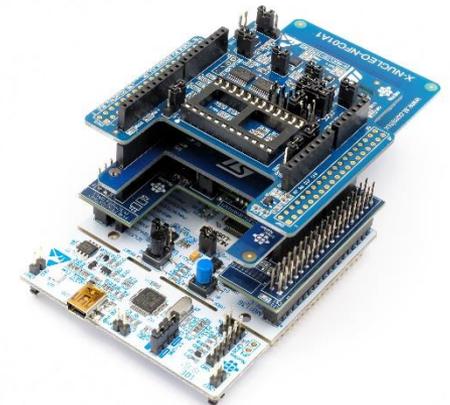
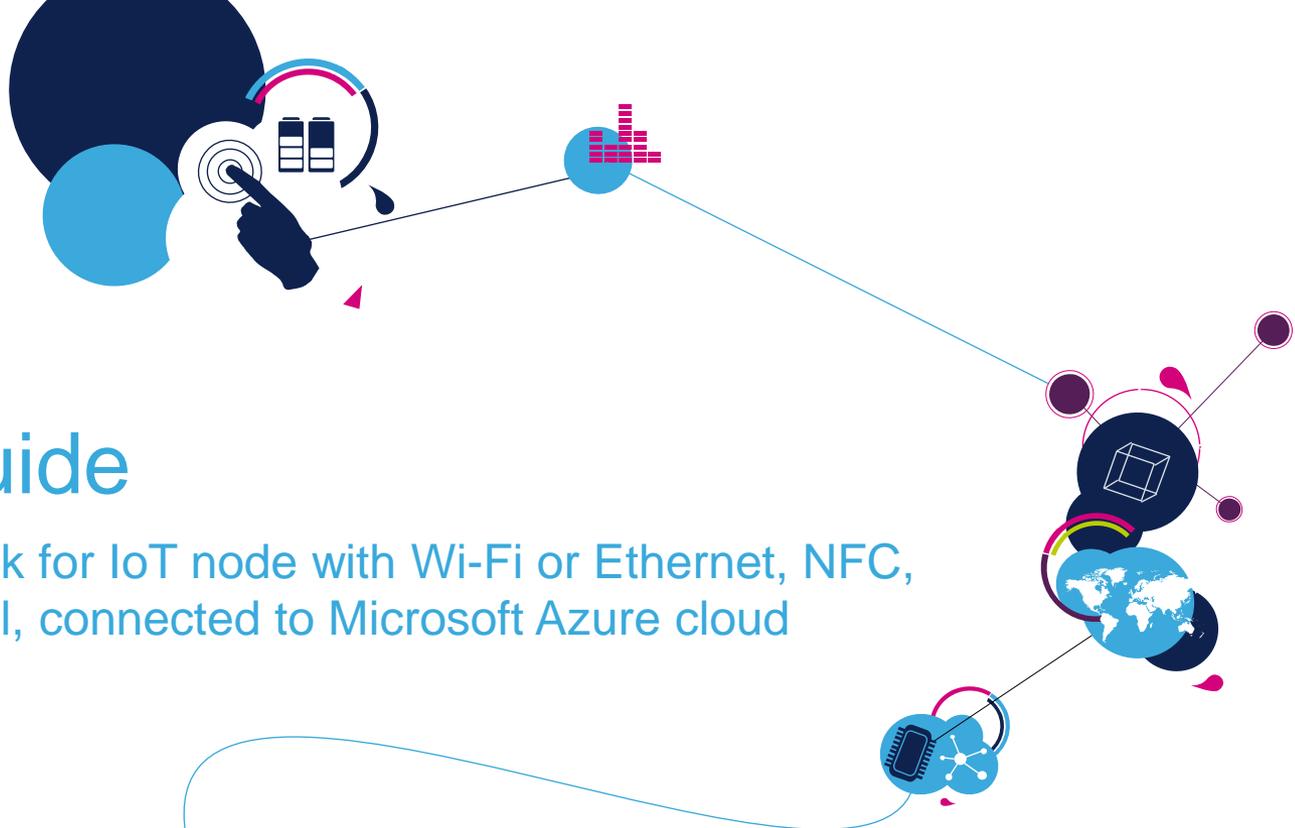
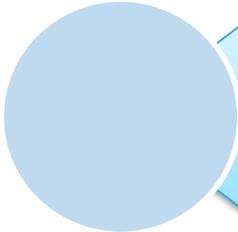


# Quick Start Guide

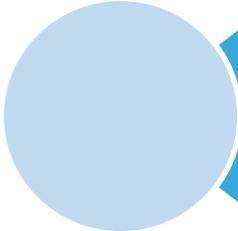
STM32 ODE function pack for IoT node with Wi-Fi or Ethernet, NFC, sensors and motor control, connected to Microsoft Azure cloud (FP-CLD-AZURE1)





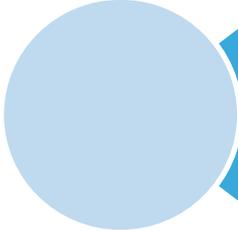
FP-CLD-AZURE: STM32 ODE function pack for IoT node with Wi-Fi or Ethernet, NFC, sensors and motor control, connected to Microsoft Azure cloud

Hardware and Software overview



Setup & Demo Examples

Documents & Related Resources



STM32 Open Development Environment: Overview

# Motion MEMS and Environmental Sensors Expansion Board Hardware Overview (2/6)

4

## X-NUCLEO-IKS01A2 Hardware Description

The X-NUCLEO-IKS01A2 is a motion MEMS and environmental sensor expansion board for the 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 I<sup>2</sup>C pin, and it is possible to change the default I<sup>2</sup>C port.

### Key Product on board

#### LSM6DSL

MEMS 3D accelerometer ( $\pm 2/\pm 4/\pm 8/\pm 16$  g) + 3D gyroscope ( $\pm 125/\pm 245/\pm 500/\pm 1000/\pm 2000$  dps)

#### LSM303AGR

MEMS 3D magnetometer ( $\pm 50$  gauss) + MEMS 3D accelerometer ( $\pm 2/\pm 4/\pm 8/\pm 16$  g)

#### LPS22HB

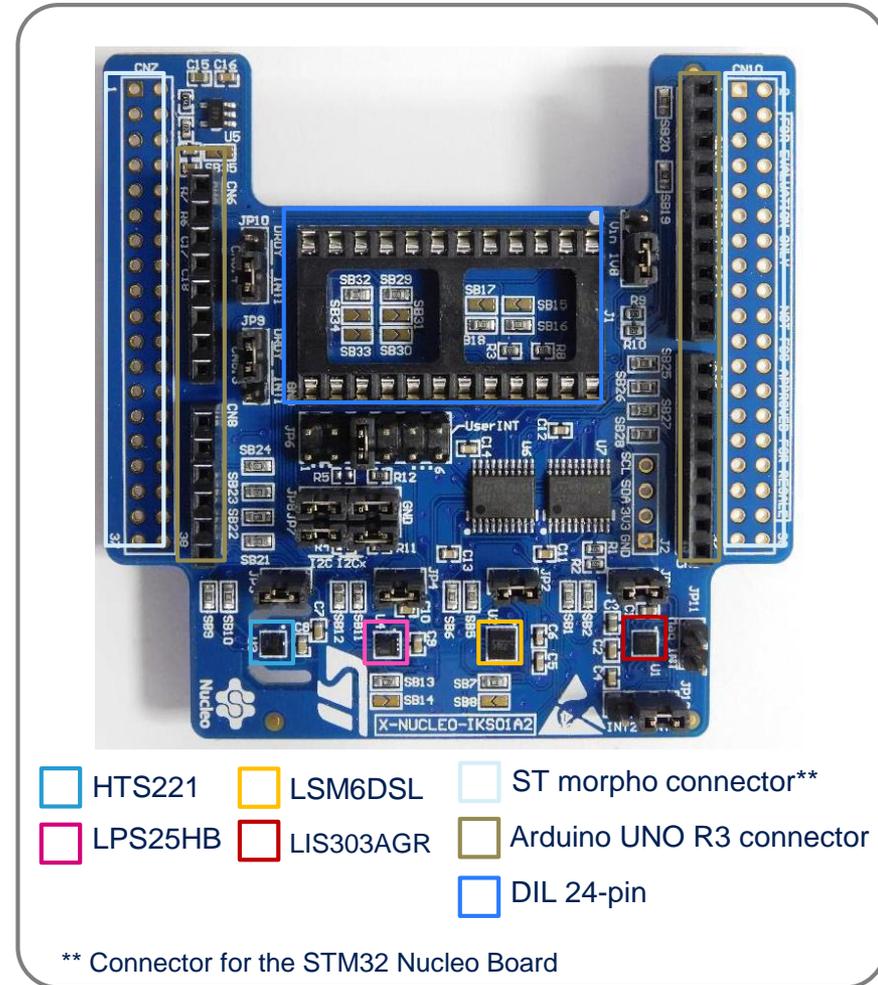
MEMS pressure sensor, 260-1260 hPa absolute digital output barometer

#### HTS221

Capacitive digital relative humidity and temperature

#### DIL 24-pin

Socket available for additional MEMS adapters and other sensors



Latest info available at [www.st.com](http://www.st.com)  
**X-NUCLEO-IKS01A2**

## X-NUCLEO-NFC04A1 Hardware Description

- 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 I<sup>2</sup>C 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.

### Key products on board

#### ST25DV04KV

Dynamic NFC/RFID tag IC with 4-Kbit, 16-Kbit or 64-Kbit EEPROM, and Fast Transfer Mode capability



# Two axis stepper motor driver expansion board

## Hardware Overview (4/6)

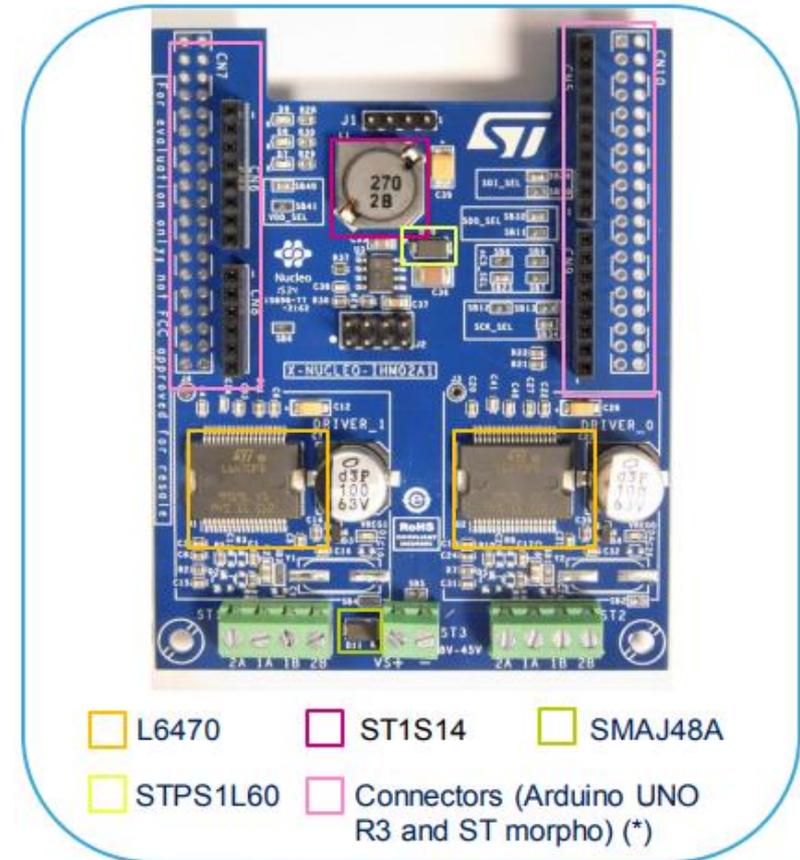
6

### X-NUCLEO-IHM02A1 Hardware description

The X-NUCLEO-IHM02A1 is a two axis stepper motor driver expansion board based on L6470. It provides an affordable and easy-to-use solution for driving low voltage motor control for Stepper Motor 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.

### Main features:

- Nominal operating voltage range: 8 V - 45 V DC
- Maximum output peak current: 7.0 A (3.0 A rms) for each motor driver
- Digital voltage supply is selectable (3.3 V or 5.0 V)
- USART communication
- SPI interface (may be connected in a daisy chain configuration)
- Equipped with Arduino UNO R3 connectors
- Layout compatible with ST morpho connectors



### Key Products on board

#### L6470

Fully integrated microstepping motor driver with motion engine and SPI

#### ST1S14

Up to 3 A step down switching regulator

#### SMAJ48A

Transil

#### STPS1L60

Low Drop Power Schottky Rectifier

Latest info available at [www.st.com](http://www.st.com)  
**X-NUCLEO-IHM02A1**

(\*) only Arduino is mounted by default

# STM32L4 Discovery Board for IoT node (B-L475E-IOT01A)

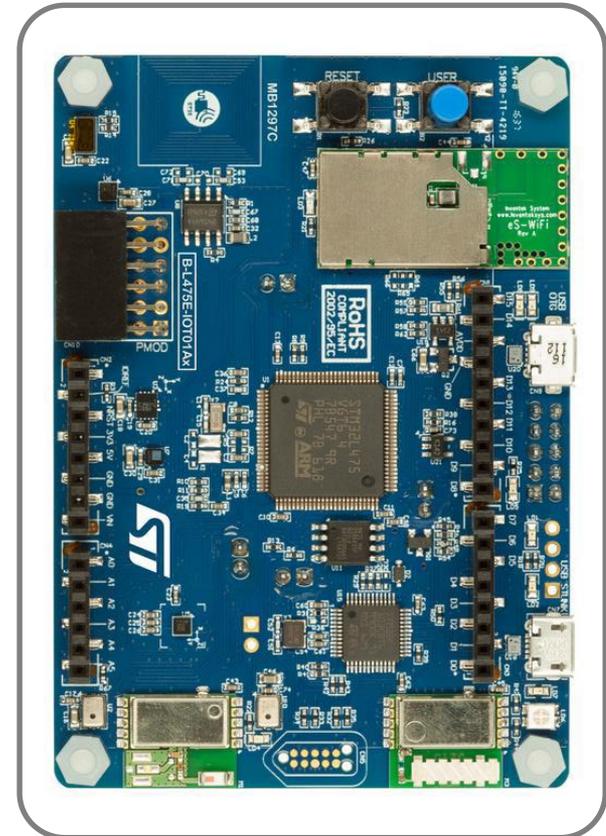
## Hardware Overview (6/6)

### STM32L4 Discovery Board for IoT node (B-L475E-IOT01A) Hardware Description

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.

