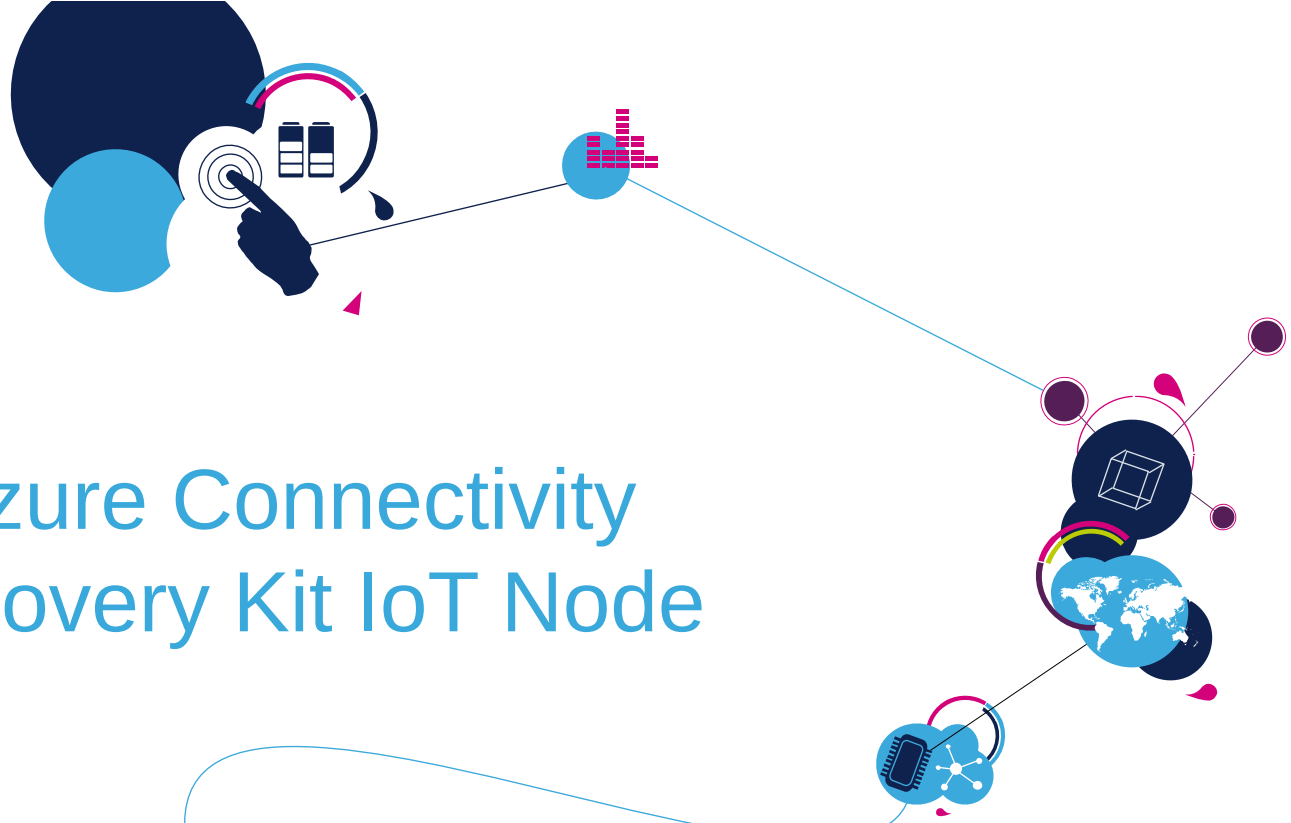


Simplifying Microsoft Azure Connectivity with the STM32L4 Discovery Kit IoT Node

v1.5

life.augmented

Agenda

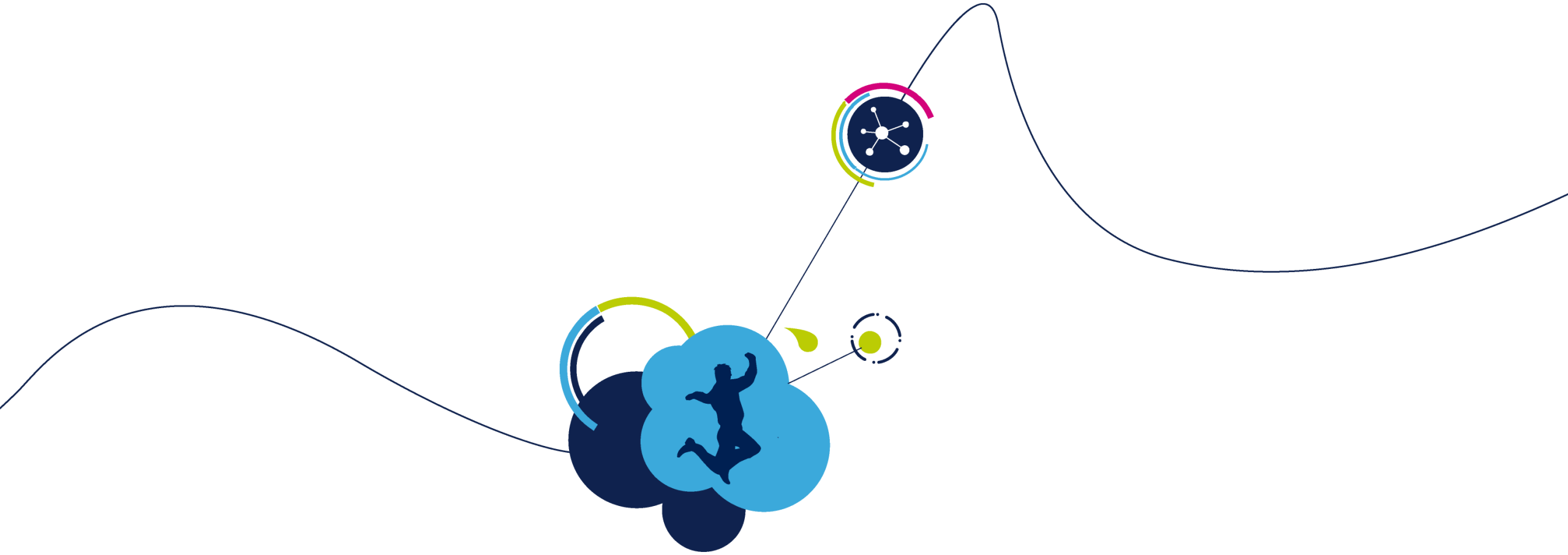
Presentation

- Training Material Check/Installation Help
- Overview of the STM32 Portfolio
- Keil MDK-ARM Installation
- Overview of the STM32L475
- Overview of the STM32L475 Discovery Kit IoT Node
- STM32Cube™ Introduction
- Keil MDK-ARM License Installation
- Lab 1: Getting Started with STM32CubeMX - Blinking LED
- Microsoft Azure IoT Overview
- Lab 2: Getting Started with Microsoft Azure IoT in 5 minutes!
- FP-CLD-AZURE1 Function Pack Overview
- Lab 3: Connect STM32 IoT Node to Azure IoT Hub
- LUNCH

Agenda

Presentation

- Lab 4: Monitor IoT Node Sensor Data sent to Azure Cloud
- Lab 5: Manage IoT Node from the Azure Portal
- Lab 6: Update IoT Node Firmware Over The Air (FOTA)
- Bluetooth® Low Energy Overview
- Demo: Simple BLE phone pairing



Training Material Check / Installation Help

life.augmented

Seminar Day Check List

- You would have received an email about a week ago with the Dropbox download link and instructions to install the necessary software.

- Please use the printed checklist handed to you today to make sure that you have completed the necessary steps.

Training Material Installation

- In case you have missed something, we have a few USB Flash drives with the Seminar Installer that we will pass around. This will install Tera Term, Device Explorer, NotePad++, and extract the seminar file to

C:\STM32L4AzureSeminar

- Please insert the USB Drive to your machine. Copy all the files to your desktop and execute the installer (Run as Admin)

