Program"
  },
```

2.7.8 IoT Central sample for Azure_Sns_DM application

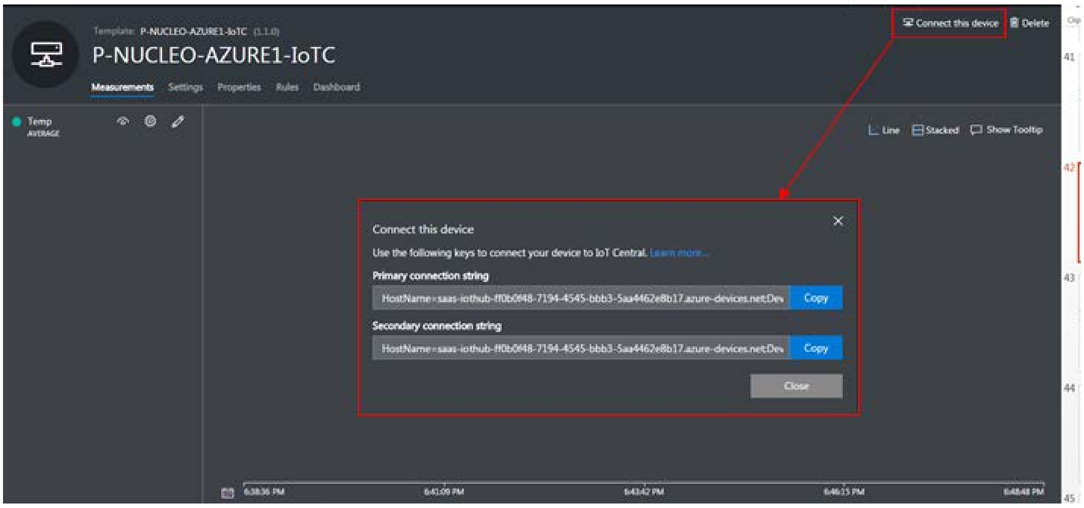
For NUCLEO-L476RG and B-L475E-IOT01Ax, **Azure_Sns_DM** includes a configuration which enables connection to Microsoft IoT Central (https://www.microsoft.com/en-us/iot-central).

Microsoft IoT Central is a fully managed SaaS (software-as-a-service) based on Azure that simplifies the development of IoT solutions. Hands-on material to create a simple application in IoT Central is available at https://docs.microsoft.com/en-gb/microsoft-iot-central/tutorial-add-device.

To enable communication with IoT Central, open *azure1_config.h* then uncomment define *AZURE_IOT_CENTRAL*; in alternative, the pre-compiled binary *Azure_Sns_DM_IoTC.bin* can be used.

Once created the application in IoT Central, you can connect a device by simply clicking on *Connect this device* and then copying the resulting connection string.

Figure 40. **Retrieve Azure IoT Central connection string**



Connect this device

Use the following keys to connect your device to IoT Central. Learn more...

Primary connection string

HostName=saas-iothub-ff0b0f48-7194-4545-bbb3-5aa4462e8b17.azure-devices.net;Dev

Copy

Secondary connection string

HostName=saas-iothub-ff0b0f48-7194-4545-bbb3-5aa4462e8b17.azure-devices.net;Dev

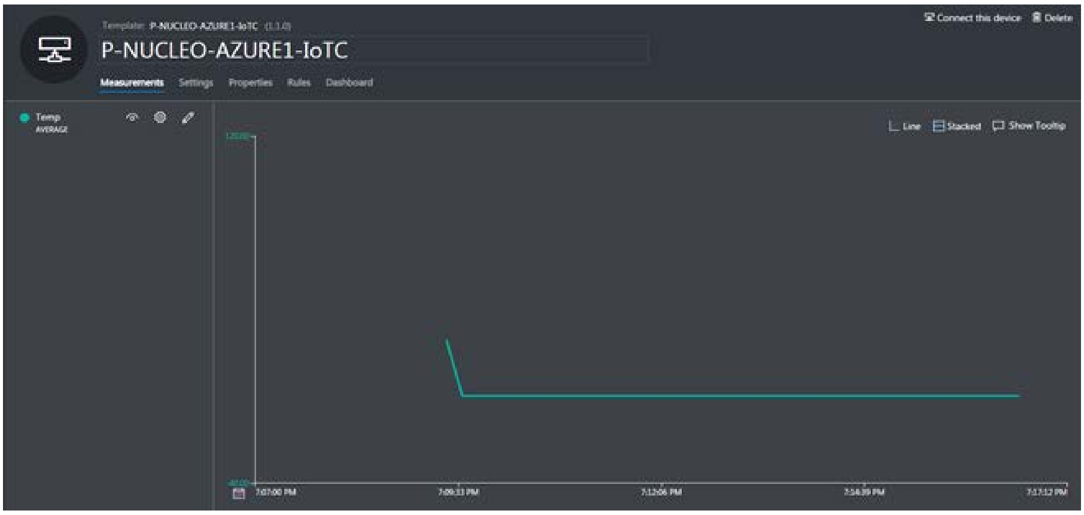
Copy

Close

IoT Central connection string can be provisioned to the device via serial interface or NFC, similarly to any other IoT Hub connection string. After that, the device connects to your IoT Central application in your IoT Central account.

Similarly to the usage of Azure IoT Hub, the firmware application starts to transmit sensor data and device properties which can be visualized in IoT Central dashboard.

Figure 41. **IoT Central dashboard**



2.8 STM32 ODE web dashboard for FP-CLD-AZURE1

A web dashboard based on Microsoft Azure has been created to offer developers a quickstart evaluation of features available in FP-CLD-AZURE1.

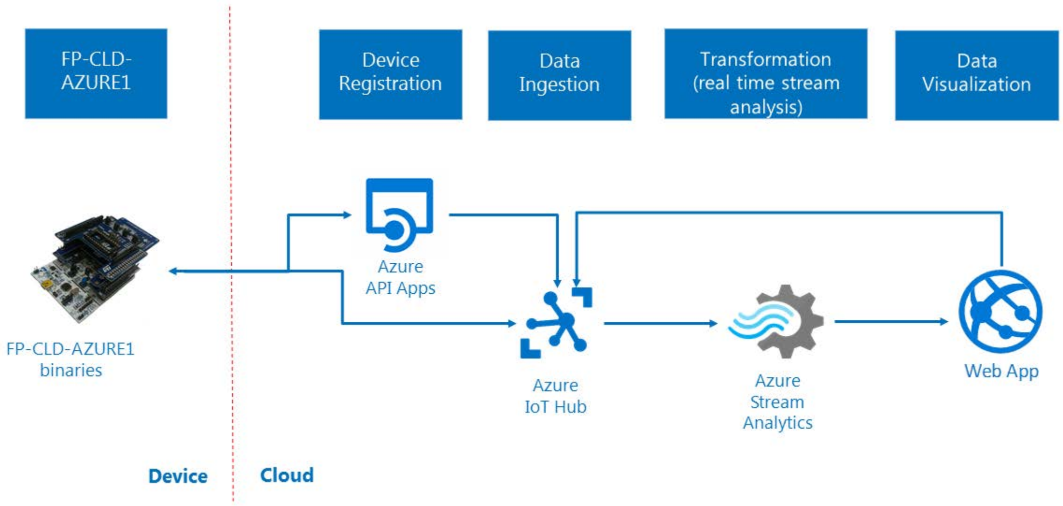
Specific ready-to-use pre-compiled binaries are provided in the FP-CLD-AZURE1 package to connect your boards with the web dashboard: *Azure_Sns_DM_BL_Web.bin* (for NUCLEO-L476RG, B-L475E-IOT01A, NUCLEO-F429ZI), *Azure_Motor_Web* (for NUCLEO-L476RG, B-L475E-IOT01A when connected to X-NUCLEO-IHM02A1), and *Azure_Sns_Web.bin* (for NUCLEO-F01RE).

You can access STM32 ODE web dashboard at .

2.8.1 Overview of Microsoft Azure services

The web dashboard implements the Microsoft Azure services detailed below.

Figure 42. **Overview of Azure services used**



- Azure API Apps: to automate board registration to a predefined IoT Hub (azure.microsoft.com/en-us/services/app-service/api/).
- Azure IoT Hub: created to collect sensor data from connected boards when FP-CLD-AZURE1 binaries are used (azure.microsoft.com/en-us/services/iot-hub/).
- Azure Stream Analytics: to analyze (e.g. compare sensor data with threshold values) real-time sensor data received through IoT Hub (azure.microsoft.com/en-us/services/stream-analytics/).
- Azure Web Apps: developed using Azure Web Apps to visualize real-time sensor data and to remotely control the connected device (azure.microsoft.com/en-us/services/app-service/web/).