#### Key Product on board

- Ultra-low-power STM32L4 Series MCUs based on ARM® Cortex® -M4 core with 1 Mbyte of Flash memory and 128 Kbytes of SRAM, in LQFP100 package
- Bluetooth® V4.1 module (SPBTLE-RF)
- Sub-GHz (868 or 915 MHz) low-power-programmable RF module (SPSGRF-868 or SPSGRF-915)
- Wi-Fi® module Inventek ISM43362-M3G-L44 (802.11 b/g/n compliant)
- Dynamic NFC tag based on M24SR with its printed NFC antenna
- 2 digital omnidirectional microphones (MP34DT01)
- Capacitive digital sensor for relative humidity and temperature (HTS221)
- High-performance 3-axis magnetometer (LIS3MDL), 3D accelerometer and 3D gyroscope (LSM6DSL), 260-1260 hPa absolute digital output barometer (LPS22HB), Time-of-Flight and gesture-detection sensor (VL53L0X)
- USB OTG FS with Micro-AB connector
- Expansion connectors: Arduino™ Uno V3, PMOD
- Flexible power-supply options: ST LINK USB VBUS or external sources
- On-board ST-LINK/V2-1 debugger/programmer with USB re-enumeration capability: mass storage, virtual COM port and debug port



Latest info available at [www.st.com](http://www.st.com)  
**B-L475E-IOT01A**

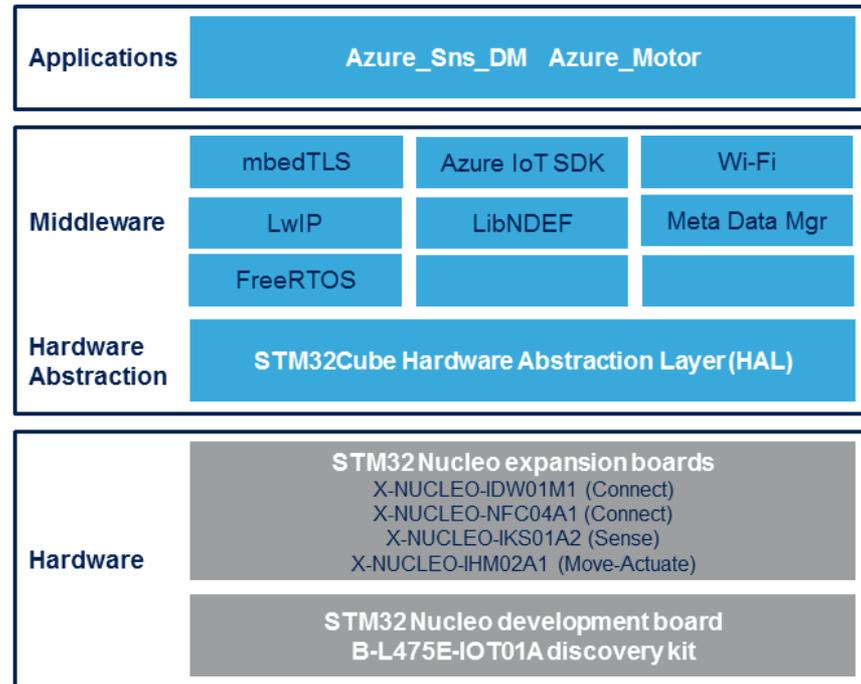
### FP-CLD-AZURE1 Software Description

FP-CLD-AZURE1 is an STM32 ODE Function Pack. Thanks to this package you can directly connect your IoT sensor node to the Microsoft Azure IoT, transmit sensors data and receive command from Cloud applications.

### Key features

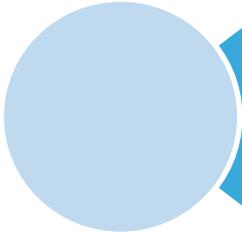
- Complete firmware to safely connect an IoT node with sensors and actuators to Microsoft Azure IoT, using Wi-Fi or Ethernet communication technology. Two sample applications for data telmetry/device management (**Azure\_Sns\_DM**) and for motor control (**Azure\_Motor**)
- Middleware libraries featuring the Microsoft Azure IoT software development kit, Wi-Fi and NFC connectivity, Motor Control, 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, actuators 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-NUCLEOIKS01A2, X-NUCLEO-IDW01M1, X-NUCLEO-IHM02A1 and X-NUCLEONFC04A1, 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 is Microsoft Azure certified for IoT (for more information on Microsoft Azure Certification please visit <http://azure.com/certifiedforiot>)

### Overall Software Architecture

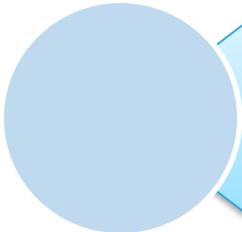


Latest info available at [www.st.com](http://www.st.com)

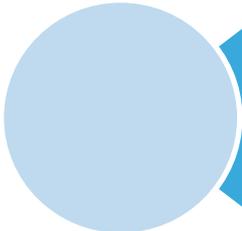
**FP-CLD-AZURE1**



FP-CLD-AZURE: STM32 ODE function pack for IoT node with Wi-Fi or Ethernet, NFC and sensors, connected to Microsoft Azure cloud  
Hardware and Software overview



Setup & Application Examples  
Documents & Related Resources



STM32 Open Development Environment: Overview

# Setup & Application Examples (Azure\_Sns\_DM)

## HW prerequisites for B-L475E-IOT01A

- 1x B-L475E-IOT01A development board
- NFC-enabled Android™ device (optional)
- Laptop/PC with Windows 7, 8 or 10
- 1 x microUSB cable
- Wi-Fi Router or access to a Wi-Fi network



MicroUSB Cable

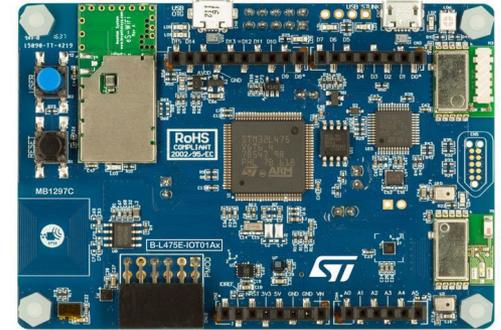


B-L475E-IOT01A

# Setup & Application Examples (Azure\_Motor)

## HW prerequisites for B-L475E-IOT01A (1/2)

- 1x B-L475E-IOT01A development board
- 1x two axis stepper motor driver expansion board (**X-NUCLEO-IHM02A1**): **HW modification required**
- 1x Dynamic NFC tag expansion board expansion board for STM32 Nucleo (**X-NUCLEO-NFC04A1**, optional)
- NFC-enabled Android™ device (optional)
- Laptop/PC with Windows 7, 8 or 10
- 1 x microUSB cable
- Wi-Fi Router or access to a Wi-Fi network



B-L475E-IOT01A



MicroUSB Cable



X-NUCLEO-IHM02A1

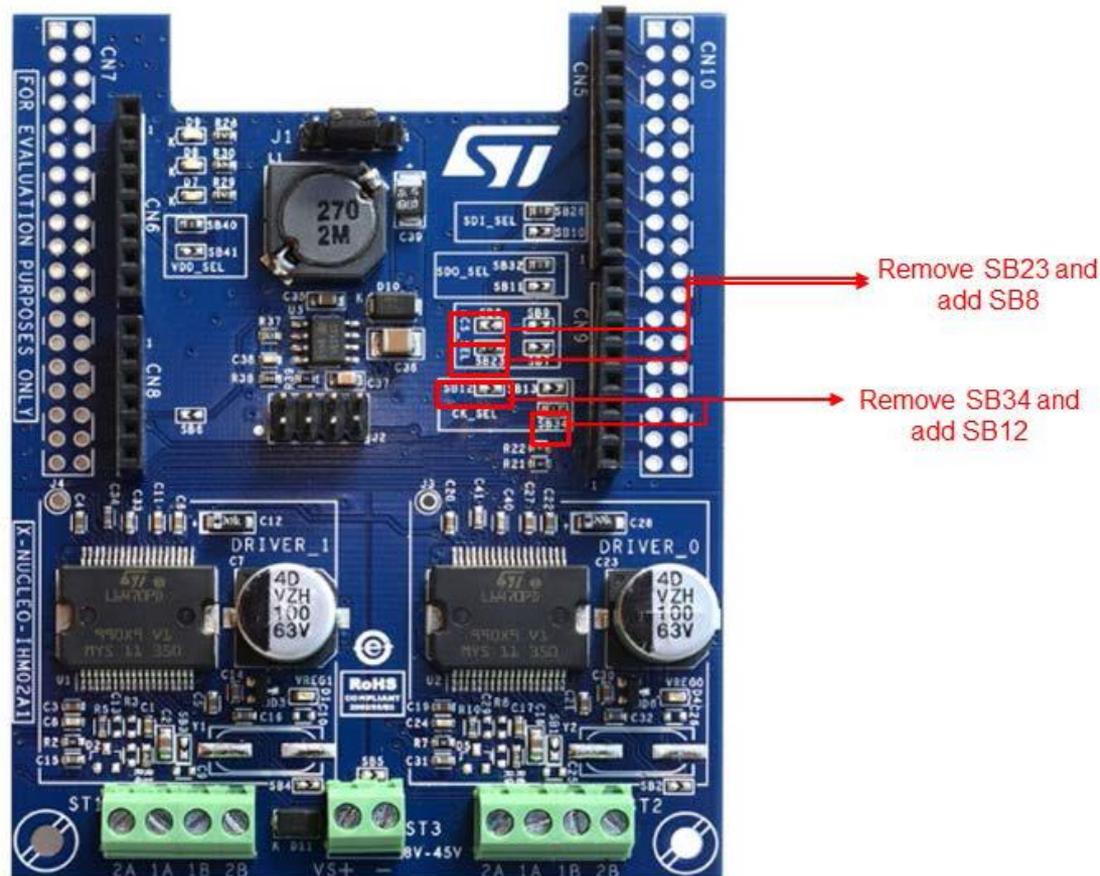


X-NUCLEO-NFC04A1

# Setup & Application Examples (Azure\_Motor)

## HW prerequisites for B-L475E-IOT01A (2/2)

- X-NUCLEO-IHM01A1 can be used with STM32L4 Discovery Kit for IoT node (B-L475-IOT01A), with the following modifications in the hardware:
  - Remove Solder Bridge SB34 and add Solder Bridge SB12
  - Remove Solder Bridge SB23 and add Solder Bridge SB8



# Setup & Application Examples

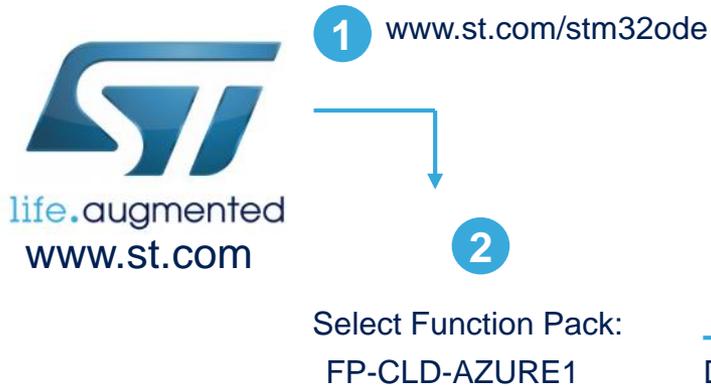
## Software and Other prerequisites

20

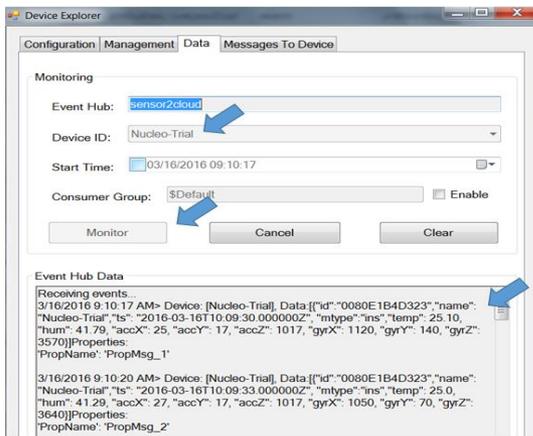
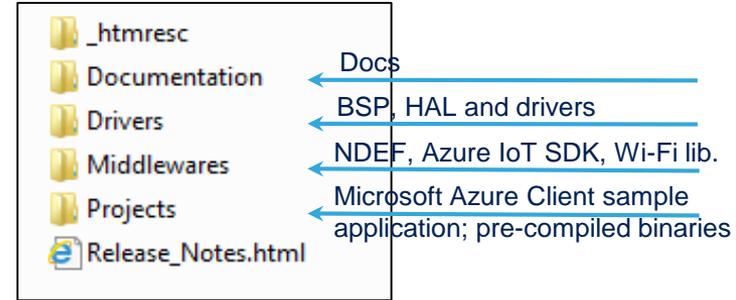
- **STM32 ST-Link Utility**
  - Download and install [STSW-LINK004](#) from [www.st.com](http://www.st.com)
- **FP-CLD-AZURE1**
  - Download [FP-CLD-AZURE1](#) package from [www.st.com](http://www.st.com), copy the .zip file content into a folder on your PC. The package contains binaries and source code with project files ([Keil](#), [IAR](#), [System Workbench](#)) based on NUCLEO-F401RE, NUCLEO-L476RG, NUCLEO-F429ZI and B-L475E-IOT01A.
- **Serial line monitor**, e.g. TeraTerm (<https://ttssh2.osdn.jp/>)
- To write/read NFC tag
  - Any **Android application capable** to read/write NFC tag (i.e. ST25 NFC <https://play.google.com/store/apps/details?id=com.st.st25nfc> )
- To register a new device in Azure IoT Hub and test FP-CLD-AZURE1 with custom Azure account:
  - An active account on Microsoft Azure (<https://azure.microsoft.com/en-us/pricing/free-trial/> )
  - **Microsoft Device Explorer utility** (<https://github.com/Azure/azure-iot-sdk-csharp/tree/master/tools/DeviceExplorer>) or **iothub-explorer** (<https://github.com/Azure/iothub-explorer>). DeviceExplorer utility is used as reference in this Quickstart Guide.
- To test pre-compiled binaries with STM32ODE web dashboard: **Chrome** web browser (<https://www.google.com/chrome/> ); tested with Chrome version v56.0.2924.76

# FP-CLD-AZURE1. Sample applications

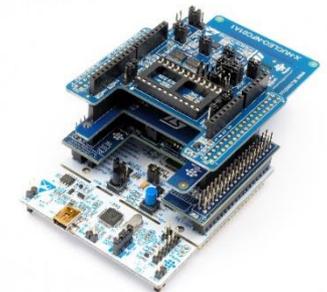
## Start coding in just a few minutes



FP-CLD-AZURE1 package structure



Register your device in Azure IoT, provision credentials in source code and recompile the project according to the selected IDE. Alternatively use available pre-compiled binaries