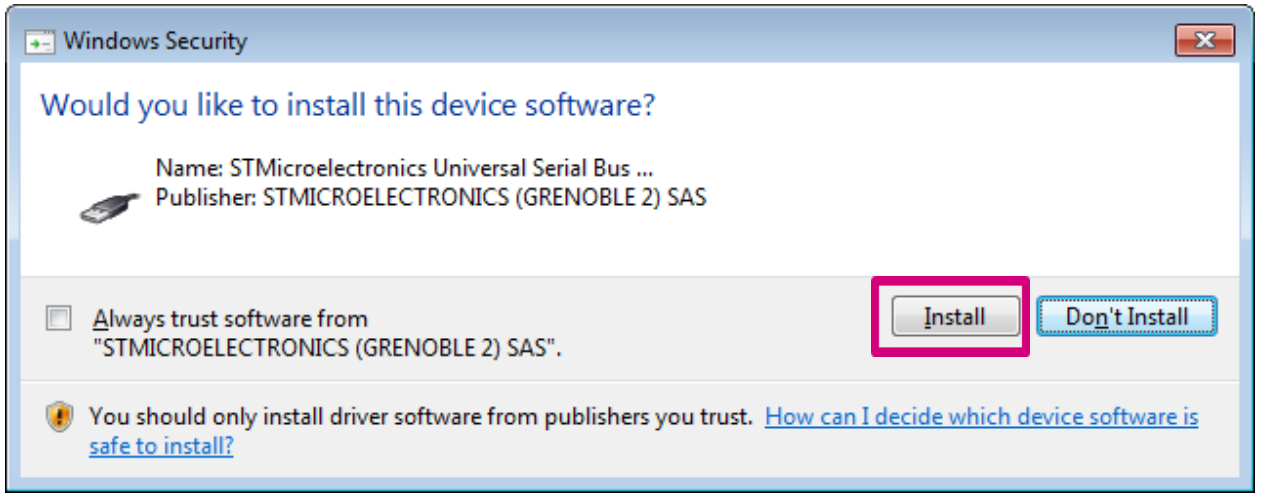
STM32L4AzureSeminar_Installer.exe

- The USB flash drive also contains this presentation in pdf format

STM32L4_DK_IoT_Azure_Seminar.pdf

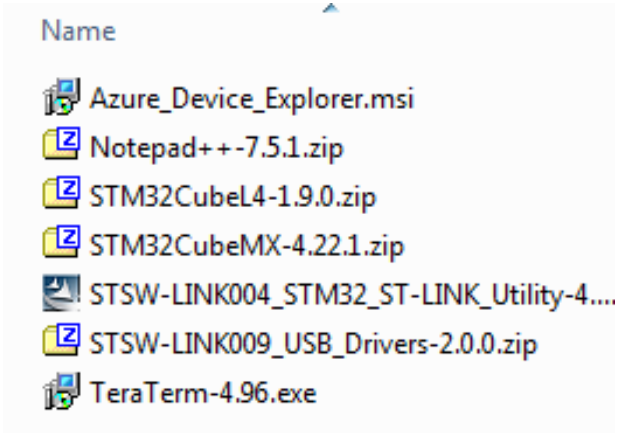