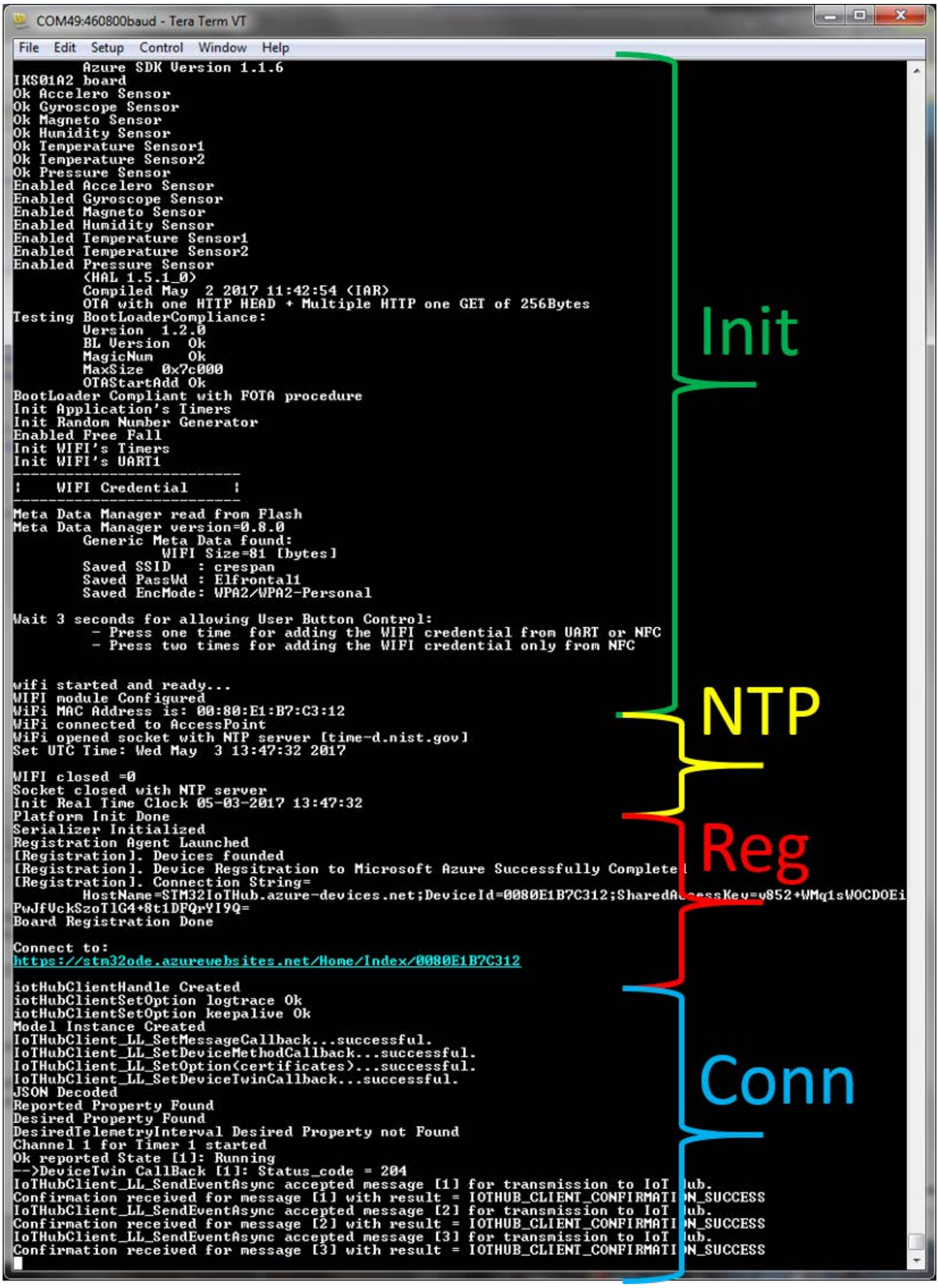
2.8.2 Automatic device registration to IoT Hub

The pre-compiled binaries include a procedure to automate the registration of your device to IoT Hub.

The MAC address of the Wi-Fi or Ethernet is used to register and identify your device in the IoT Hub and can be read from the serial terminal.

After the initialization phase, the application contacts the registration service and retrieves the connection string (see *Reg*. section in picture below). The complete URL to open the web dashboard is also printed over the serial interface.

Figure 43. **Messages printed over serial terminal for device registration**


COM49:460800baud - Tera Term VT
File Edit Setup Control Window Help
Init
NTP
Reg
Conn

```
        Azure SDK Version 1.1.6
IKS01A2 board
Ok Accelero Sensor
Ok Gyroscope Sensor
Ok Magneto Sensor
Ok Humidity Sensor
Ok Temperature Sensor1
Ok Temperature Sensor2
Ok Pressure Sensor
Enabled Accelero Sensor
Enabled Gyroscope Sensor
Enabled Magneto Sensor
Enabled Humidity Sensor
Enabled Temperature Sensor1
Enabled Temperature Sensor2
Enabled Pressure Sensor
        (HAL 1.5.1_0)
        Compiled May  2 2017 11:42:54 (IAR)
        OTA with one HTTP HEAD + Multiple HTTP one GET of 256Bytes
Testing BootLoaderCompliance:
        Version  1.2.0
        BL Version  Ok
        MagicNum    Ok
        MaxSize  0x7c000
        OTAStartAdd Ok
BootLoader Compliant with FOTA procedure
Init Application's Timers
Init Random Number Generator
Enabled Free Fall
Init WIFI's Timers
Init WIFI's UART1
--------------------------
|    WIFI Credential     |
--------------------------
Meta Data Manager read from Flash
Meta Data Manager version=0.8.0
        Generic Meta Data found:
                 WIFI Size=81 [bytes]
        Saved SSID   : crespan
        Saved PassWd : Elfrontal1
        Saved EncMode: WPA2/WPA2-Personal

Wait 3 seconds for allowing User Button Control:
           - Press one time  for adding the WIFI credential from UART or NFC
           - Press two times for adding the WIFI credential only from NFC


wifi started and ready...
WIFI module Configured
WiFi MAC Address is: 00:80:E1:B7:C3:12
WiFi connected to AccessPoint
WiFi opened socket with NTP server [time-d.nist.gov]
Set UTC Time: Wed May  3 13:47:32 2017

WIFI closed =0
Socket closed with NTP server
Init Real Time Clock 05-03-2017 13:47:32
Platform Init Done
Serializer Initialized
Registration Agent Launched
[Registration]. Devices founded
[Registration]. Device Regsitration to Microsoft Azure Successfully Completed
[Registration]. Connection String=
        HostName=STM32IoTHub.azure-devices.net;DeviceId=0080E1B7C312;SharedAccessKey=v852+WMq1sWOCDOEi
PwJfVckSzoTlG4+8t1DFQrYI9Q=
Board Registration Done

Connect to:
https://stm32ode.azurewebsites.net/Home/Index/0080E1B7C312

iotHubClientHandle Created
iotHubClientSetOption logtrace Ok
iotHubClientSetOption keepalive Ok
Model Instance Created
IoTHubClient_LL_SetMessageCallback...successful.
IoTHubClient_LL_SetDeviceMethodCallback...successful.
IoTHubClient_LL_SetOption(certificates)...successful.
IoTHubClient_LL_SetDeviceTwinCallback...successful.
JSON Decoded
Reported Property Found
Desired Property Found
DesiredTelemetryInterval Desired Property not Found
Channel 1 for Timer 1 started
Ok reported State [1]: Running
-->DeviceTwin CallBack [1]: Status_code = 204
IoTHubClient_LL_SendEventAsync accepted message [1] for transmission to IoT Hub.
Confirmation received for message [1] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [2] for transmission to IoT Hub.
Confirmation received for message [2] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [3] for transmission to IoT Hub.