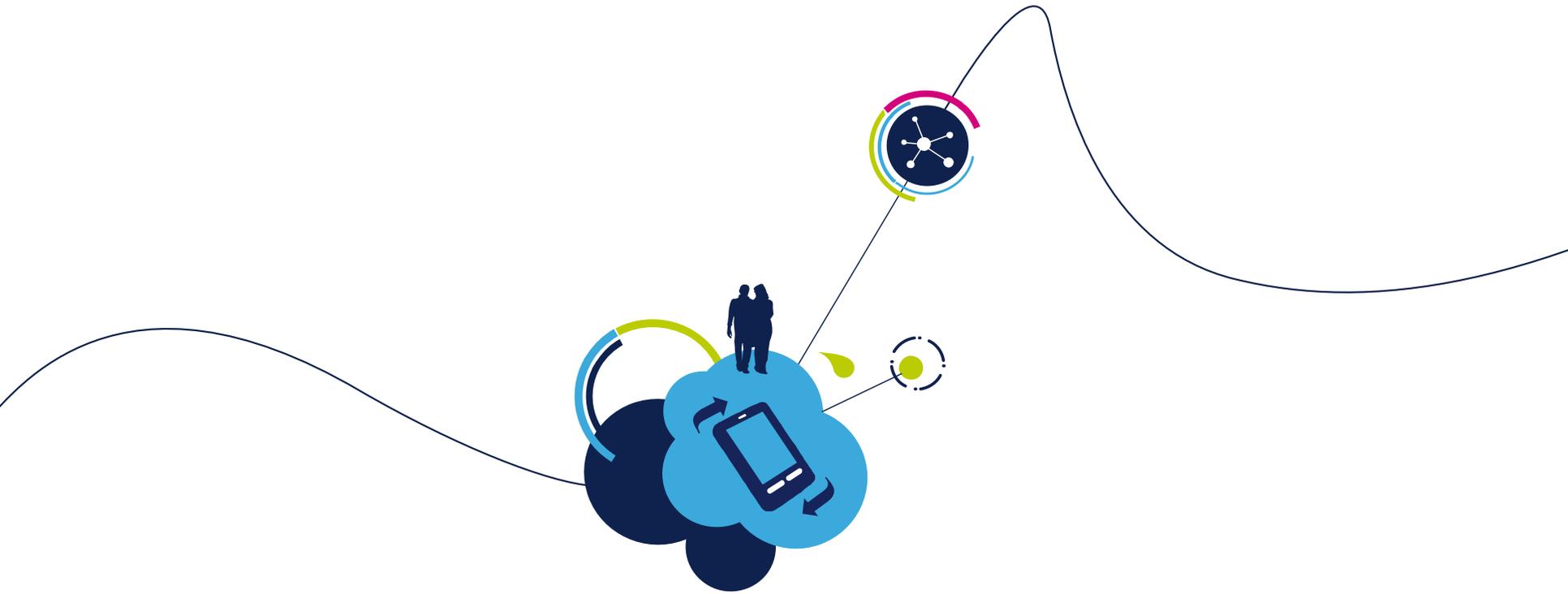


# FP-CLD-AZURE1. Sample applications for different platforms

## Azure\_Sns\_DM / Azure\_Motor

22

Application name and platform supported	Main Features	Precompiled binary name for usage with personal Azure account	Precompiled binary name for usage with personal IoT Central account	Precompiled binary name for usage with ST Web dashboard
<b>Azure_Sns_DM</b> (NUCLEO-F401RE)	Sensors data telemetry / device control	Azure_Sns_DM.bin	X	Azure_Sns_DM_Web.bin
<b>Azure_Sns_DM</b> (NUCLEO-L476RG)	Sensors data telemetry / device control / FOTA	Azure_Sns_DM_BL.bin	Azure_Sns_DM_BL_IoTCentral.bin	Azure_Sns_DM_BL_Web.bin
<b>Azure_Sns_DM</b> (NUCLEO-F429ZI)	Sensors data telemetry / device control / FOTA	Azure_Sns_DM_BL.bin	X	Azure_Sns_DM_BL_Web.bin
<b>Azure_Sns_DM</b> (B-L475E-IOT01A)	Sensors data telemetry / device control / FOTA	Azure_Sns_DM_BL.bin	Azure_Sns_DM_BL_IoTCentral.bin	Azure_Sns_DM_BL_Web.bin
<b>Azure_Motor</b> (NUCLEO-L476RG)	Device control / Motor Control	Azure_Motor.bin	X	Azure_Motor_Web.bin
<b>Azure_Motor</b> (B-L475E-IOT01A)	Device control / Motor Control	Azure_Motor.bin	X	Azure_Motor_Web.bin



# Test FP-CLD-AZURE1 with personal Azure account

## Create Azure IoT Hub and generate device connection string (1/4)

- Enter Azure Portal with your credentials ( <https://azure.portal.com> )
- Create an instance of the Azure IoT Hub following instruction provided here: [https://github.com/Azure/azure-iot-device-ecosystem/blob/master/setup\\_iothub.md](https://github.com/Azure/azure-iot-device-ecosystem/blob/master/setup_iothub.md)
- Note down in a text editor your *IoT Hub connection string*

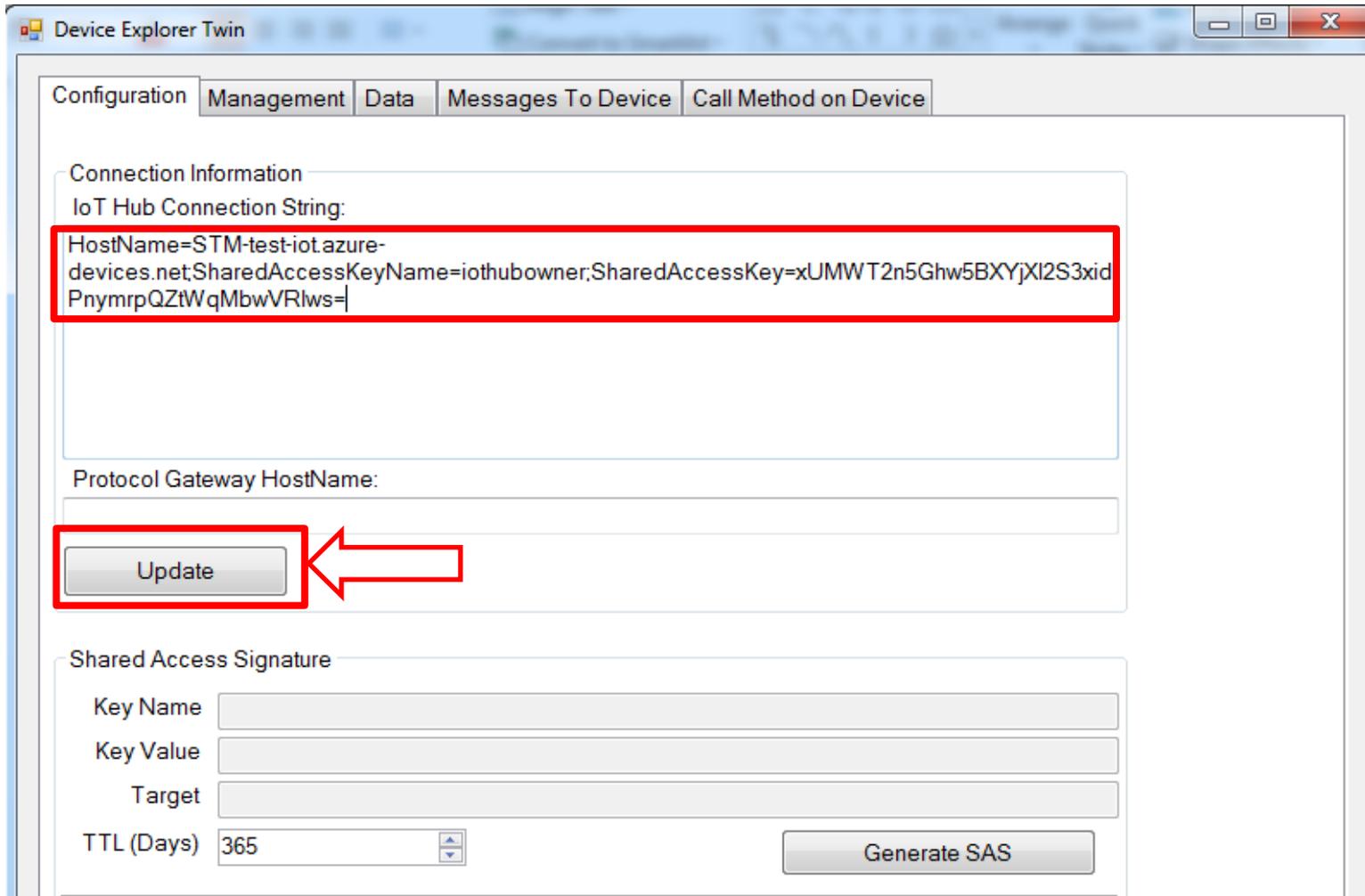
The screenshot shows the Azure IoT Hub Shared access policies configuration page. The 'iothubowner' policy is selected, and its permissions are listed in the table below:

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

The 'Connection string—secondary key' field is highlighted with a red box and an arrow pointing to it. The connection string is: `HostName=STM-test-iot.azure-devices.net`

## Create Azure IoT Hub and generate device connection string (2/4)

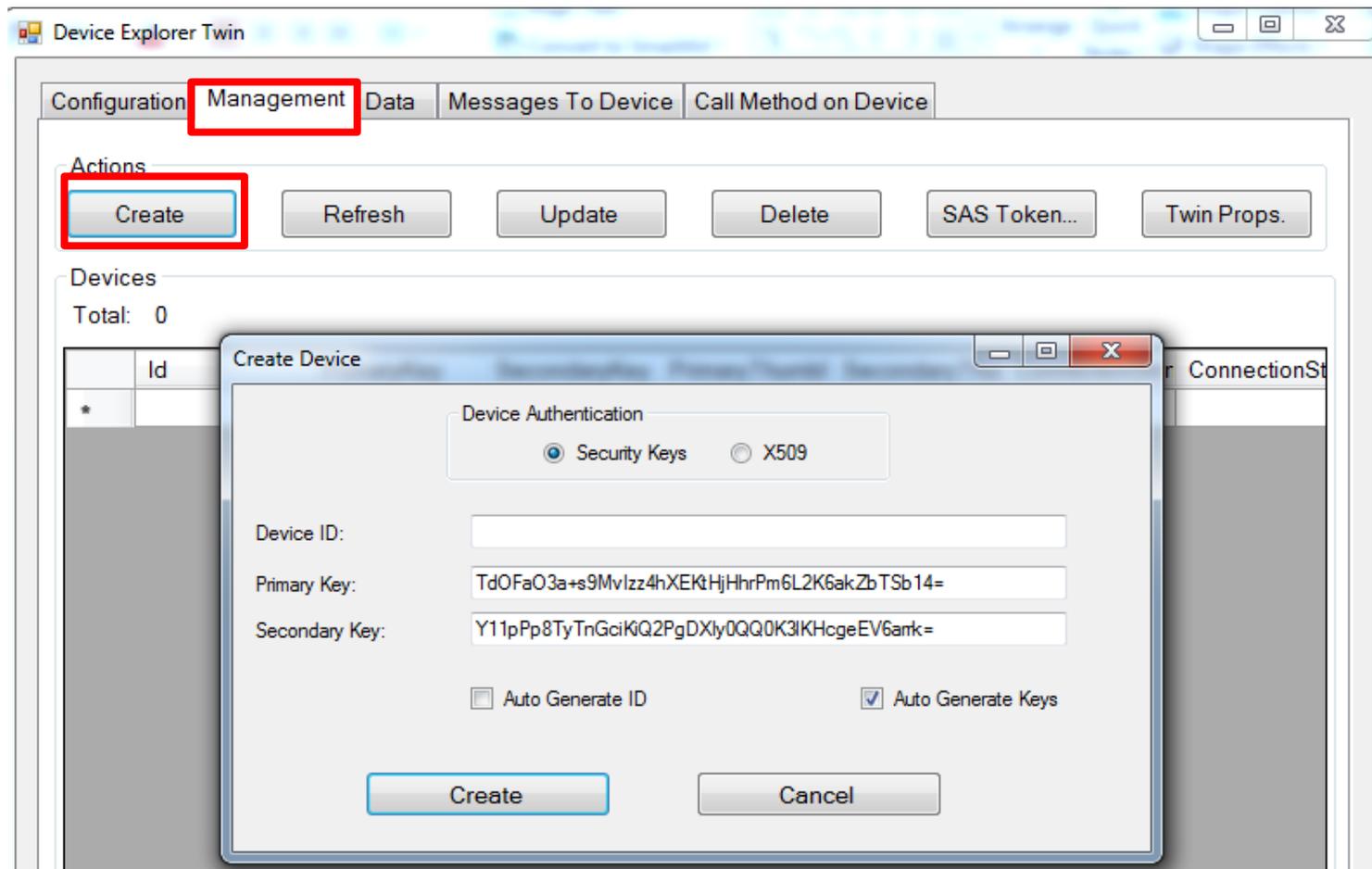
- Launch Device Explorer; open **Configuration** tab, paste the *IoT Hub connection string* and then click on **Update**



The screenshot shows the 'Device Explorer Twin' application window with the 'Configuration' tab selected. The 'IoT Hub Connection String' field is highlighted with a red box and contains the text: `HostName=STM-test-iot.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=xJMWT2n5Ghw5BXYjXl2S3xidPnymrpQZtWqMbwVRlws=`. Below this field is the 'Protocol Gateway HostName' field. The 'Update' button is also highlighted with a red box and has a red arrow pointing to it from the right. At the bottom of the window, there is a 'Shared Access Signature' section with fields for 'Key Name', 'Key Value', 'Target', and 'TTL (Days)' (set to 365), and a 'Generate SAS' button.

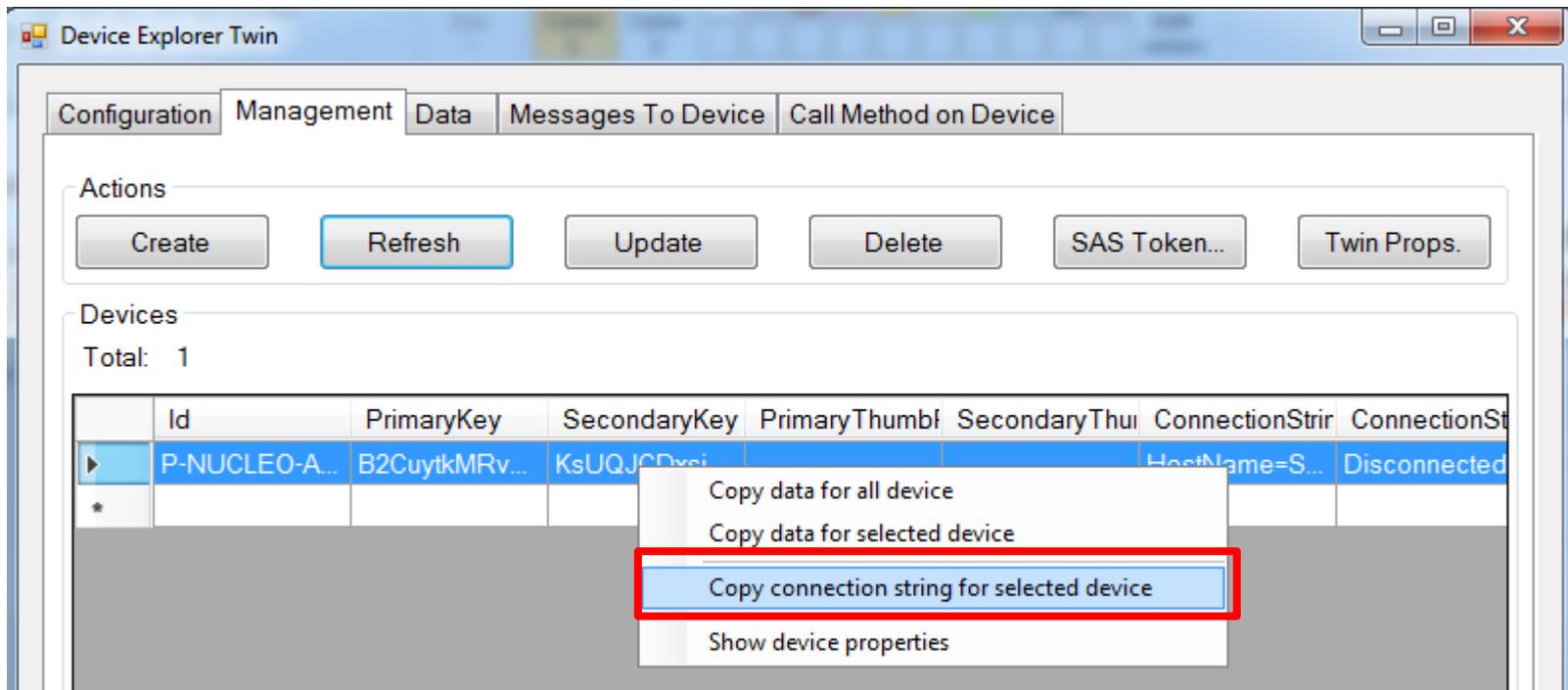
## Create Azure IoT Hub and generate device connection string (3/4)

- In **Management** tab click on Create and then insert a unique device ID for your STM32 Nucleo board. An entry for your device is created and listed.