Training Material Installation (Cont'd)

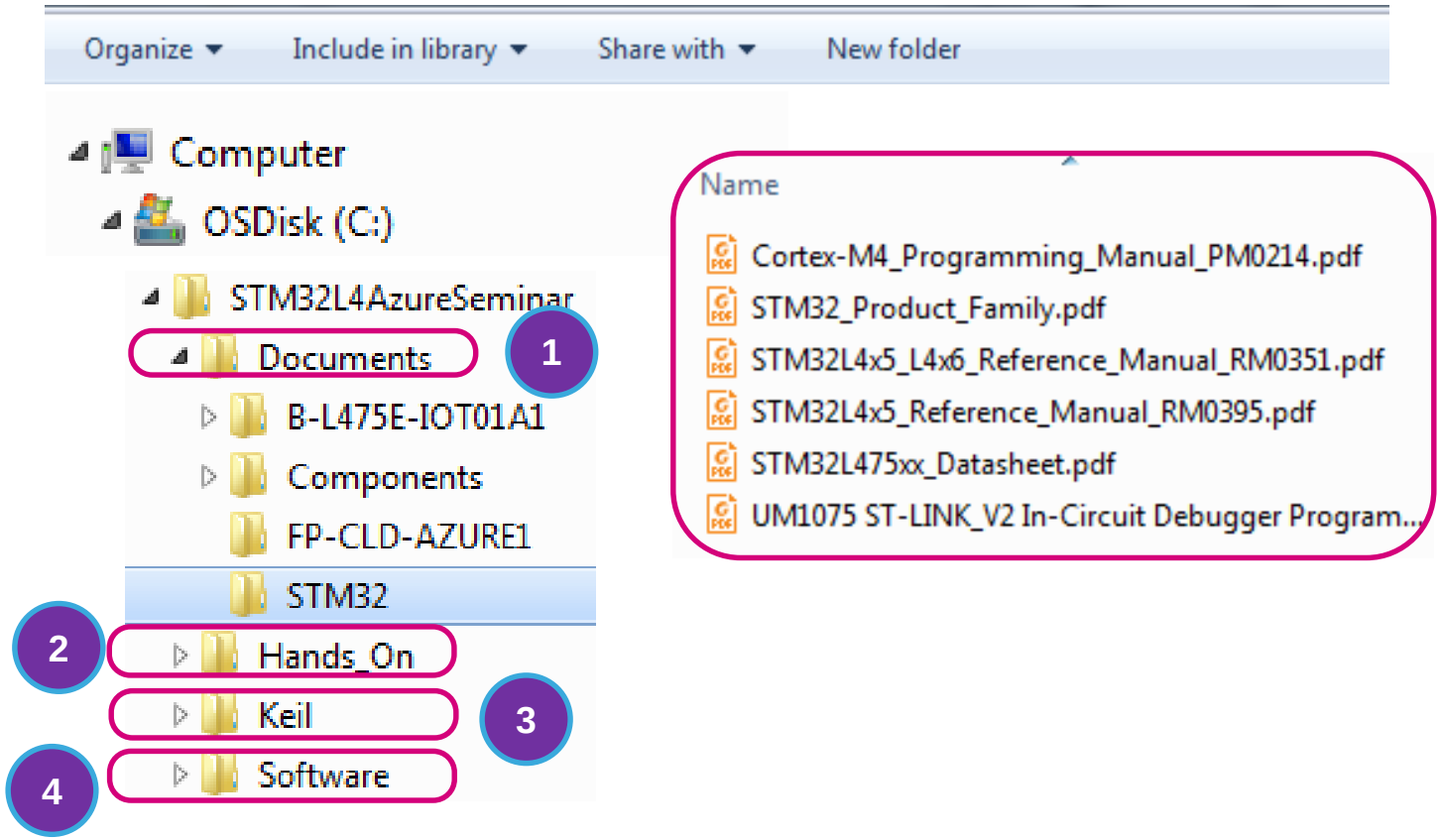
- If you see prompts such as the one below, please click "Install" to allow installing the ST-Link Device Drivers (there would be typically two such prompts).