Confirmation received for message [3] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
```

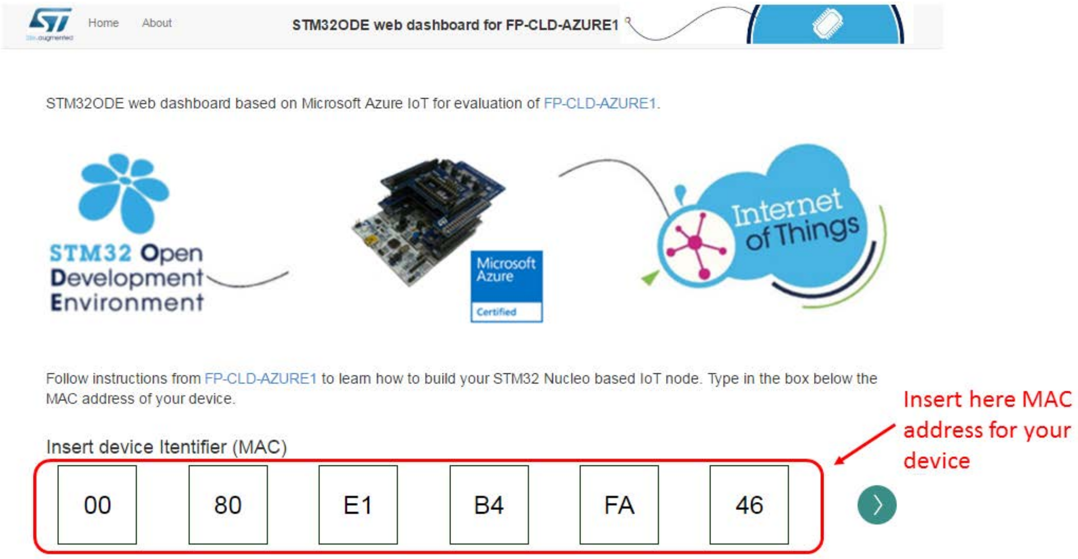
2.8.3 Usage of web dashboard for sensor data visualization

Once your board is registered with the IoT Hub, you can use it to visualize sensor data and control the device.

Step 1. Open your web browser at page : stm32ode.azurewebsites.net

Step 2. Type in the MAC address of your device (alternatively, you can copy and paste the URL shown in the serial terminal).

Figure 44. **Insert your device MAC address**



Home About

STM32ODE web dashboard for FP-CLD-AZURE1

STM32ODE web dashboard based on Microsoft Azure IoT for evaluation of FP-CLD-AZURE1.

Follow instructions from FP-CLD-AZURE1 to learn how to build your STM32 Nucleo based IoT node. Type in the box below the MAC address of your device.

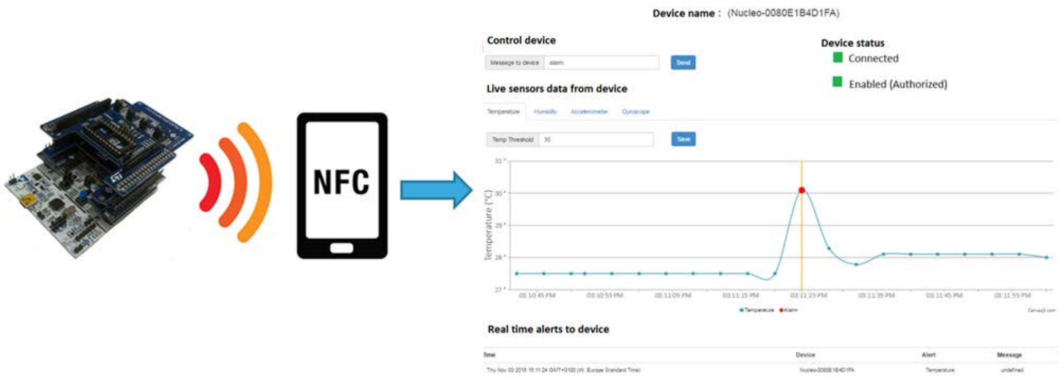
Insert device Itentifier (MAC)

00 | 80 | E1 | B4 | FA | 46

Insert here MAC address for your device

Step 3. If you are using an Android mobile device, the web page visualization of sensor data from your device can be automatically opened by placing the device near to the NFC tag.

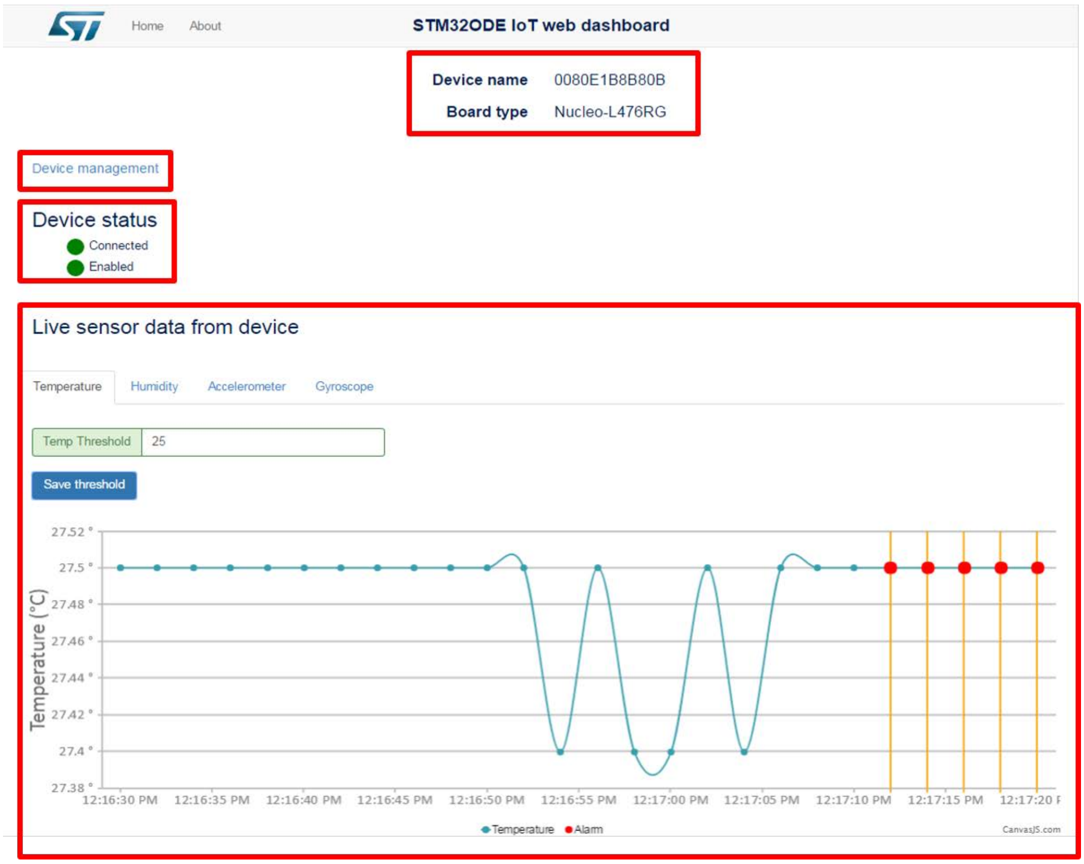
Figure 45. **Open web dashboard with NFC enabled Android phone**



2.8.3.1 *Overview of the web page sections*

The STM32 ODE web dashboard consists of two views (**Data Telemetry** and **Device Management**) and different sections.