## Create Azure IoT Hub and generate device connection string (4/4)

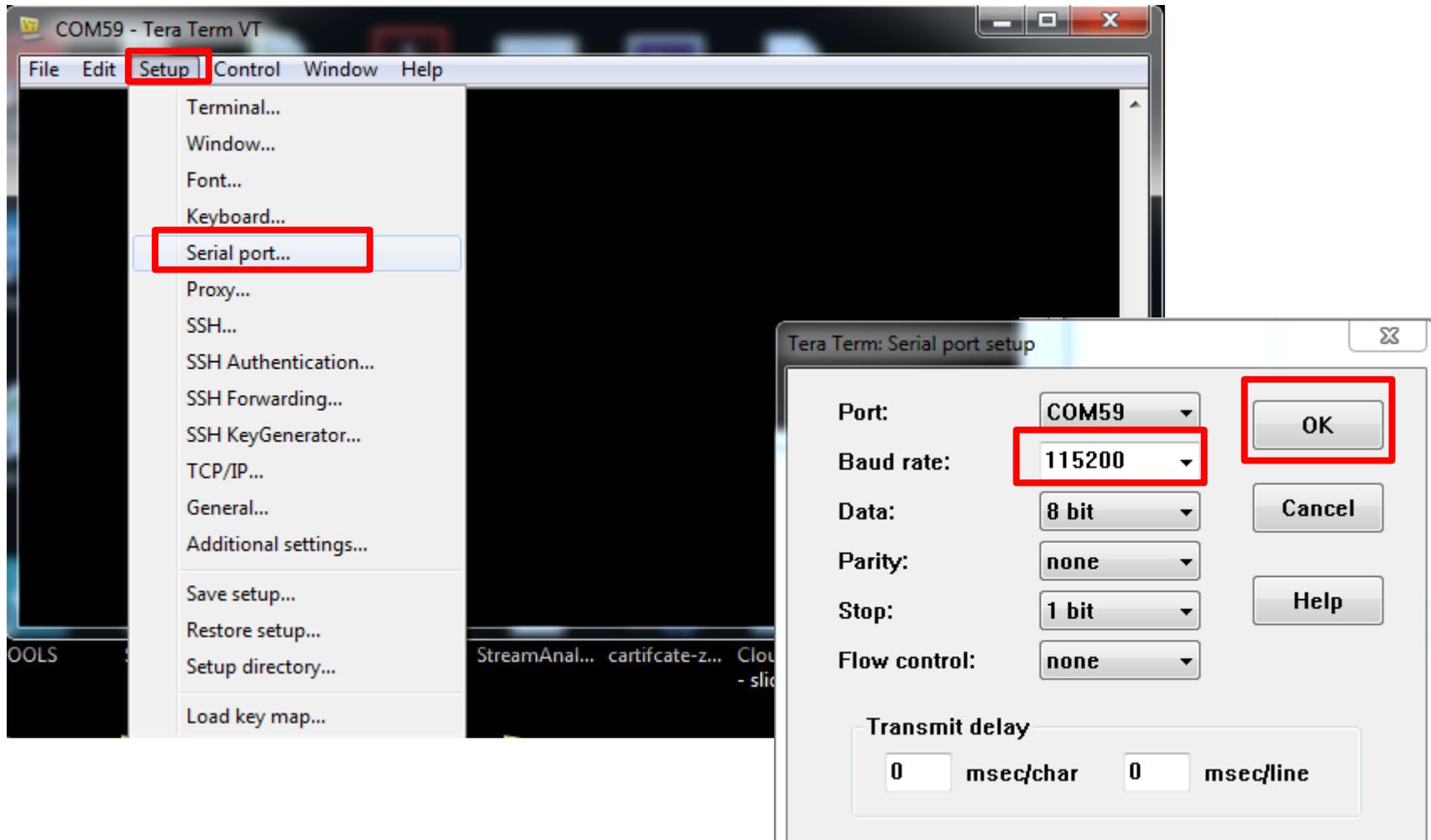
- In **Management** tab select the device just created, use the mouse right click for context menu then select “**Copy connection string for selected device**”. Note down the connection string in a text editor for later usage.



# FP-CLD-AZURE1. Step by step setup in details

## Launch sample application. Configure Serial Terminal

- Open serial terminal then configure baud rate speed to 115200 (**Setup** → **Serial port** in TeraTerm).

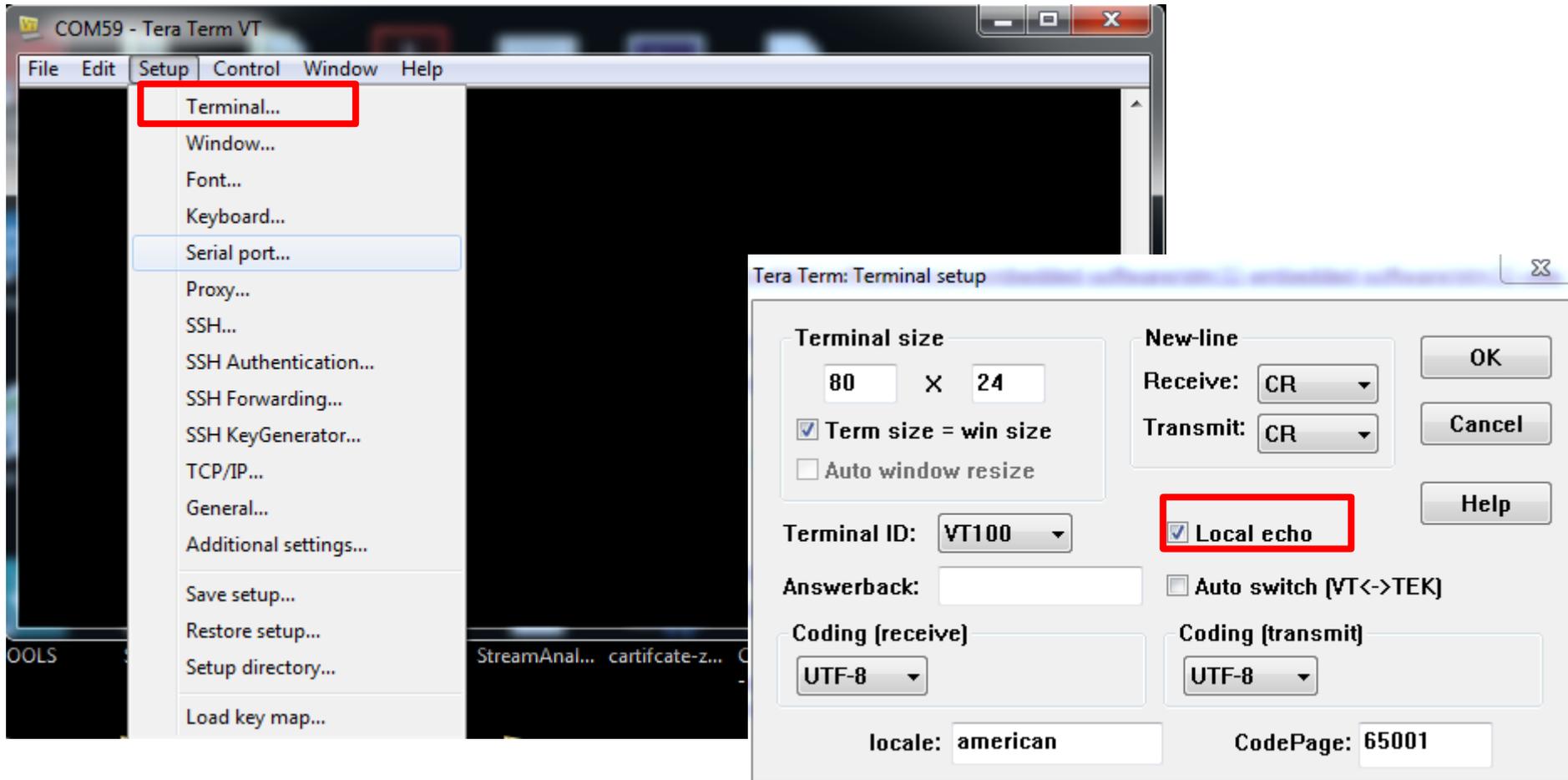


# FP-CLD-AZURE1. Step by step setup in details

## Launch sample application. Configure Serial Terminal

29

- Enable local echo in Terminal configuration (**Setup** → **Terminal** in TeraTerm).

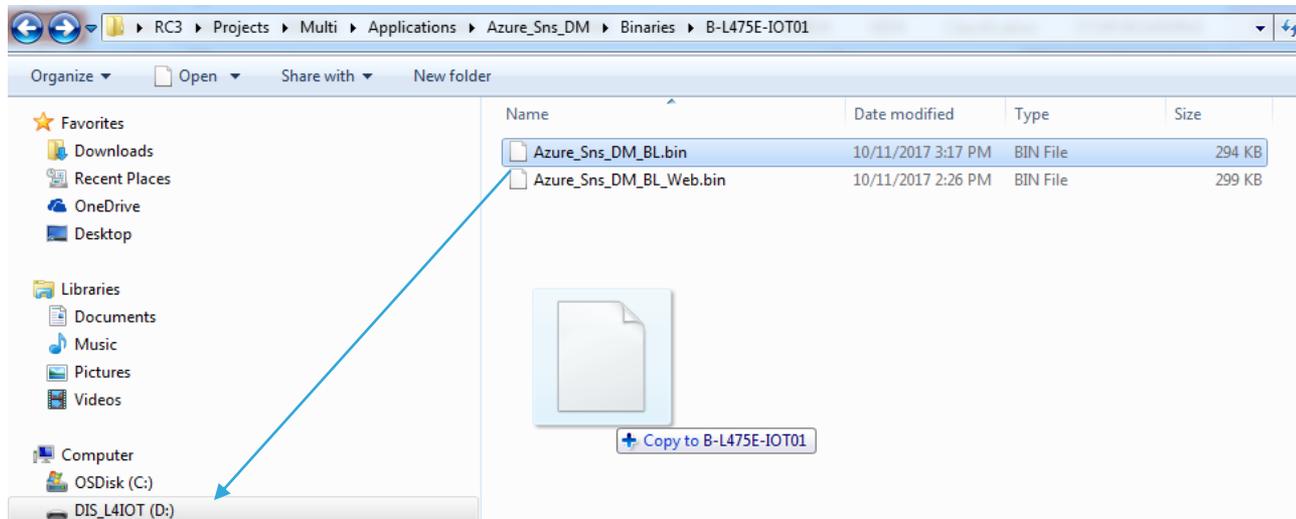


# FP-CLD-AZURE1. Step by step setup in details

## Launch sample application. Use pre-compiled binaries

30

- Depending on application and hardware combination used, configurable pre-compiled binaries can be found inside folders:
  - Projects/Multi/Applications/Azure\_Sns\_DM/Binaries/STM32F401RE-Nucleo/Azure\_Sns\_DM.bin
  - Projects/Multi/Applications/Azure\_Sns\_DM/Binaries/STM32F476RG-Nucleo/Azure\_Sns\_DM\_BL.bin
  - Projects/Multi/Applications/Azure\_Sns\_DM/Binaries/STM32F429ZI-Nucleo/Azure\_Sns\_DM\_BL.bin
  - Projects/Multi/Applications/Azure\_Sns\_DM/Binaries/B-L475E-IOT01/Azure\_Sns\_DM\_BL.bin
  - Projects/Multi/Applications/Azure\_Motor/Binaries/STM32F476RG-Nucleo/Azure\_Motor.bin
  - Projects/Multi/Applications/Azure\_Motor/Binaries/B-L475E-IOT01/Azure\_Motor.bin
- To start the application, simply connect to the laptop your board and drag the binary

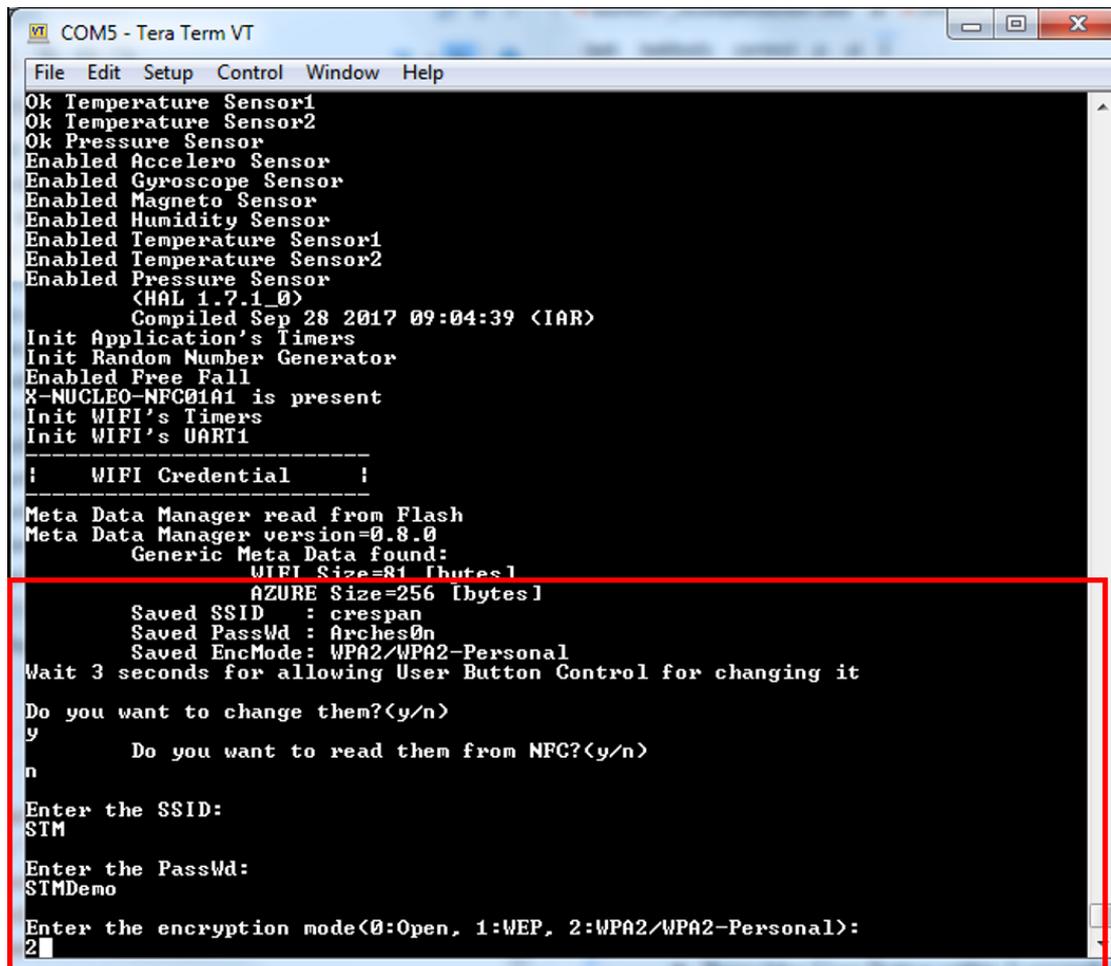


# FP-CLD-AZURE1. Step by step setup in details

## Configure Wi-Fi parameters (NUCLEO-L476RG/F401RE, B-L475E-IOT01A)

31

- Open serial terminal to visualize the log of messages
- Default values for Wi-Fi SSID and PWD can be modified after pressing once USER button within 3 seconds. Press n when asked to read from NFC, then enter SSID, PWD and Encryption mode (WPA2/WPA2-Personal) when requested



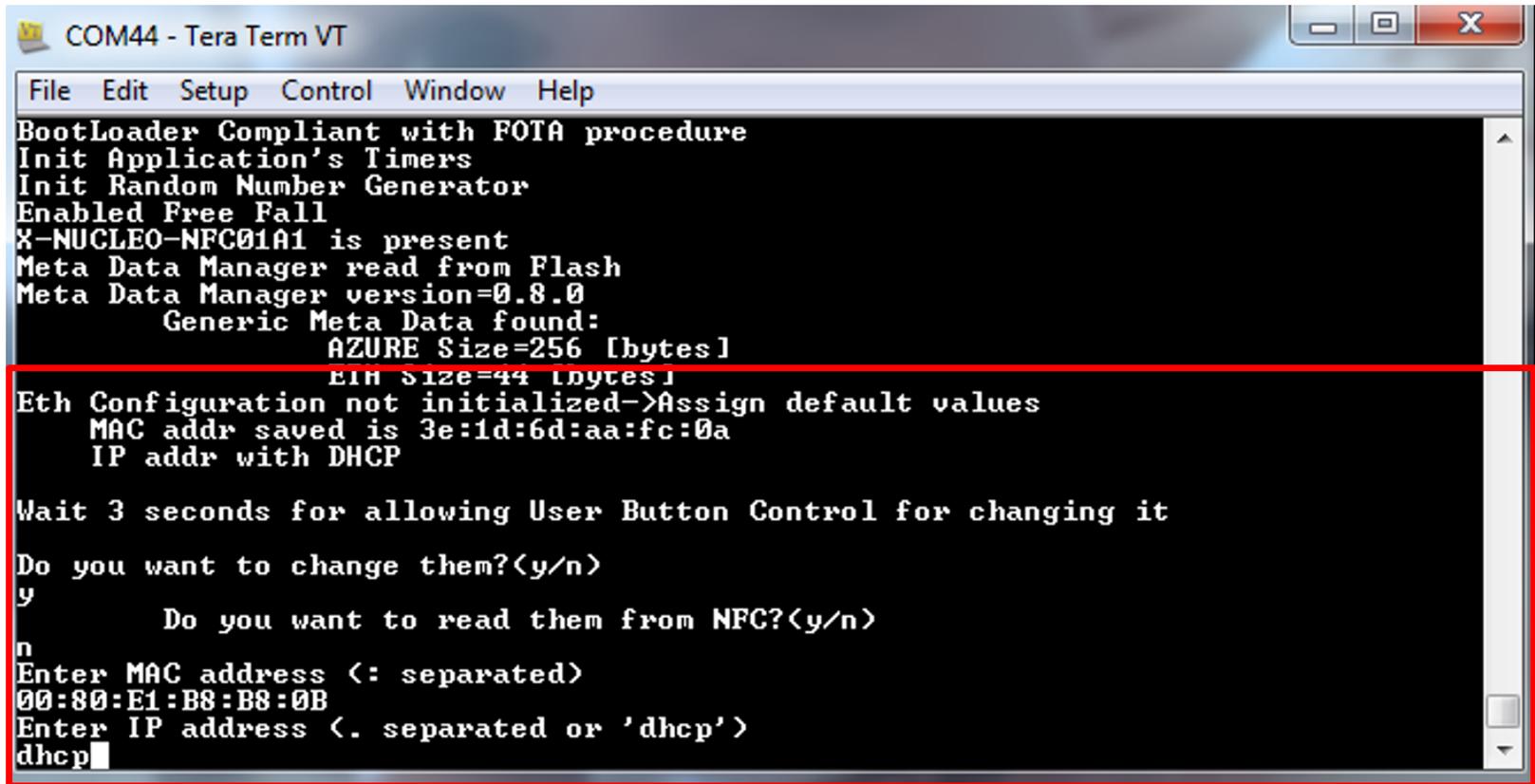
```
COM5 - Tera Term VT
File Edit Setup Control Window Help
Ok Temperature Sensor1
Ok Temperature Sensor2
Ok Pressure Sensor
Enabled Accelero Sensor
Enabled Gyroscope Sensor
Enabled Magneto Sensor
Enabled Humidity Sensor
Enabled Temperature Sensor1
Enabled Temperature Sensor2
Enabled Pressure Sensor
(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
```

# FP-CLD-AZURE1. Step by step setup in details

## Configure Ethernet parameters (NUCLEO-F429ZI)

32

- Open serial terminal to visualize the log of messages
- Default Ethernet configuration can be modified after pressing once USER button within 3 seconds. Press n when asked to read from NFC; enter the MAC address and then dhcp for automatic IP configuration or static IP and gateway addresses



```
COM44 - Tera Term VT
File Edit Setup Control Window Help
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
```

# FP-CLD-AZURE1. Step by step setup in details

## Configure device connection string (all platforms)

33

- Once completed Wi-Fi or Ethernet configuration, you need to enter the Azure IoT Hub device connection string (default is NULL)
- Once inserted, the connection string is save into FLASH memory. Press the blue **User Button** within 3 seconds to change it.

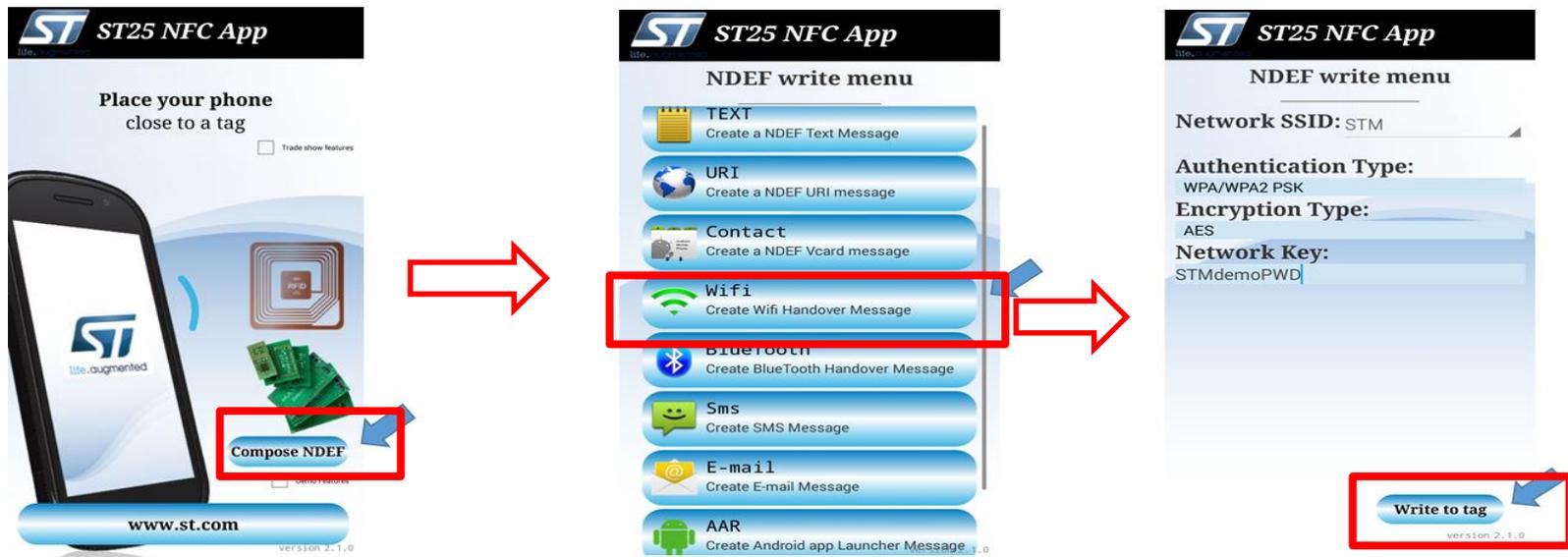
```
-----
:   WIFI Credential   :
-----
Meta Data Manager read from Flash
Meta Data Manager version=0.8.0
  Generic Meta Data found:
    WIFI Size=81 [bytes]
    AZURE Size=256 [bytes]
  Saved SSID   : STM
  Saved PassWd : STMDemo
  Saved EncMode: WPA2/WPA2-Personal
Wait 3 seconds for allowing User Button Control for changing it
-----
:   Connection String :
-----
  Saved Connection String :
    HostName=STM-test-iot.azure-devices.net;DeviceId=0080E1B8871F;Sh
aredAccessKey=zT+qkZUKn06Y8Mg5+nvqKx7rB1WUv4d3nwi0fbfhDC4=
Wait 3 seconds for allowing User Button Control for changing it
  Do you want to change it?(y/n)

```

# FP-CLD-AZURE1. Step by step setup in details

Optional: use NFC for device configuration. Wi-Fi parameters

- Launch the ST25 NFC mobile app and click the Compose NDEF button, then select in menu the Wi-Fi option
- Insert SSID and Password, then approach the mobile phone to the NFC tag and click on *Write to tag*



- Reset the board. Press the blue user button within 3 seconds, and press y when requested to read from NFC.

# FP-CLD-AZURE1. Step by step setup in details

Optional: use NFC for device configuration. Ethernet parameters

- Launch the ST25 NFC mobile app and click the Compose NDEF button.
- Click on the TEXT button and enter the MAC address as "MAC" followed 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. Then click on the "Write to tag" button with your mobile phone near the NFC expansion board on your system.



- Reset the board. Press the blue user button within 3 seconds, and press y when requested to read from NFC.

# FP-CLD-AZURE1. Step by step setup in details

Optional: use NFC for device configuration. Device connection string

- After Wi-Fi/Ethernet configuration, press again the blue user button within 3 seconds to update the IoT Hub device connection string.
- Go back to your mobile application, click on the **Text** button and paste the IoT Hub connection string



- Reset the board. In the serial terminal, press **y** when asked to read connection string from NFC

# FP-CLD-AZURE1. Step by step setup in details

## Visualize messages in DeviceExplorer (1/2)

- After device configuration, the sample application contact the IoT Hub
- Each 2 seconds, a message is created containing sensors data and transmitted to the IoT Hub using MQTT protocol. For each message transmitted, a confirmation acknowledge is received back by the device

```
ok reported state [1] naming
IoTHubClient_LL_SendEventAsync accepted message [1] for transmission to IoT Hub.
IoTHubClient_LL_SendEventAsync accepted message [2] for transmission to IoT Hub.
IoTHubClient_LL_SendEventAsync accepted message [3] for transmission to IoT Hub.
-->DeviceTwin Callback [1]: Status code = 204
Confirmation received for message [1] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
Confirmation received for message [2] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [4] for transmission to IoT Hub.
Confirmation received for message [3] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [5] for transmission to IoT Hub.
Confirmation received for message [4] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [6] for transmission to IoT Hub.
Confirmation received for message [5] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [7] for transmission to IoT Hub.
Confirmation received for message [6] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [8] for transmission to IoT Hub.
Confirmation received for message [7] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
IoTHubClient_LL_SendEventAsync accepted message [9] for transmission to IoT Hub.
Confirmation received for message [8] with result = IOTHUB_CLIENT_CONFIRMATION_SUCCESS
```

# FP-CLD-AZURE1. Step by step setup in details

## Visualize messages in DeviceExplorer (2/2)

- Open DeviceExplorer; select **Data** tab, then in the drop down menu for **Device Id**: select the id of your registered device. Then click on **monitor** to start visualizing the log of messages received by IoT Hub

The screenshot shows the 'Device Explorer Twin' application window. The 'Data' tab is selected and highlighted with a red box. The 'Monitoring' section contains the following fields:

- Event Hub: STM-test-iot
- Device ID: P-NUCLEO-AZURE1-EQ (highlighted with a red box)
- Start Time: 05/03/2017 12:50:51
- Consumer Group: \$Default
- Enable:

At the bottom of the 'Monitoring' section, the 'Monitor' button is highlighted with a red box. Below this section is the 'Event Hub Data' section, which displays a log of messages. Two messages are visible, both highlighted with a red box:

```
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'
```

# FP-CLD-AZURE1. Step by step setup in details

## Send Cloud to Device messages (1/3)

- Open Device Explorer; select **Message To Device** tab, then in the drop down menu for Device ID select the id of your registered device. Write a message, then click on **Send**. Message sent to device will be displayed in **Output** box

Configuration Management Data **Messages To Device** Call Method on Device

Send Message to Device:

IoT Hub: STM-test-iot

Device ID: P-NUCLEO-AZURE1-EQ

Message: {"Name": "LedOn", "Parameters": {}}

Add Time Stamp  Monitor Feedback Endpoint

Properties:

	Key	Value
*		

**Send** Clear

Output

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

# FP-CLD-AZURE1. Step by step setup in details

## Send Cloud to Device messages (2/3)

- Each message received by the device from IoT Hub is printed over serial terminal.
- By default the application can interpret the following Cloud to Device messages:
  - **Pause**: pause the application
  - **Play**: restart the application
  - **LedOn/LedOff**: turn on/off LED2 onboard STM32-Nucleo
  - **LedBlink**: LED2 onboard STM32-Nucleo will blink for each message received.
- Messages need to be typed in Device Explorer in the following format:
  - {"Name": "*Message*", "Parameters": {}} ), i.e.
    - {"Name": "Pause", "Parameters": {}}
    - {"Name": "Play", "Parameters": {}}
    - ...

# FP-CLD-AZURE1. Step by step setup in details

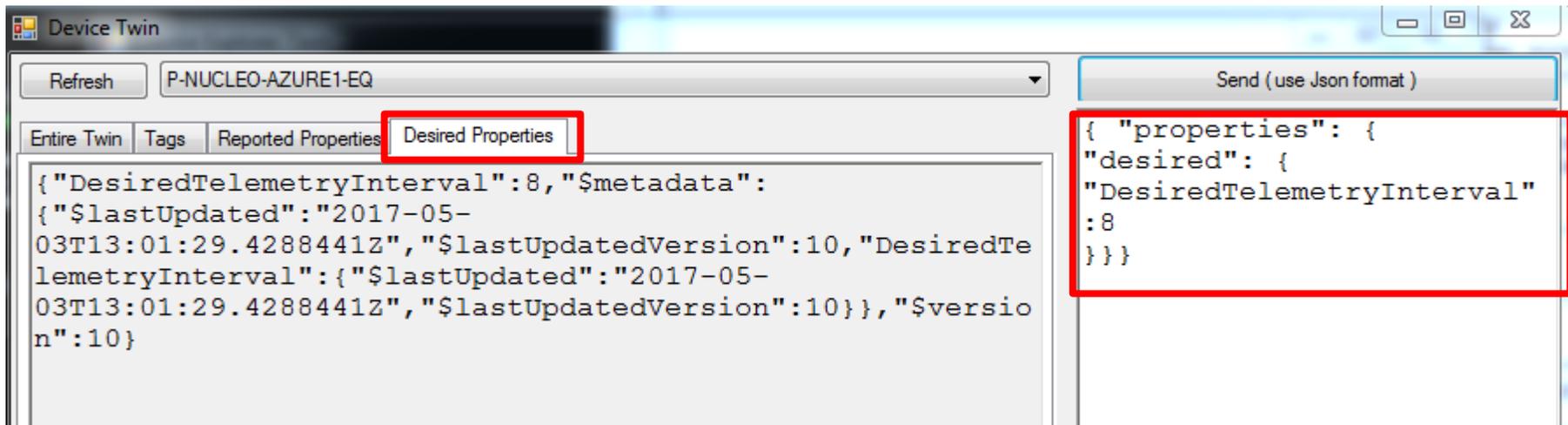
## Send Cloud to Device messages (3/3)

- For Azure\_Motor application, some more messages are interpreted and used by X-NUCLEO-IHM02A1
  - **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
- To learn more on the syntax for motor control commands, see the UM1963 at [X-CUBE-SPN2](#)
- Messages need to be typed in Device Explorer in the following format:
  - {"Name": "*Message*", "Parameters": {}} ), i.e.
    - {"Name" : "MoveMotor", "Parameters" : {"MotorNum" : 1, "Angle" : 45}}
    - ...