Figure 46. STM32 ODE IoT web dashboard: Data Telemetry view



Device name 0080E1B8B80B
Board type Nucleo-L476RG

Device status
● Connected
● Enabled

Live sensor data from device

Temp Threshold: 25

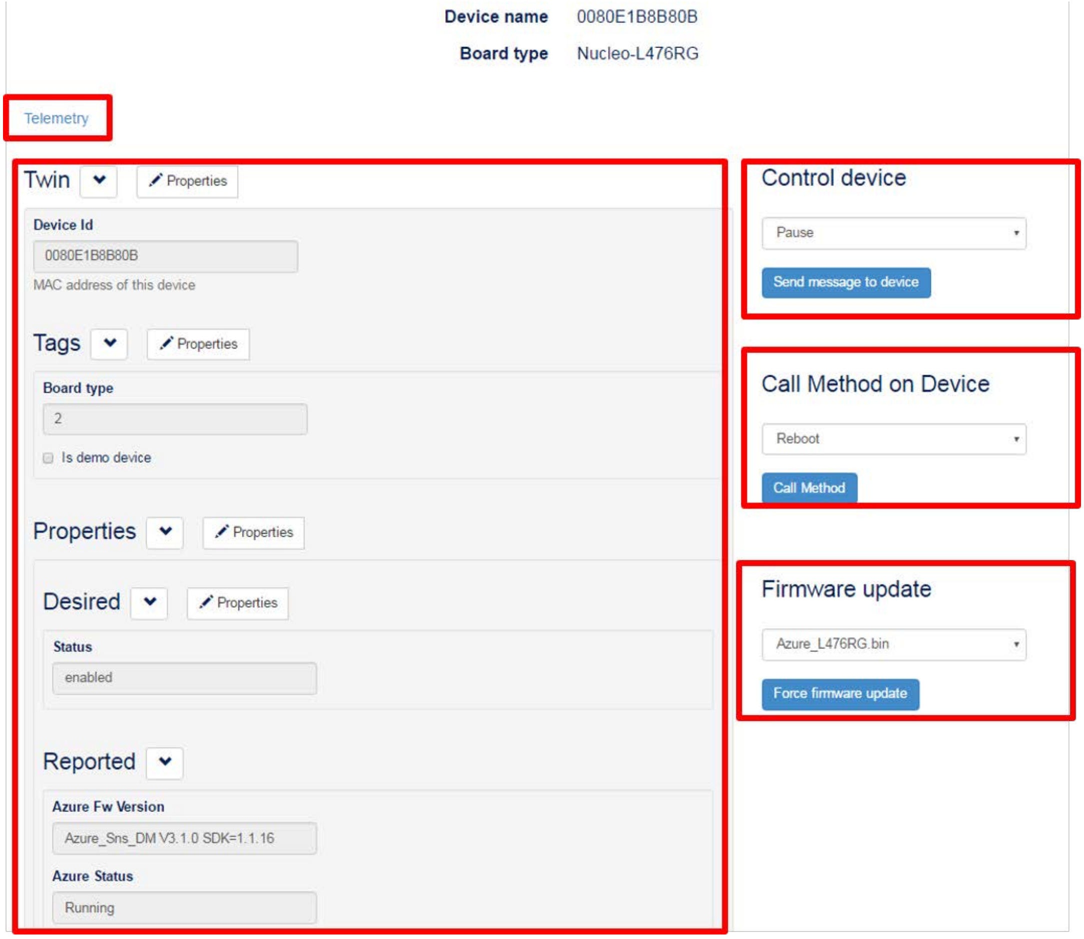
- Device name: MAC address of the device
- Board type: type platform used (Nucleo or Discovery Kit)
- Device management: link to Device Management view
- Device status: contains the following information:
  1. Connected/Disconnected: green if the device is connected to IoT Hub, red when device is disconnected
  2. Enabled/Warning/Disabled:

     — green if the device has not reached the maximum number of messages allowed per hour (see Section 2.8.4 Service limitations);

     — orange if device is approaching the maximum number of messages allowed per hour

     — red if device has exceeded the maximum number of messages allowed and has been consequently disabled.

Figure 47. **STM32 ODE IoT web dashboard: Device Management view**



Device name 0080E1B8B80B

Board type Nucleo-L476RG

Twin

Device Id: 0080E1B8B80B

MAC address of this device

Tags

Board type: 2

□ Is demo device

Properties

Desired

Status: enabled

Reported

Azure Fw Version: Azure_Sns_DM V3.1.0 SDK=1.1.16

Azure Status: Running

Control device: Pause

Call Method on Device: Reboot

Firmware update: Azure_L476RG.bin

- In the **Live sensor data from device** section, several tabs are provided to view environmental and inertial sensor data from the X-NUCLEO-IKS01A2 board or embedded in B-L475E-IOT01A.

  The Thresholds textbox can be used to set a limit value which triggers an alert message in the **Real time alerts** section; alert messages are also transmitted back to the device and printed over the serial terminal.
- Click on **Device management** link to control your device from the dashboard.
- Twin: shows the device Twin and can be expanded to read reported properties and modified to change a desired property
- Control device: selects Cloud-to-device messages and send the corresponding message to the device. If the Azure_Motor application is used, the drop-down menu includes the commands for motor control.
- Call Method on Device: triggers one of the methods activated in the firmware
- Firmware update: triggers the firmware update procedure (binary name depends on board type); it is possible to upload a custom firmware (not included in those pre-loaded and visible in drop-down menu). This feature is not available for NUCLEO-F401RE and for Azure_Motor application.
- Data telemetry: links to Data Telemetry view

2.8.4 Service limitations

- Usage of STM32 ODE web dashboard is provided free of charge for evaluation purposes only of FP-CLD-AZURE1; it does not require user registration.
- STM32 ODE web dashboard offers only a limited set of features and functions intended to provide a quickstart evaluation of STM32 Nucleo, expansion boards, and functionalities available in FP-CLD-AZURE1. The service is provided without warranties of any kind. Any use of the web dashboard in a production environment or for commercial purposes is not recommended or supported; any such use is therefore at proper risk.

- Sensor data sent to the service, alerts and cloud to device messages are not stored
- A maximum number of 500 message per hour per single device is allowed. Once such limit is approaching, a warning is sent to the device, forcing it to reduce transmission rate to one message each 15 seconds. Device is disabled once maximum number of allowed messages per hour is reached. Device status (enabled, warning, disabled) can be seen on web dashboard. Once disabled, a device is automatically re-enabled by the system the next day.
- ST may at any time suspend, revoke, or limit the usage of the service.

3 System setup guide

3.1 Hardware description

This section describes the hardware components needed to connect the STM32 Nucleo platforms and Discovery Kit for IoT node to the Microsoft Azure IoT Hub.

3.1.1 STM32 Nucleo platform

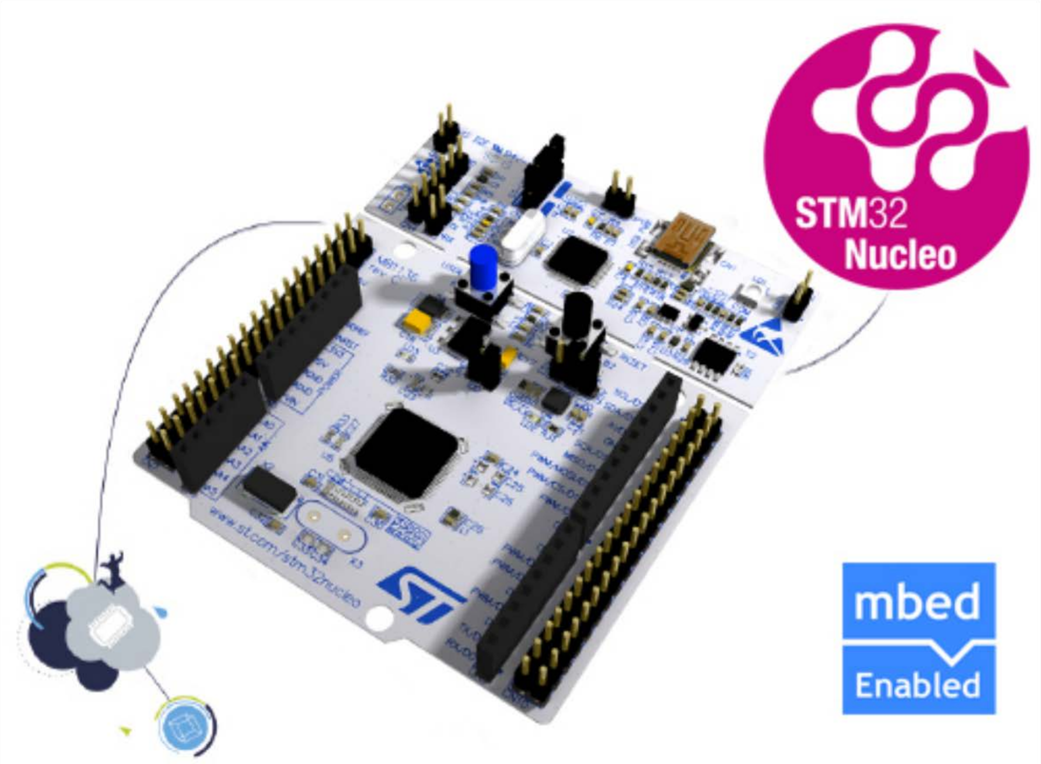
STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

The Arduino™ connectivity support and ST morpho connectors make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide range of specialized expansion boards to choose from.

The STM32 Nucleo board does not require separate probes as it integrates the ST-LINK/V2-1 debugger/ programmer.

The STM32 Nucleo board comes with the comprehensive STM32 software HAL library together with various packaged software examples.