# FP-CLD-AZURE1. Step by step setup in details

## Change desired properties in DeviceExplorer

- In Device Explorer select **Management** tab, then **Twin Props** and the Id of your device
- In **Twin Props** select **Desired Properties** tab. Then type in the right window the new desired property
- Only DesiredTelemetryInterval is supported as desired property by the application



The screenshot shows the 'Device Twin' window in Device Explorer. The device ID is 'P-NUCLEO-AZURE1-EQ'. The 'Desired Properties' tab is active, displaying the following JSON:

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

The right-hand pane shows the JSON being sent, which is:

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

# FP-CLD-AZURE1. Step by step setup in details

Call direct method to force Firmware Update over-the-air (NUCLEO-L476RG, NUCLEO-F429ZI, B-L475E-IOT01A) (1/2)

43

Configuration Management Data Messages To Device **Call Method on Device** ← Select **“Call Method on Device”**

Call Method on Device

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

Call Method Cancel

Return status: 201

Return payload: "Initiating Firmware Update" ← Returned message from the device

Type in **FirmwareUpdate** method adding as a parameter the URL of web server hosting the new binary (the one reported here can be used for testing)

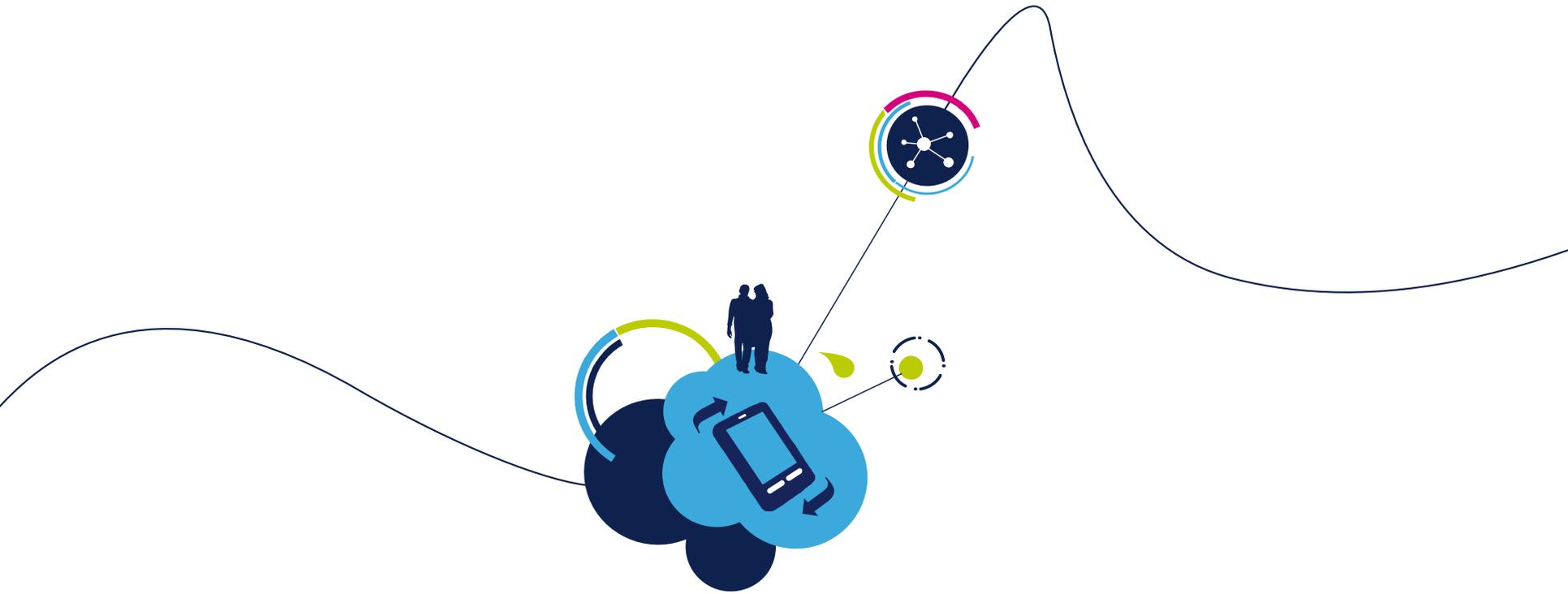
# FP-CLD-AZURE1. Step by step setup in details

Call direct method to force Firmware Update over-the-air (NUCLEO-L476RG, NUCLEO-F429ZI, B-L475E-IOT01A) (2/2)

44

```
Device Method called
Device Method name:      FirmwareUpdate
Device Method payload: <"FwPackageUri": "https://stm32blob.blob.core.windows.net/firmware-nucleo/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]
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_setopt
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
```

Erase FLASH memory and download new firmware  
When download is finished, restart the board and load the new firmware



# Test FP-CLD-AZURE1 with personal Microsoft IoT Central account

# FP-CLD-AZURE1. Step by step setup in details

Use pre-compiled binaries for Microsoft IoT Central (NUCLEO-L476RG, B-L475E-IOT01A) (1/2)

46

- Microsoft IoT Central is a fully managed SaaS (software-as-a-service) based on Azure that simplify the development of IoT Solutions. Learn more on IoT Central at <https://www.microsoft.com/en-us/iot-central>
- 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>.
- Once created the application, you can connect a device by simply clicking on **Connect this device** and then copying the resulting connection string

The screenshot displays the Microsoft IoT Central interface for a template named 'P-NUCLEO-AZURE1-IoTC (1.1.0)'. The interface includes a navigation menu with 'Measurements', 'Settings', 'Properties', 'Rules', and 'Dashboard'. A 'Temp AVERAGE' widget is visible on the left. In the top right corner, there is a 'Connect this device' button and a 'Delete' button. A red box highlights the 'Connect this device' button, and a red arrow points from it to a modal dialog box titled 'Connect this device'. The dialog box contains the following text: 'Use the following keys to connect your device to IoT Central. Learn more...'. It lists two connection strings: 'Primary connection string' and 'Secondary connection string', both with a 'Copy' button next to them. The 'Close' button is at the bottom of the dialog. The background shows a chart area with a 'Line' chart selected and a 'Show Tooltip' button. The bottom of the screen shows a timeline with timestamps: 6:38:36 PM, 6:41:09 PM, 6:43:42 PM, 6:46:15 PM, and 6:48:48 PM.

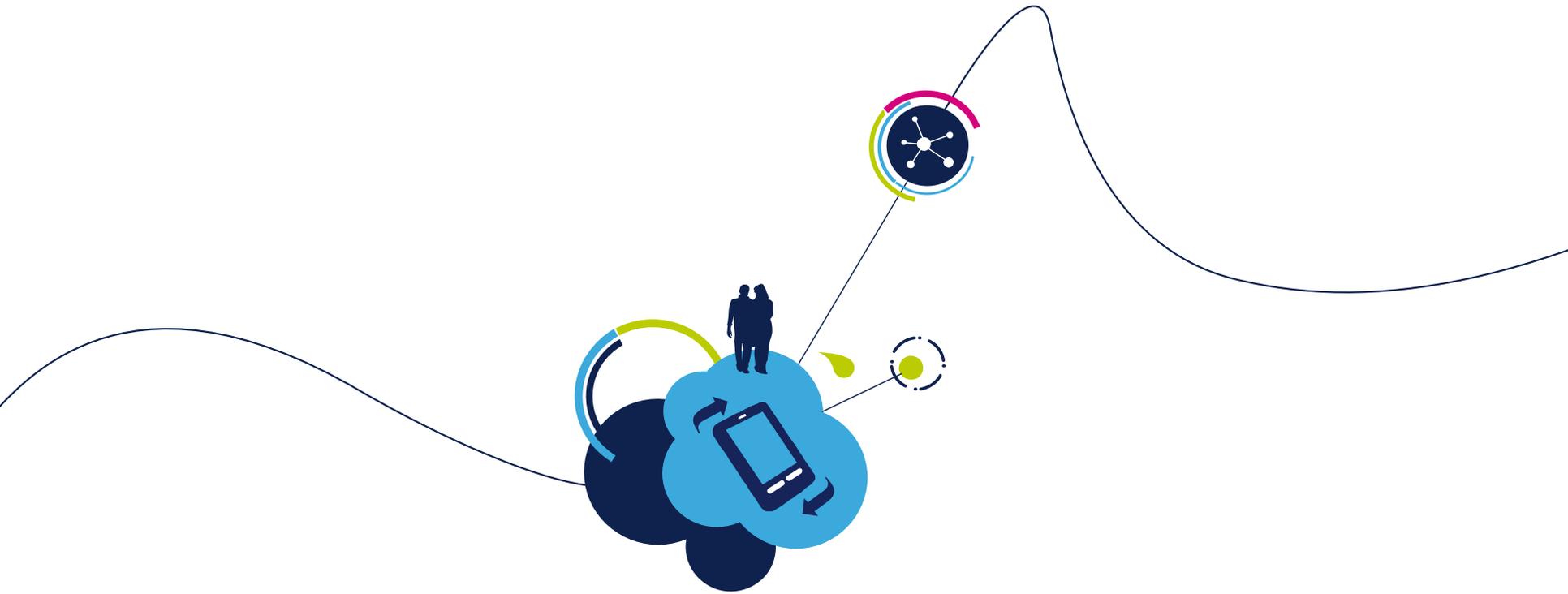
# FP-CLD-AZURE1. Step by step setup in details

Use pre-compiled binaries for Microsoft IoT Central (NUCLEO-L476RG, B-L475E-IOT01A) (2/2)

47

- Pre-compiled binaries for NUCLEO-L476RG/B-L475E-IOT01A are provided in folders:
  - Projects/Multi/Applications/Azure\_Sns\_DM/Binaries/STM32F476RG-Nucleo/Azure\_Sns\_DM\_BL\_IoTCentral.bin
  - Projects/Multi/Applications/Azure\_Sns\_DM/Binaries/B-L475E-IOT01/Azure\_Sns\_DM\_BL\_IoTCentral.bin
- Binaries for IoT Central can be used and configured by dragging the file to the connected board, setting Wi-Fi SSID and Password, and then entering IoT Central connection string in serial terminal or by using NFC
- After connecting with IoT Central, the application will start to transmit sensor data and device properties which can be visualize in IoT Central dashboard





# Test FP-CLD-AZURE1 with ST Web dashboard

## Use pre-compiled binaries for Azure Web dashboard (1/5)

- 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 for Azure\_Sns\_DM and Azure\_Motor applications to connect your boards with the web dashboard:
  - `Projects/Multi/Applications/Azure_Sns_DM/Binaries/STM32F401RE-Nucleo/Azure_Sns_Web.bin`
  - `Projects/Multi/Applications/Azure_Sns_DM/Binaries/STM32F476RG-Nucleo/Azure_Sns_DM_BL_Web.bin`
  - `Projects/Multi/Applications/Azure_Sns_DM/Binaries/STM32F429ZI-Nucleo/Azure_Sns_DM_BL_Web.bin`
  - `Projects/Multi/Applications/Azure_Sns_DM/Binaries/B-L475E-IOT01/Azure_Sns_DM_BL_Web.bin`
  - `Projects/Multi/Applications/Azure_Motor/Binaries/STM32F429ZI-Nucleo/Azure_Motor_Web.bin`
  - `Projects/Multi/Applications/Azure_Motor/Binaries/B-L475E-IOT01/Azure_Sns_Motor_Web.bin`
- To start the application, simply connect to the laptop your Nucleo board and drag one of the binaries according to your platform
- The usage of these binaries doesn't require the creation of an account in Azure since devices can automatically register and retrieve connection string from an ST account in Azure. To learn more on developed Azure solution, check the following link in github:
  - [https://github.com/MicrosoftBizSparkItaly/IoTCamp/blob/master/Docs/cloud\\_architecture\\_configuration.md](https://github.com/MicrosoftBizSparkItaly/IoTCamp/blob/master/Docs/cloud_architecture_configuration.md)

- Open serial terminal to visualize the log of messages. Configure Wi-Fi credentials as described in previous slides (using NFC or in the serial terminal).
- The board register itself to a custom IoT Hub using MAC address as device id and automatically retrieves the connection string.
- Web URL is printed over the serial terminal (otherwise go to <https://stm32ode.azurewebsites.net> and manually insert the MAC address of the board)

```
wifi started and ready...
WiFi module Configuration
WiFi MAC Address is: 00:80:E1:B8:0B
WiFi connected to AccessPoint
Init Real Time Clock 08-04-2017 10:13:01
Platform Init Done
Serializer Initialized
BoardType: 2
Ok io_interface_description
Ok xio_create
Ok xio_setopt
onOpenCompleteReg
Ok xio_open
Ok xio_send GET Request
[Registration]. Device not registered.
[Registration]. Device not yet registered. Sending registration request...
Ok xio_send POST request
[Registration]. Registered device and retrieved connection string.
Ok xio_close
onCloseCompleteReg callback
Ok xio_destroy
Board Registration Done

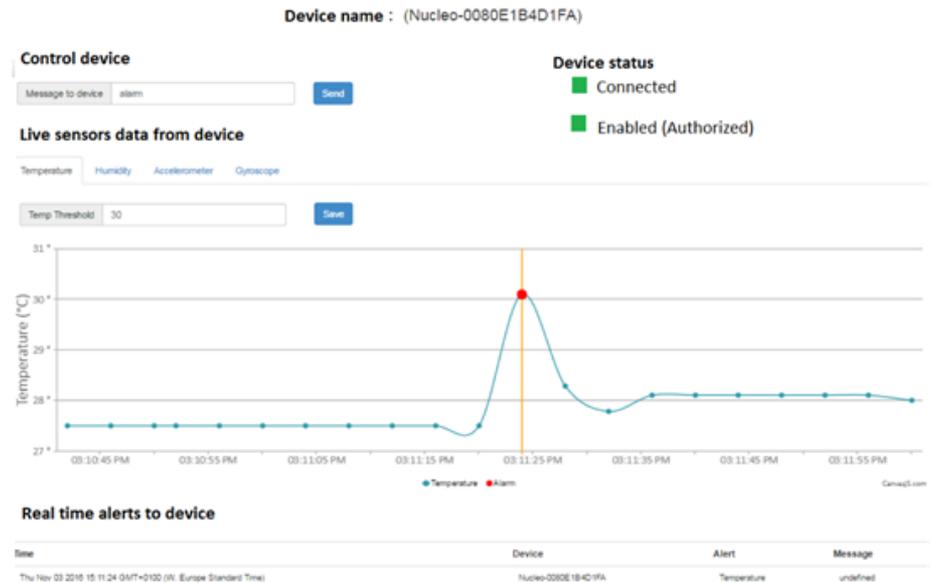
Connect to:
https://stm32ode.azurewebsites.net/Home/Index/0080E1B80B
```

# FP-CLD-AZURE1. Step by step setup in details

## Use pre-compiled binaries for Azure Web dashboard (3/5)

51

- If you are using an Android mobile, the web page can be automatically opened in your mobile browser by placing the device near to the NFC tag.



# FP-CLD-AZURE1. Step by step setup in details

## Use pre-compiled binaries for Azure Web dashboard (4/5)



Home About

STM32ODE IoT web dashboard

Go to Device Management view

Device name 0080E1B8B80B

Board type Nucleo-L476RG

Device management

Device status

- Connected
- Enabled

- Connected/Disconnected: indication of device connection to IoT Hub
- Enabled/Warning/Disabled: indication if maximum number of messages per day has been reached

Live sensor data from device

Select dataset to be visualized

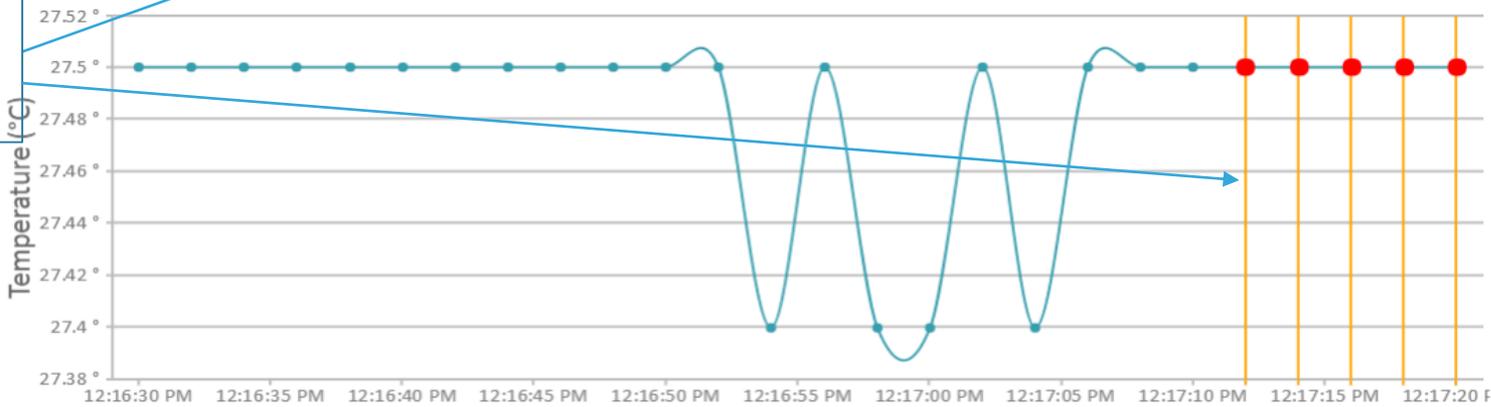
Temperature Humidity Accelerometer Gyroscope

Temp Threshold

25

Save threshold

Tab to set thresholds for alerts in graphs



life.ougmen

Temperature Alarm

CanvasJS.com

# FP-CLD-AZURE1. Step by step setup in details

## Use pre-compiled binaries for Azure Web dashboard (5/5)

Device name 0080E1B8B80B  
Board type Nucleo-L476RG

Go Back to telemetry view

Telemetry

Cloud to device messages

Visualize DeviceTwin:  
• Device status  
• Reported properties  
• Desired properties

**Twin** [v] [Properties]

Device Id  
0080E1B8B80B  
MAC address of this device

**Tags** [v] [Properties]

Board type  
2  
 Is demo device

**Properties** [v] [Properties]

**Desired** [v] [Properties]

Status  
enabled

**Reported** [v]

Azure Fw Version  
Azure\_Sns\_DM V3.1.0 SDK=1.1.16

Azure Status  
Running

**Control device**

Pause [v]  
Send message to device

Direct Methods called on device (Quit and Reboot)

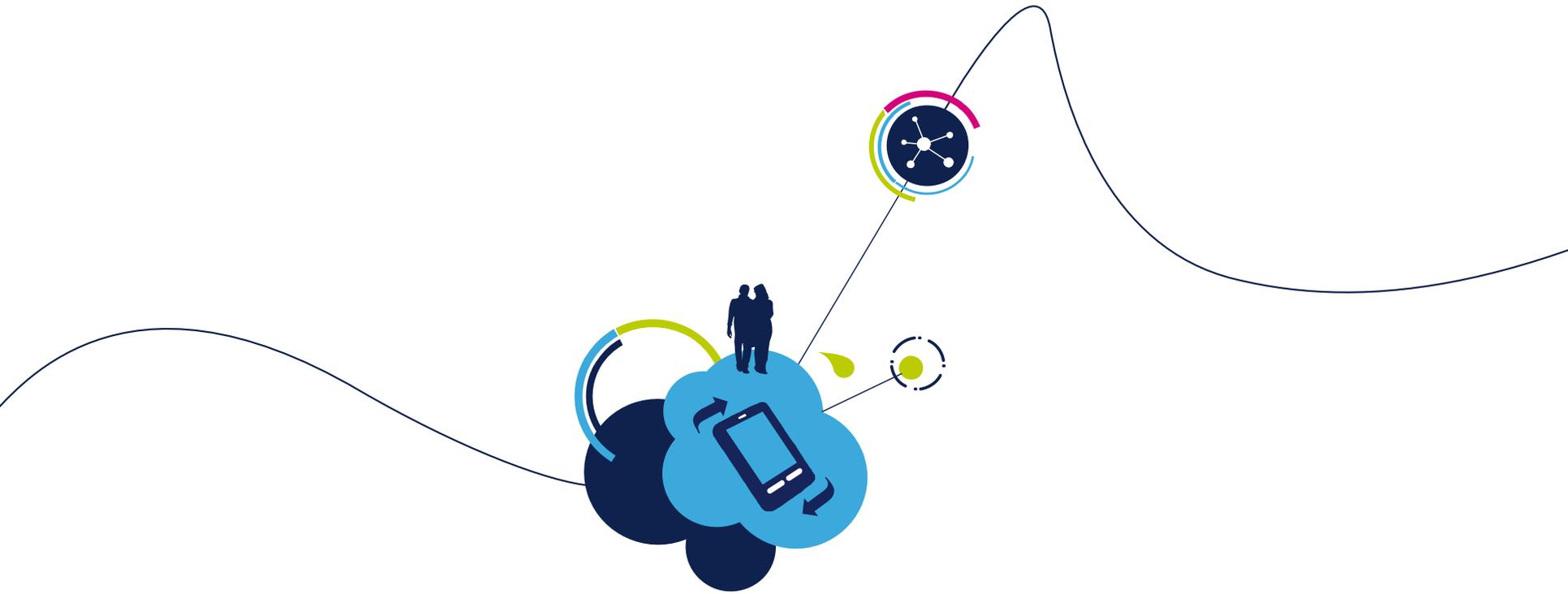
**Call Method on Device**

Reboot [v]  
Call Method

Firmware update (Nucleo-L476RG only)

**Firmware update**

Azure\_L476RG.bin [v]  
Force firmware update

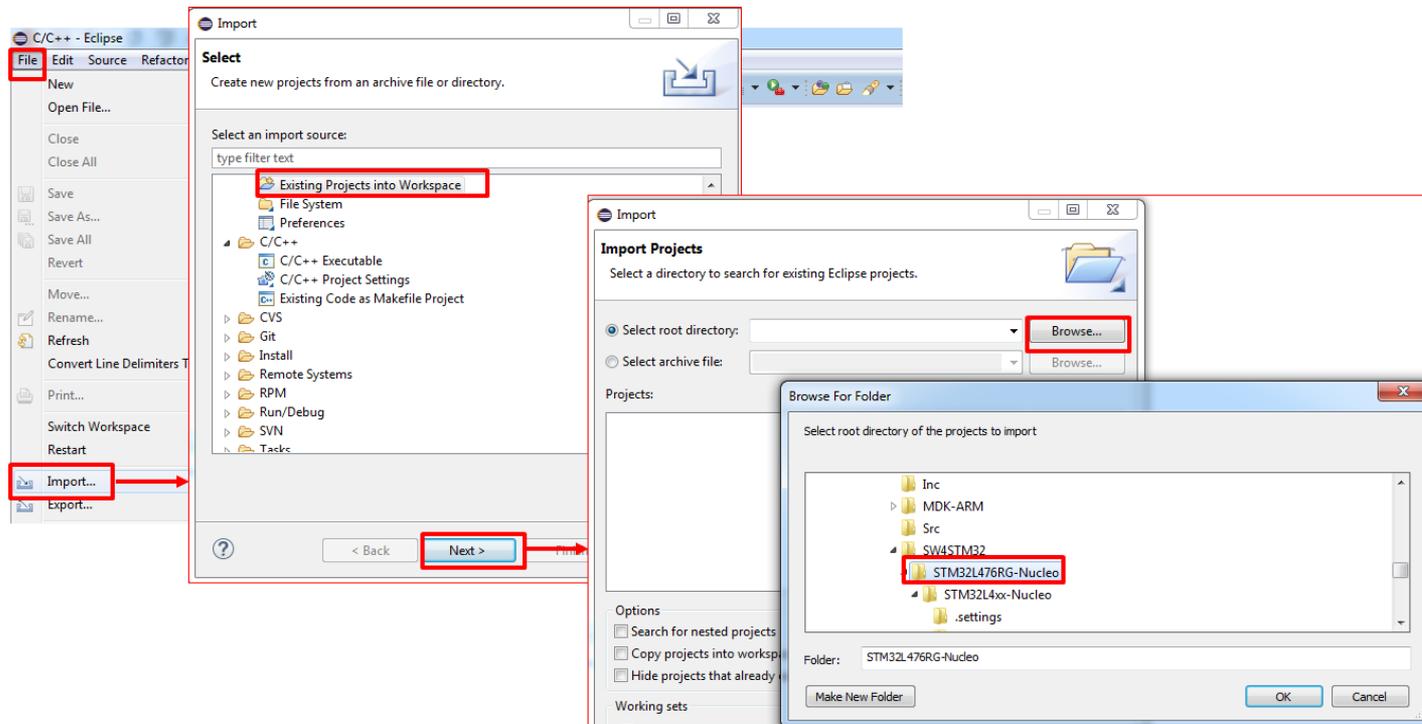


# Rebuild solution files in FP-CLD- AZURE1

# FP-CLD-AZURE1. Step by step setup in details

## Rebuild solution files (1/3)

- Open the available pre-configured projects according to the selected IDE and boards combination. Solution files for each IDE (IAR,ARM-MDK,SystemWorkbench4STM32) can be found in the folder *Projects\Multi\Application\Azure\_Sns\_DM* or *Projects\Multi\Application\Azure\_Motor*
- In SystemWorkbench, click on **File** → **Import** → **Existing Project into Workspace**. Then Browse project folder (SW4STM32/STM32XXXXX-Nucleo) and click on OK to finish



# FP-CLD-AZURE1. Step by step setup in details

## Rebuild solution files (2/3)

56

- To enter in source code **Wi-Fi parameters** (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*.
- To enter in source code **Ethernet parameters** (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.
- To enter **IoT Hub device connection string** open *azure1\_config.h*, uncomment and add a valid connection string for *AZUREDEVICECONNECTIONSTRING*, *i.e.*

```
#define AZUREDEVICECONNECTIONSTRING "HostName=trial.azurewebsites.net;DeviceId=Nucleo-Trial;SharedAccessKey=XXXXXXX"
```
- To enable communication with IoT Central (for NUCLEO-L476RG and B-L475E-IOT01), open *azure1\_config.h* then uncomment define *AZURE\_IOT\_CENTRAL*

# FP-CLD-AZURE1. Step by step setup in details

## Rebuild solution files (3/3)

57

- Rebuild the solution according to the selected IDE.
- In SystemWorkbench click on **Project** → **Build All**

The screenshot shows the Eclipse IDE interface. The 'Project' menu is open, and 'Build All' is highlighted with a red box. The CDT Build Console at the bottom shows the output of the build process, also highlighted with a red box.

```
C/C++ - STM32L4xx-Nucleo/Azure_Sns_DM/User/AzureClient_mqtt_DM_TM.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Open Project
Close Project
Build All Ctrl+B
Build Configurations
Build Project
Build Working Set
Clean...
Build Automatically
Make Target
C/C++ Index
Properties

Project Explorer
STM32L4xx-Nucleo
  Binaries
  Includes
  Azure_Sns_DM
    SW4STM32
      User
        agenttime_STM32Cube.c
        AzureClient_mqtt_DM_TM.c
        console.c
        HWAdvanceFeatures.c
        main.c
        OTA.c
        platform_STM32Cube.c
        RegistrationAgent.c
        socketio_STM32Cube.c
        STM32CubeRTCInterface.c
        stm32l4xx_hal_msp.c
        stm32l4xx_it.c
        tcpsocketSTM32Cube.c

CDT Build Console [STM32L4xx-Nucleo]
09:34:28 **** Build of configuration mbedTLS for project STM32L4xx-Nucleo ****
make all
'Building file: C:/Users/emanuele quacchio/Desktop/STM32/GITLP/AzureDM/Middlewares/Third_Parties/mbedtls/library/aes.c'
'Invoking: MCU GCC Compiler'
C:/Users/emanuele quacchio/Desktop/STM32/GITLP/AzureDM/Projects/Multi/Applications/Azure_Sns_DM/SW4STM32/STM32L476RG-Nucleo
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -std=c99 -DUSE_HAL_DRIVER -DSTM32_NUCLEO -DST
'Finished building: C:/Users/emanuele quacchio/Desktop/STM32/GITLP/AzureDM/Middlewares/Third_Parties/mbedtls/library/aes.c'
'Building file: C:/Users/emanuele quacchio/Desktop/STM32/GITLP/AzureDM/Middlewares/Third_Parties/mbedtls/library/aesni.c'
'Invoking: MCU GCC Compiler'
C:/Users/emanuele quacchio/Desktop/STM32/GITLP/AzureDM/Projects/Multi/Applications/Azure_Sns_DM/SW4STM32/STM32L476RG-Nucleo
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -std=c99 -DUSE_HAL_DRIVER -DSTM32_NUCLEO -DST
```

# FP-CLD-AZURE1. Step by step setup in details

## Flash binary to NUCLEO-F401RE board

58

- For the NUCLEO-F401RE platform, which does not support the firmware update-over-the-air procedure and does not require to install a BootLoader, the firmware can be written to the microcontroller from the IDE
- In SystemWorkbench click on **Run** → **Run Configuration**. Then select **STM32F4xx-Nucleo mbedTLS** and type in mbedTLS\STM32F4xx-Nucleo.elf; then click on **Run**.

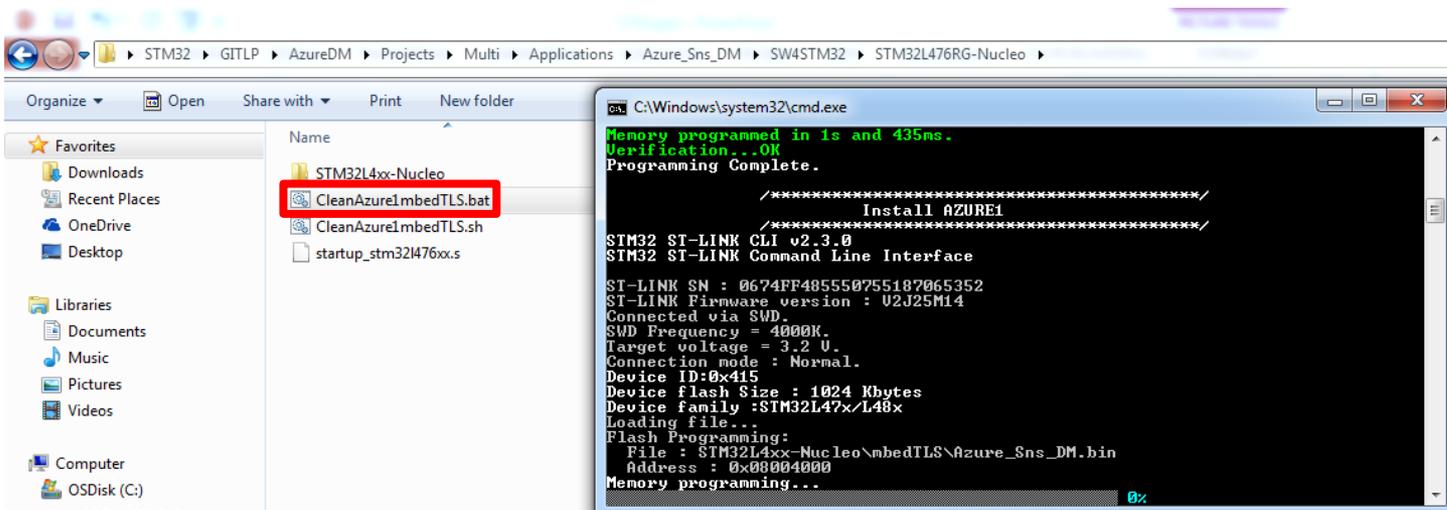
The screenshot displays the IDE interface with the following elements:

- Run Menu:** A dropdown menu is open from the 'Run' button in the top toolbar. The 'Run Configuration...' option is highlighted with a red box.
- Run Configurations Dialog:** A dialog box titled 'Run Configurations' is open. The 'Name' field is 'STM32F4xx-Nucleo mbedTLS'. The 'C/C++ Application' field contains 'mbedTLS\STM32F4xx-Nucleo.elf', which is highlighted with a red box. The 'Project' field is 'STM32F4xx-Nucleo'. The 'Build configuration' is set to 'Select Automatically'. The 'Run' button at the bottom right of the dialog is highlighted with a red box.
- Project Explorer:** The left sidebar shows the project structure. The 'User' folder contains several files, including 'main.c' and 'platform\_STM32Cube.c'. The 'Run As' option in the Run menu is also highlighted with a red box.
- Debugger Console:** The bottom of the window shows the 'Debug Console' with a search filter 'Filter matched 6 of 9 items'.