Figure 48. STM32 Nucleo board



Information regarding the STM32 Nucleo board is available at www.st.com/stm32nucleo

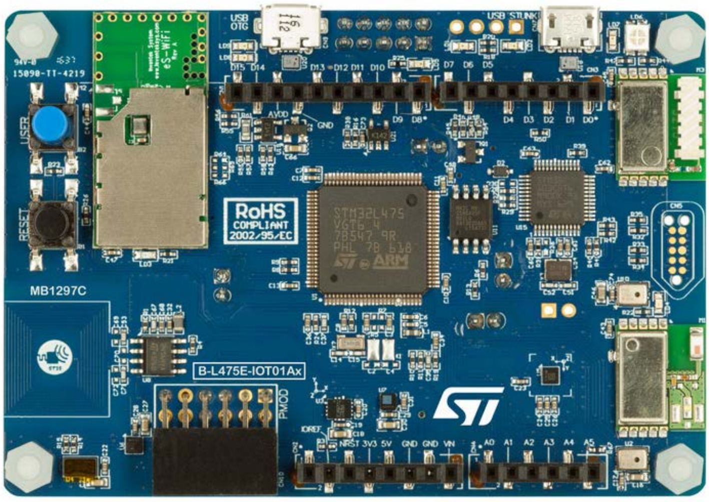
3.1.2 STM32L4 Discovery kit for IoT node

The STM32L4 Discovery kit for the IoT node (B-L475E-IOT01A) allows users to develop applications with direct connection to cloud servers. The STM32L4 Discovery kit enables a wide diversity of applications by exploiting low-power multilink communication (BLE, Sub-GHz), multiway sensing (detection, environmental awareness) and

ARM® Cortex®-M4 core-based STM32L4 Series features. Arduino™ Uno V3 and PMOD connectivity provide unlimited expansion capabilities with a large choice of specialized add-on boards.

The STM32L4 Discovery kit includes an ST-LINK debugger/programmer and comes with the comprehensive STM32Cube software libraries together with packaged software samples for a smooth connection to cloud servers.

Figure 49. **STM32L4 Discovery kit for IoT node**



Information regarding the STM32L4 Discovery kit for the IoT node is available at

3.1.3 X-NUCLEO-IDW01M1 expansion board

The X-NUCLEO-IDW01M1 is a Wi-Fi evaluation board based on the SPWF01SA module, which expands the STM32 Nucleo boards.

The CE, IC and FCC certified SPWF01SA module has an embedded STM32 MCU, a low-power Wi-Fi b/g/n SoC with integrated power amplifier and power management and an SMD antenna.

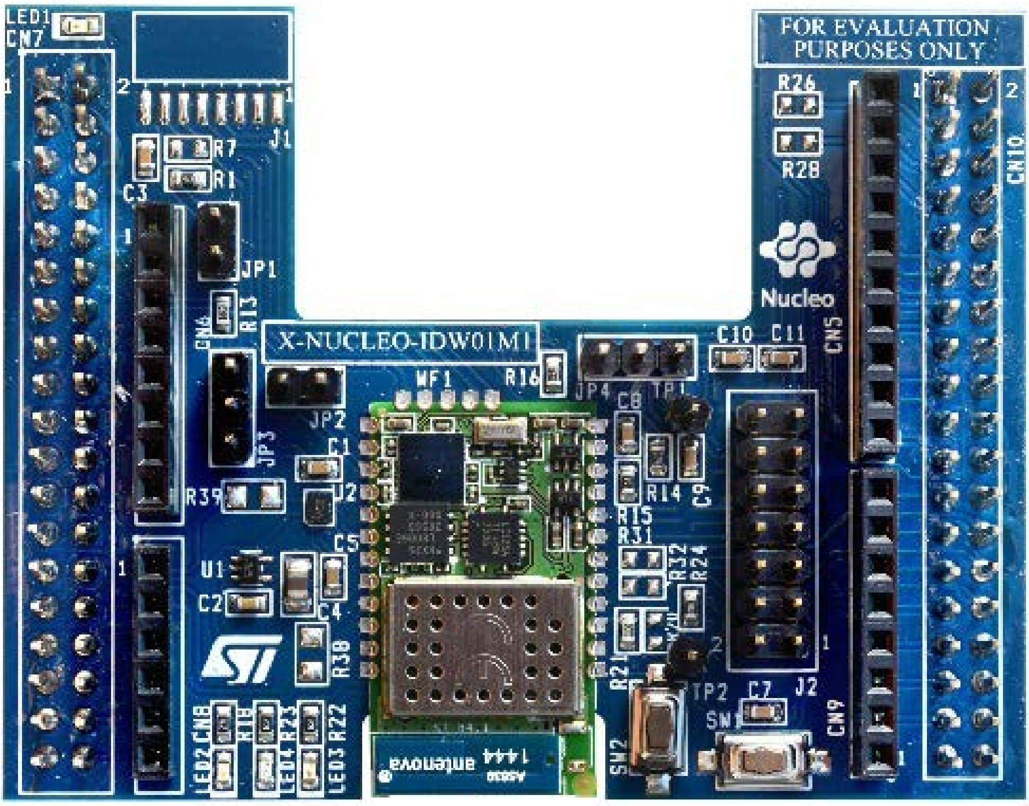
The SPWF01SA module is also equipped with 1 MByte of external FLASH for firmware update over-the-air (FOTA).

The firmware features a complete software IP stack to open up to 8 TCP/UDP sockets, as well as dynamic web pages with SSI to interact with the module and a REST API (get & post) for conveniently transferring files to/from servers in the cloud. The module can simultaneously behave as a socket server and socket client.

The firmware supports secure sockets with TLS/SSL encryption, ensuring secure end-to-end communications with the cloud, with or without authentication. The module operates as a client STA, IBSS, or miniAP (with up to 5 client STAs).

The X-NUCLEO-IDW01M1 interfaces with the MCU on the STM32 Nucleo board via the UART serial port; the user can easily access the stack functions using the AT command. X-NUCLEO-IDW01M1 is compatible with both the ST morpho and Arduino UNO R3 connector layout.

Figure 50. X-NUCLEO-IDW01M1 Wi-Fi expansion board



Information regarding the expansion board is available on www.st.com at http://www.st.com/x-nucleo.

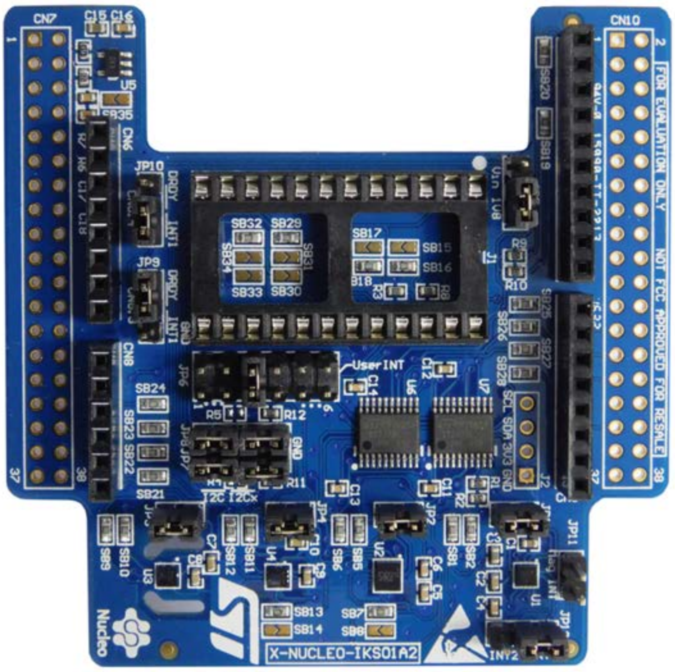
3.1.4 X-NUCLEO-IKS01A2 expansion board

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

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

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

Figure 51. X-NUCLEO-IKS01A2 MEMS and environmental sensor expansion board



3.1.5 X-NUCLEO-IHM02A1 expansion board

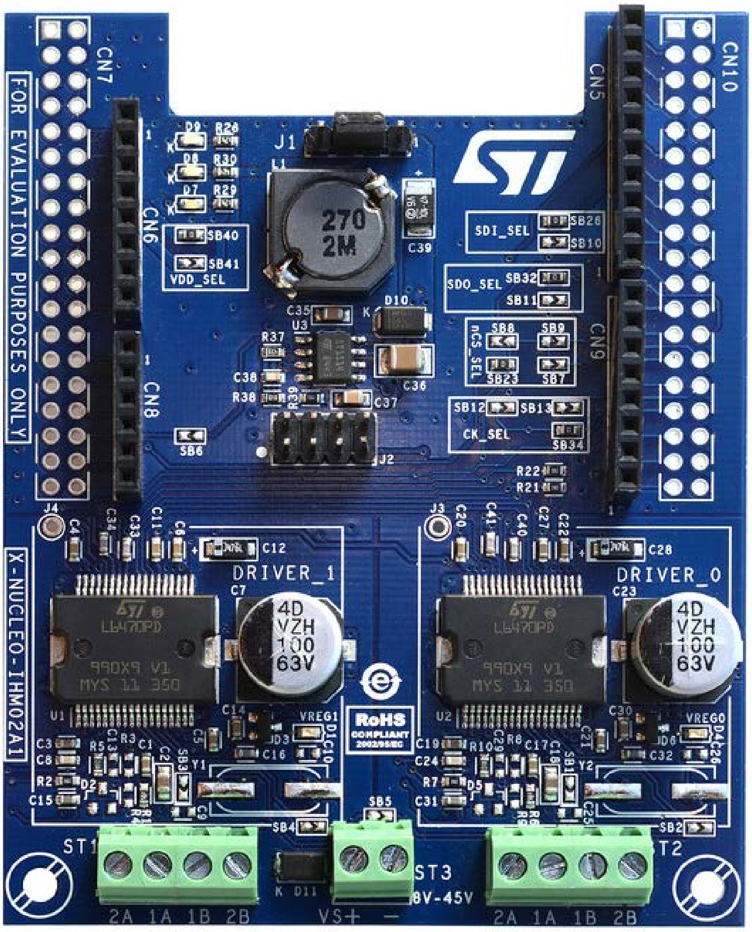
The X-NUCLEO-IHM02A1 is a two axis stepper motor driver expansion board based on the L6470. It provides an affordable and easy-to-use solution for low voltage motor control driving for stepper motors in your STM32 Nucleo project.

The expansion board includes two L6470s, a fully-integrated micro stepping motor driver used to control stepper motors by means of high-end motion control commands received through SPI. It is capable of driving one or two stepper motors when plugged into an STM32 Nucleo board.

This board is equipped with Arduino UNO R3 connectors and the layout is also compatible with ST morpho connectors. One or more of these expansion boards can be plugged into an STM32 Nucleo board to control one or more stepper motors.

Each SPI peripheral of each L6470 is connected in a daisy chain configuration.

Figure 52. X-NUCLEO-IHM02A1 expansion board



3.1.6 X-NUCLEO-NFC04A1 expansion board

The X-NUCLEO-NFC04A1 dynamic NFC/RFID tag IC expansion board is based on the ST25DV04K NFC Type V/ RFID tag IC with a dual interface 4 Kbits EEPROM that also features an I²C interface. It can be powered by the pin of Arduino connector or directly by the received carrier electromagnetic field.

The X-NUCLEO-NFC04A1 expansion board is compatible with the Arduino™ UNO R3 connector pin assignment and can easily be plugged onto any STM32 Nucleo board. Various expansion boards can also be stacked to evaluate different devices operating together with the dynamic NFC tag.

The board also features an antenna with a 54 mm iso 24.2 diameter, single layer, copper etched on PCB.

Figure 53. X-NUCLEO-NFC04A1 expansion board

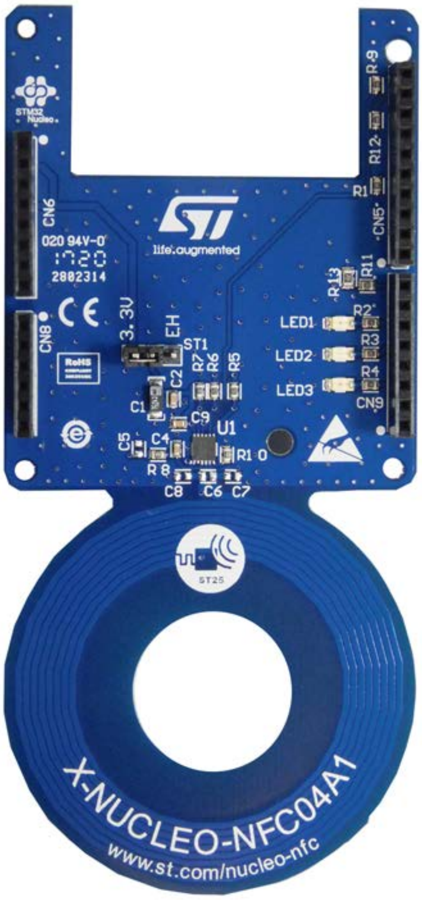
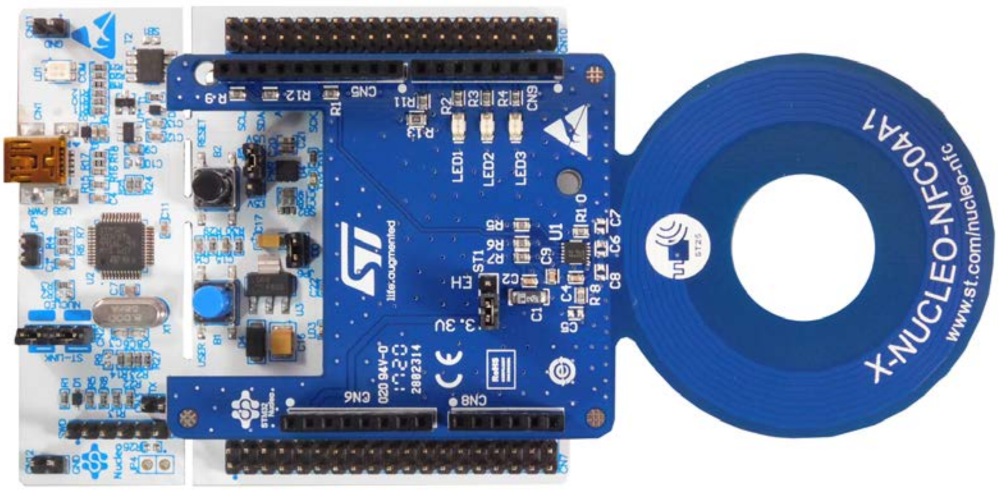


Figure 54. X-NUCLEO-NFC04A1 expansion board plugged to an STM32 Nucleo board



Information about the X-NUCLEO-NFC04A1 expansion board is available on www.st.com at http://www.st.com/x-nucleo

3.2 Software requirements

The following software components are needed to set up a suitable development environment for compiling and running the FP-CLD-AZURE1 package:

- FP-CLD-AZURE1 software available on www.st.com/stm32ode
- Development tool-chain and compiler; the FP-CLD-AZURE1 software supports the three following environments:
  - IAR Embedded Workbench for ARM® (IAR-EWARM) toolchain + ST-LINK
  - RealView Microcontroller Development Kit (MDK-ARM-STM32) toolchain + ST-LINK
  - System Workbench for STM32 + ST-LINK
- Serial line monitor (e.g., TeraTerm, https://ttssh2.osdn.jp/)
- Microsoft Device Explorer tool available on https://github.com/Azure/azure-iot-sdk-csharp/tree/master/tools/DeviceExplorer
- For using the pre-compiled binaries and web dashboard: Chrome® web browser (http://www.google.com/chrome/)

3.3 Hardware and software setup

3.3.1 Hardware setup

The following hardware components are needed:

1. One STM32 Nucleo development board (order code: NUCLEO-F401RE or NUCLEO-L476RG)
2. One Wi-Fi expansion board (order code: X-NUCLEO-IDW01M1)
3. For **Azure_Sns_DM** application: one sensor expansion board (order code: X-NUCLEO-IKS01A2)
4. For **Azure_Motor** application: one motor control expansion board (order code: X-NUCLEO-IHM02A1)
5. One dynamic NFC tag expansion board (optional, order code: X-NUCLEO-NFC04A1)
6. One USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC

Or in alternative:

1. One STM32 Nucleo-144 development board (order code: NUCLEO-F429ZI)
2. One sensor expansion board (order code: X-NUCLEO-IKS01A2)
3. One dynamic NFC tag expansion board (optional, order code: X-NUCLEO-NFC04A1)
4. One USB type A to Micro-B USB cable to connect the STM32 Nucleo-144 to the PC
5. One RJ-45 cable to connect the STM32 Nucleo-144 to Ethernet.

Or in alternative:

1. One STM32L4 Discovery Kit for IoT node (order code: B-L475E-IOT01A)
2. For **Azure_Motor** application: one motor control expansion board (order code: X-NUCLEO-IHM02A1)
3. One USB type A to Micro-B USB cable to connect the STM32L4 Discovery Kit to the PC

3.3.2 Software setup

3.3.2.1 *Development tool-chains and compilers*

Select one of the IDEs in Section 3.2 Software requirements and refer to the system and setup information provided by the selected IDE provider. Project files for all of the supported IDEs can be found inside one of the following FP-CLD-AZURE1 package folders, i.e. for IAR Embedded Workbench:

- Projects\Multi\Applications\Azure_Sns_DM\EWARM\B-L475E-IOT01
- Projects\Multi\Applications\Azure_Sns_DM\EWARM\STM32F401RE-Nucleo
- Projects\Multi\Applications\Azure_Sns_DM\EWARM\STM32F476RG-Nucleo

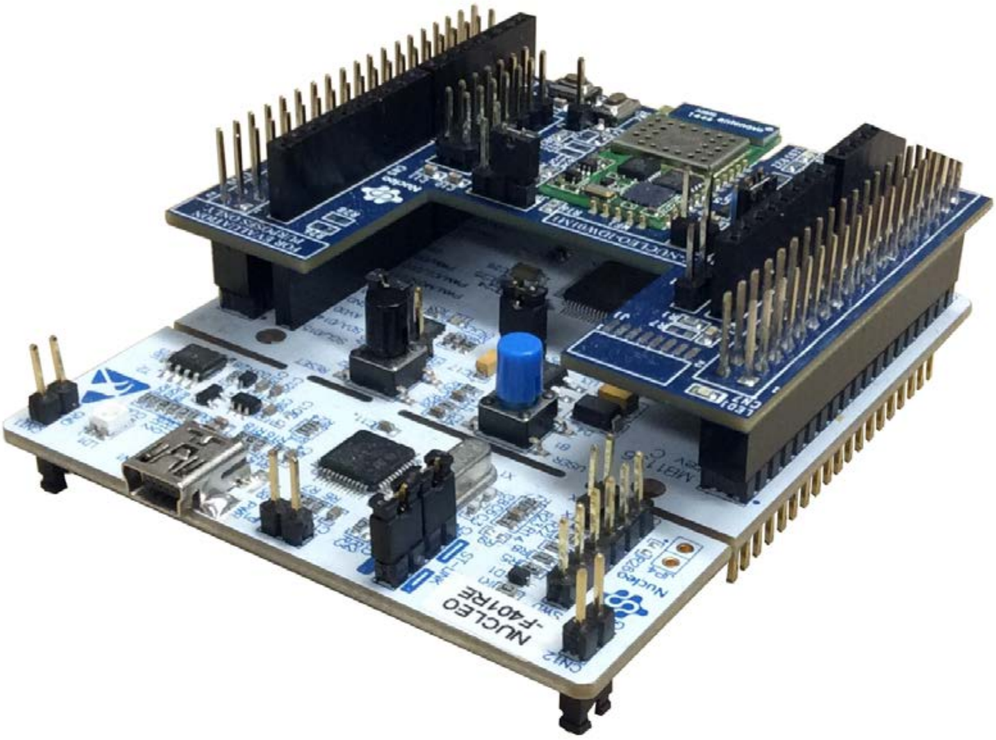
- Projects\Multi\Applications\Azure_Sns_DM\EWARM\STM32F429ZI-Nucleo

3.3.3 System setup guide for STM32 Nucleo and expansion boards

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer. You can download the relevant version of the ST-LINK/V2-1 USB driver by searching STSW-LINK008 or STSW-LINK009 on www.st.com (based on your version of Microsoft Windows).

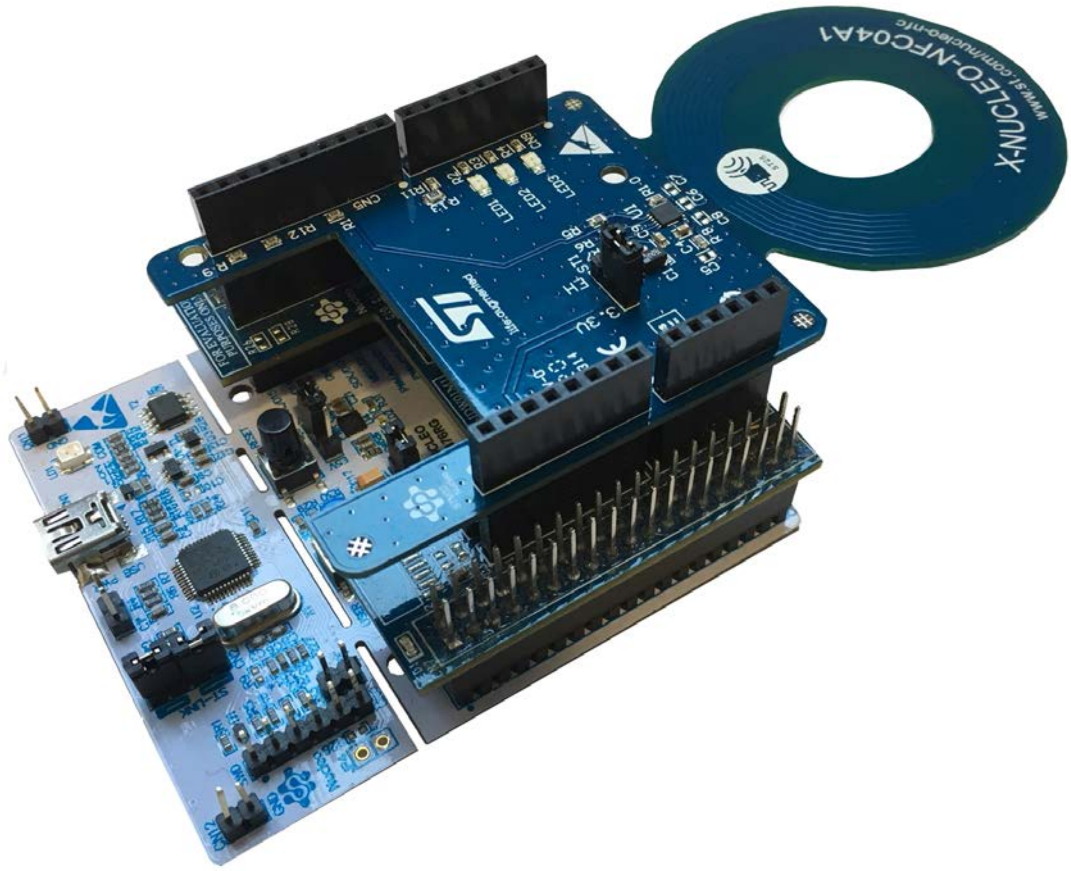
The X-NUCLEO-IDW01M1 Wi-Fi expansion board is connected on the STM32 Nucleo board ST morpho extension connector, as shown below.

Figure 55. X-NUCLEO-IDW01M1 Wi-Fi expansion board connected to the STM32 Nucleo board ST morpho connectors



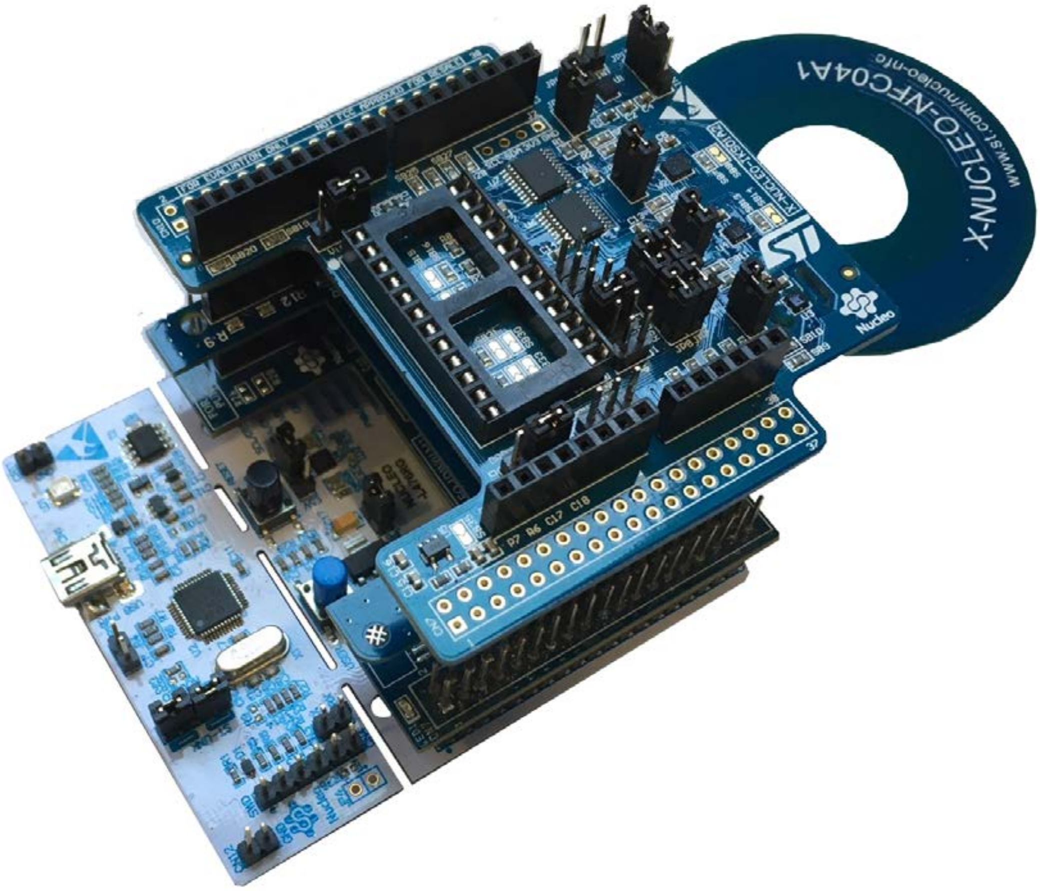
The Dynamic NFC tag board X-NUCLEO-NFC04A1 is connected to X-NUCLEO-IDW01M1 expansion board Arduino UNO R3 extension connector, as shown below.