# FP-CLD-AZURE1. Step by step setup in details

## Flash binary and Bootloader (1/2)

- For NUCLEO-F476RG, NUCLEO-F429ZI, B-L475E-IOT0Ax platforms, once the code has been recompiled, it is necessary to use external scripts to flash the binary together with the bootloader
- Bootloader is necessary to implement Azure IoT device management primitives together with Firmware update application example
- Windows scripts based on ST-LINK command line are available according to the board used and IDE selected (i.e. for SW4STM32):
  - Projects/Multi/Applications/Azure\_Sns\_DM/SW4STM32/B-L475E-IOT01/CleanAzure\_Sns\_DM.bat
  - Projects/Multi/Applications/Azure\_Sns\_DM/SW4STM32/STM32F429ZI-Nucleo/CleanAzure\_Sns\_DM.bat
  - Projects/Multi/Applications/Azure\_Sns\_DM/SW4STM32/STM32L476RG-Nucleo/CleanAzure\_Sns\_DM.bat
- Double click on .bat script to flash bootloader and binary to the device



# FP-CLD-AZURE1. Step by step setup in details

## Flash binary and Bootloader (2/2)

- For the Linux/OSx operating system there is a similar script, CleanAzure1mbedTLS.sh, that uses OpenOCD command line
- The script is included only in the System Workbench folder and requires to set:
  - 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.win32_1.10.0.201607261143/tools/openocd/" OpenOCD_DIR=""
```

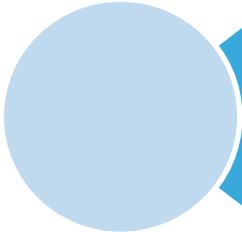
# 2) Set the installation path for stm32 OpenOCD scripts # example:

```
#OpenOCD_CFC="C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.debug_1.10.0.201607251855/resources/openocd/scripts" OpenOCD_CFC=""
```

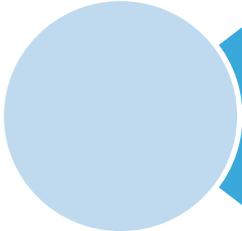
# 3) Only for Linux/iOS add openocd library path to \_LIBRARY\_PATH: # For iOS example:

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

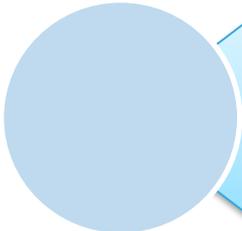
```
#export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${OpenOCD_DIR}lib/"
```



FP-CLD-AZURE: STM32 ODE function pack for IoT node with Wi-Fi or Ethernet, NFC and sensors, connected to Microsoft Azure cloud  
Hardware and Software overview



Setup & Demo Examples  
Documents & Related Resources



STM32 Open Development Environment: Overview

All documents are available in the DESIGN tab of the related products webpage

## FP-CLD-AZURE1:

- **DB2891:** STM32 ODE function pack for IoT node with Wi-Fi, NFC and sensors connected to Microsoft Azure cloud– **databrief**
- **UM2043:** Getting started with the FP-CLD-AZURE1 software for IoT node with Wi-Fi, NFC and sensors, connected to Microsoft Azure cloud – **user manual**
- Software setup file

## X-NUCLEO-NFC01A1:

- Gerber files, BOM, Schematic
- **DB2353:** Dynamic NFC tag expansion board based on M24SR for STM32 Nucleo – **databrief**
- **AN4624:** Getting started with the STM32 Nucleo and the M24SR expansion board X-NUCLEO-NFC01A1 – **application note**
- **UM1793:** Dynamic NFC tag expansion board based on M24SR for STM32 Nucleo – **user manual**

## X-NUCLEO-IDW01M1:

- Gerber files, BOM, Schematic
- **DB2726:** Wi-Fi expansion board based on SPWF01SA module for STM32 Nucleo – **databrief**
- **UM1973:** Getting started with the X-CUBE-WIFI1 Wi-Fi functions and applications software expansion for STM32Cube – **user manual**
- **UM1765:** Getting started with X-NUCLEO-IDW01M1 Wi-Fi expansion board based on SPWF01SA module for STM32 Nucleo – **user manual**

## X-NUCLEO-IKS01A1:

- Gerber files, BOM, Schematic
- **DB10619:** Motion MEMS and environmental sensor expansion board for STM32 Nucleo – **product specification**
- **UM1820:** Getting started with motion MEMS and environmental sensor expansion board for STM32 Nucleo – **user manual**

## X-NUCLEO-IKS01A2:

- Gerber files, BOM, Schematic
- **DB3009:** Motion MEMS and environmental sensor expansion board for STM32 Nucleo – **product specification**
- **UM2121:** Getting started with motion MEMS and environmental sensor expansion board for STM32 Nucleo – **user manual**

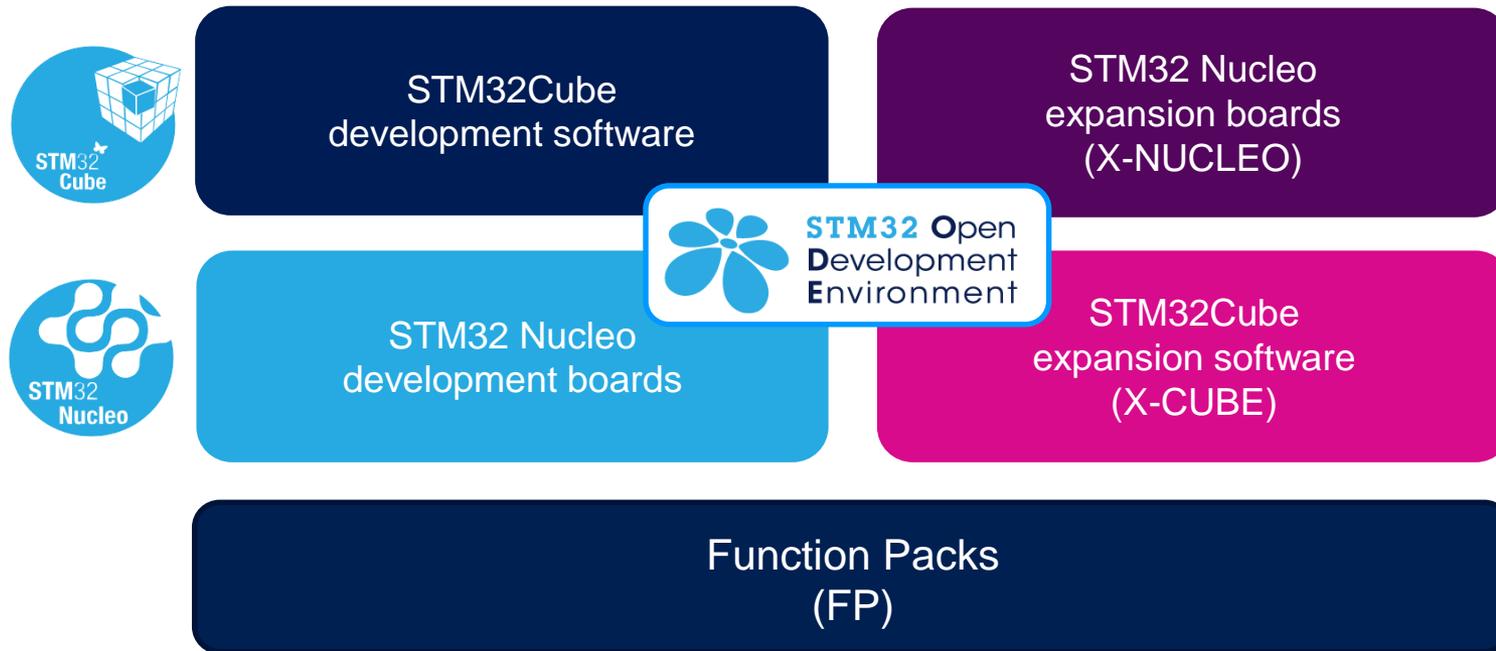


Consult [www.st.com](http://www.st.com) for the complete list

# STM32 Open Development Environment

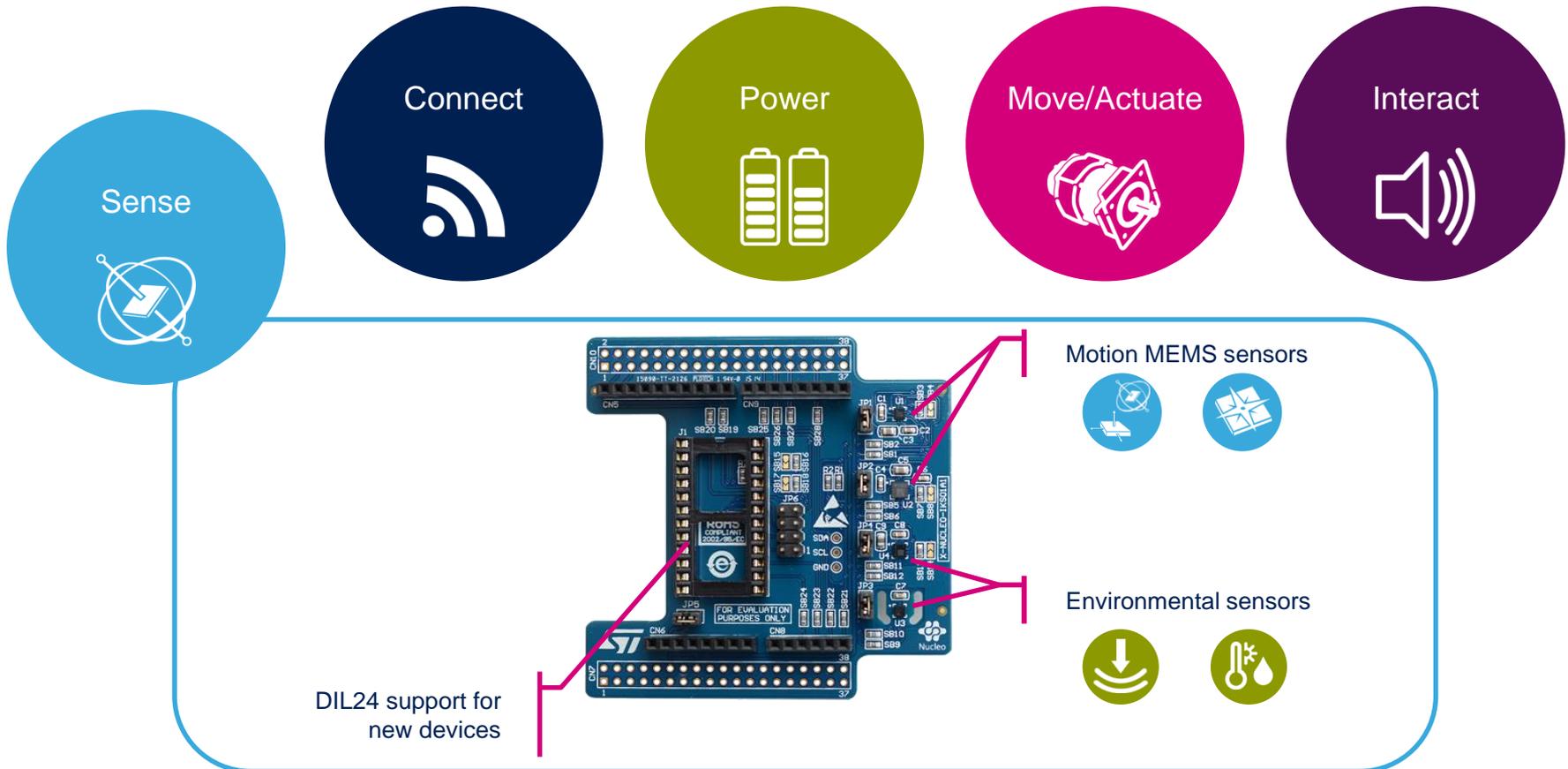
## Fast, affordable Prototyping and Development

- The STM32 Open Development Environment (ODE) consists of a set of stackable boards and a modular open SW environment designed around the STM32 microcontroller family.



# STM32 Nucleo Expansion Boards (X-NUCLEO)

- Boards with additional functionality that can be plugged directly on top of the STM32 Nucleo development board directly or stacked on another expansion board.



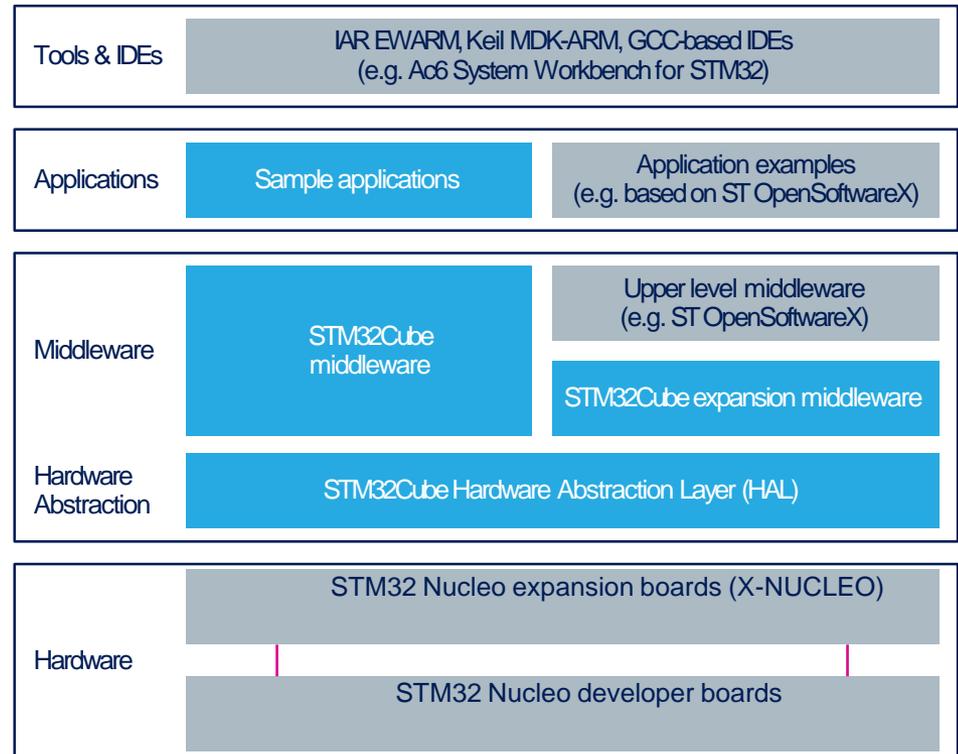
Example of STM32 expansion board (X-NUCLEO-IKS01A1)

# STM32 Open Development Environment

## Software components

66

- **STM32Cube software (CUBE)** - A set of free tools and embedded software bricks to enable fast and easy development on the STM32, including a Hardware Abstraction Layer and middleware bricks.
- **STM32Cube expansion software (X-CUBE)** - Expansion software provided free for use with the STM32 Nucleo expansion board and fully compatible with the STM32Cube software framework. It provides abstracted access to expansion board functionality through high-level APIs and sample applications.



- **Compatibility with multiple Development Environments** - The STM32 Open Development Environment is compatible with a number of IDEs including IAR EWARM, Keil MDK, and GCC-based environments. Users can choose from three IDEs from leading vendors, which are free of charge and deployed in close cooperation with ST. These include Eclipse-based IDEs such as Ac6 System Workbench for STM32 and the MDK-ARM environment.



**OPEN LICENSE MODELS:** STM32Cube software and sample applications are covered by a mix of fully open source BSD license and ST licenses with very permissive terms.

[www.st.com/stm32cube](http://www.st.com/stm32cube)

[www.st.com/x-cube](http://www.st.com/x-cube)