Figure 56. STM32 Nucleo development board plus X-NUCLEO-IDW01M1 plus X-NUCLEO-NFC04A1 expansion boards



Finally, the X-NUCLEO-IKS01A2 sensor board can be easily connected to X-NUCLEO-NFC04A1 expansion board Arduino UNO R3 extension connector, as shown below.

Figure 57. **STM32 Nucleo development board plus X-NUCLEO-IDW01M1 plus X-NUCLEO-NFC04A1 plus X-NUCLEO-IKS01A2 expansion boards**

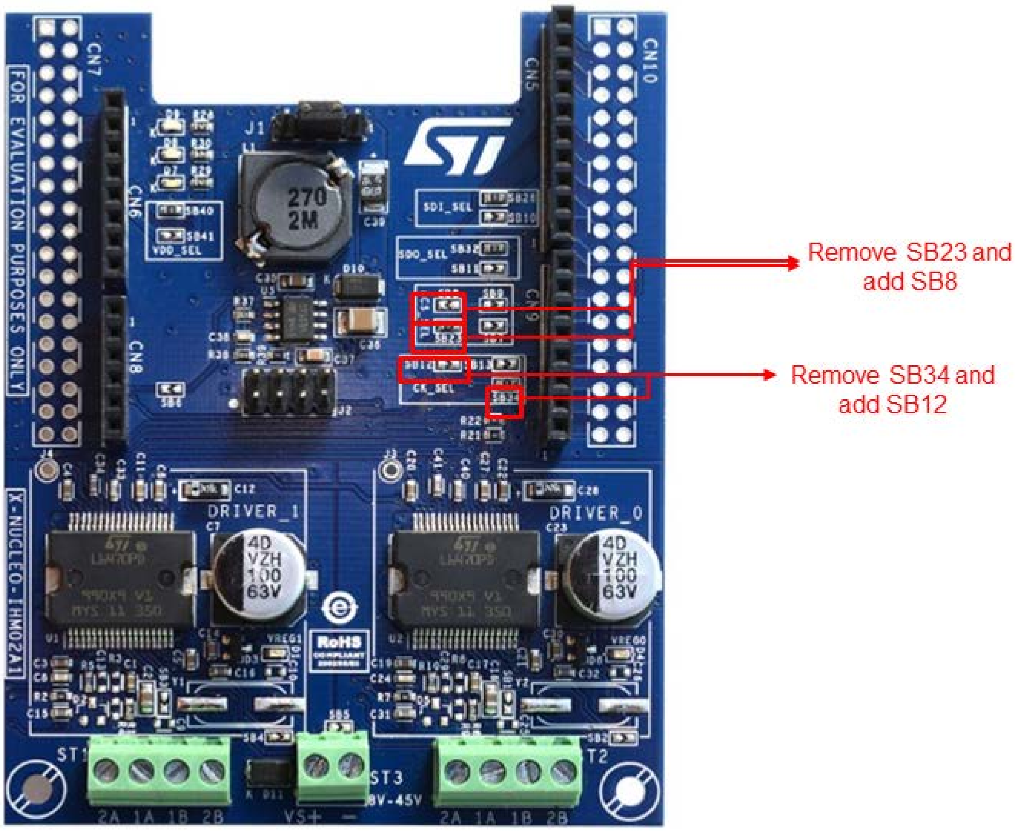


For **Azure_Motor** application, in the above precedure replace X-NUCLEO-IKS01A2 with X-NUCLEO-IHM01A1 expansion board for motor control.

X-NUCLEO-IHM01A1 can also be used with STM32L4 Discovery Kit for IoT node (B-L475-IOT01A), but it requires the following modifications in the hardware:

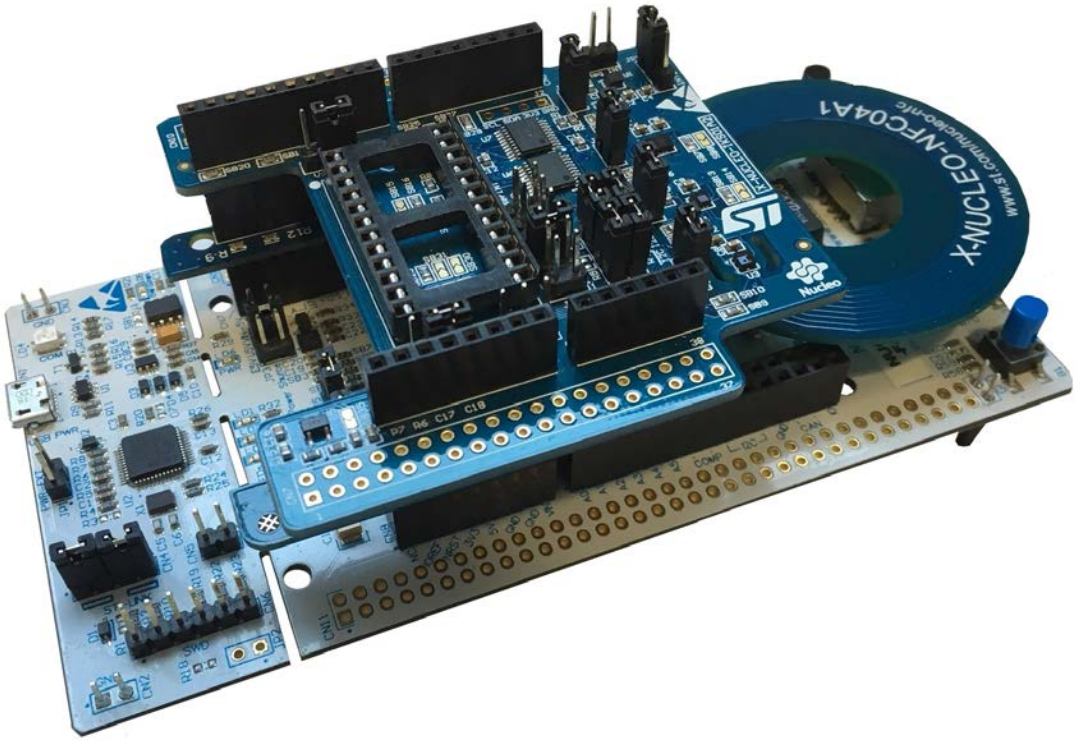
- Remove Solder Bridge SB34 and add Solder Bridge SB12
- Remove Solder Bridge SB23 and add Solder Bridge SB8

Figure 58. Hardware configuration for X-NUCLEO-IHM01A1 expansion board when used together with B-L475E-IOT01A



For **Azure_Sns_DM** application, in alternative to Wi-Fi connectivity, it is possible to use a NUCLEO-144 F429ZI board with integrated Ethernet interface, connected to either X-NUCLEO-IKS01A2 sensor boards and optionally to X-NUCLEO-NFC04A1, as shown below.

Figure 59. NUCLEO-144 with X-NUCLEO-IKS01A2 and X-NUCLEO-NFC04A1 expansion boards



Revision history

Table 2. **Document revision history**

Date	Version	Changes
23-Mar-2016	1	Initial release.
29-Apr-2016	2	Minor text edits.
13-Dec-2016	3	Updated for v2.0 firmware. Added companion web application information. Added X-NUCLEO-IKS01A2 support information.
06-Jun-2017	4	Updated all content to reflect v3.0 firmware.
19-Oct-2017	5	Updated all content to reflect v3.1 firmware.
07-May-2018	6	To reflect v3.3 firmware: • updated Introduction, Section 2.1 Overview, Section 2.3 Folder structure, Section 2.7 Azure IoT sample application description, Section 2.7.2.2 Use pre-compiled binaries , Section 2.7.2.3 Modify and rebuild solution files, Section 2.8.3.1 Overview of the web page sections, Section 3.3.1 Hardware setup and Section 3.3.3 System setup guide for STM32 Nucleo and expansion boards; • added Section 2.7.7 Motor control sample application, Section 2.7.8 IoT Central sample for Azure_Sns_DM application, Section 3.1.5 X-NUCLEO-IHM02A1 expansion board and Section 3.1.6 X-NUCLEO-NFC04A1 expansion board; • removed references to X-NUCLEO-IKS01A1 and X-NUCLEO-NFC01A1 expansion boards.

Contents

List of figures

List of tables

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.