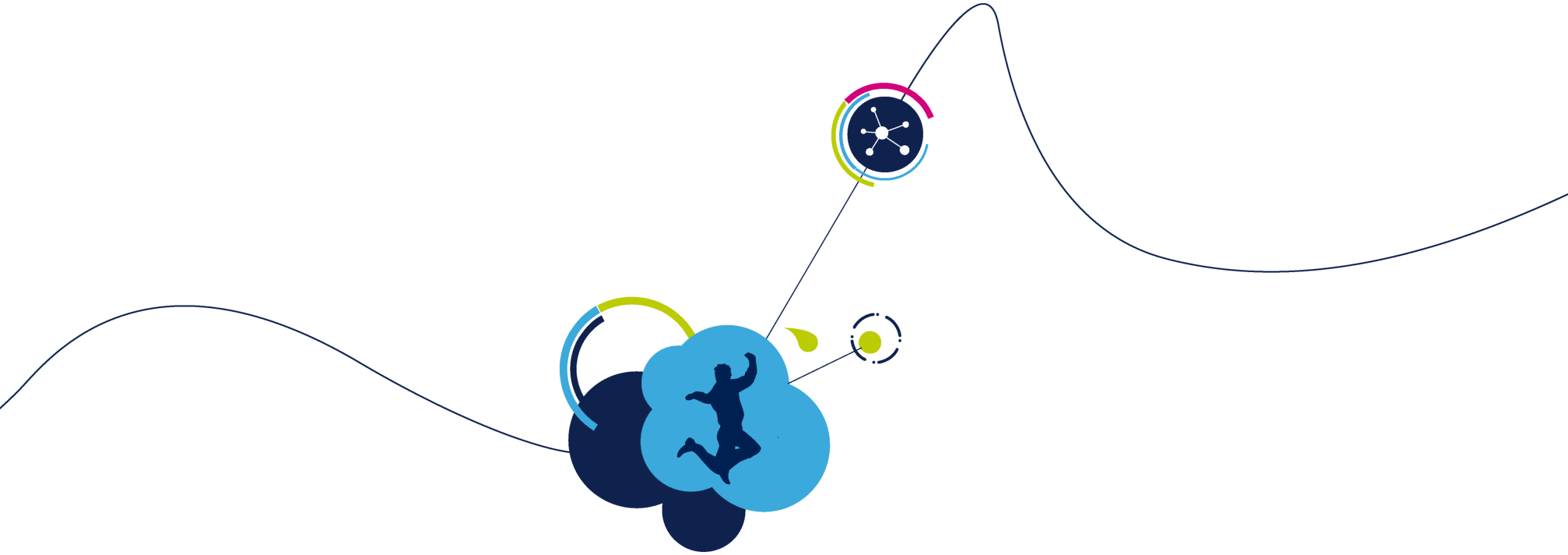
Workshop Directory Content

1. Documents
2. Hands on
3. Keil
4. Software





life.augmented

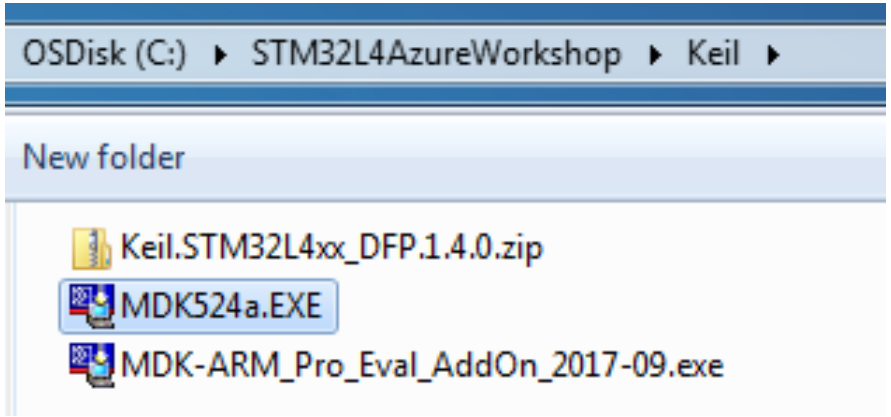


Keil MDK-ARM Tool Installation

life.augmented

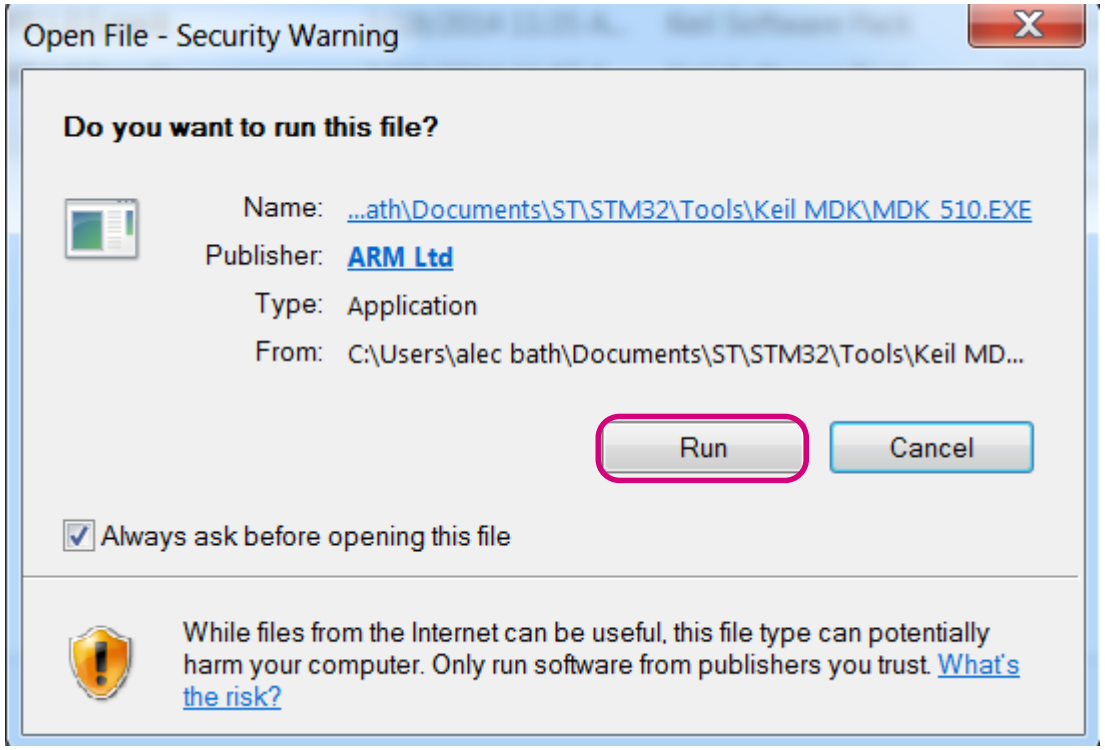
Step 1. Keil Installation

- Browse to **C:\STM32L4AzureSeminar\Keil** and Double-click on the file **MDK524a.exe** to begin the Keil toolchain installation.
  - Click through the default options and accept the 'license agreement' when prompted.
  - If you have an existing installation of Keil you should install in a new directory



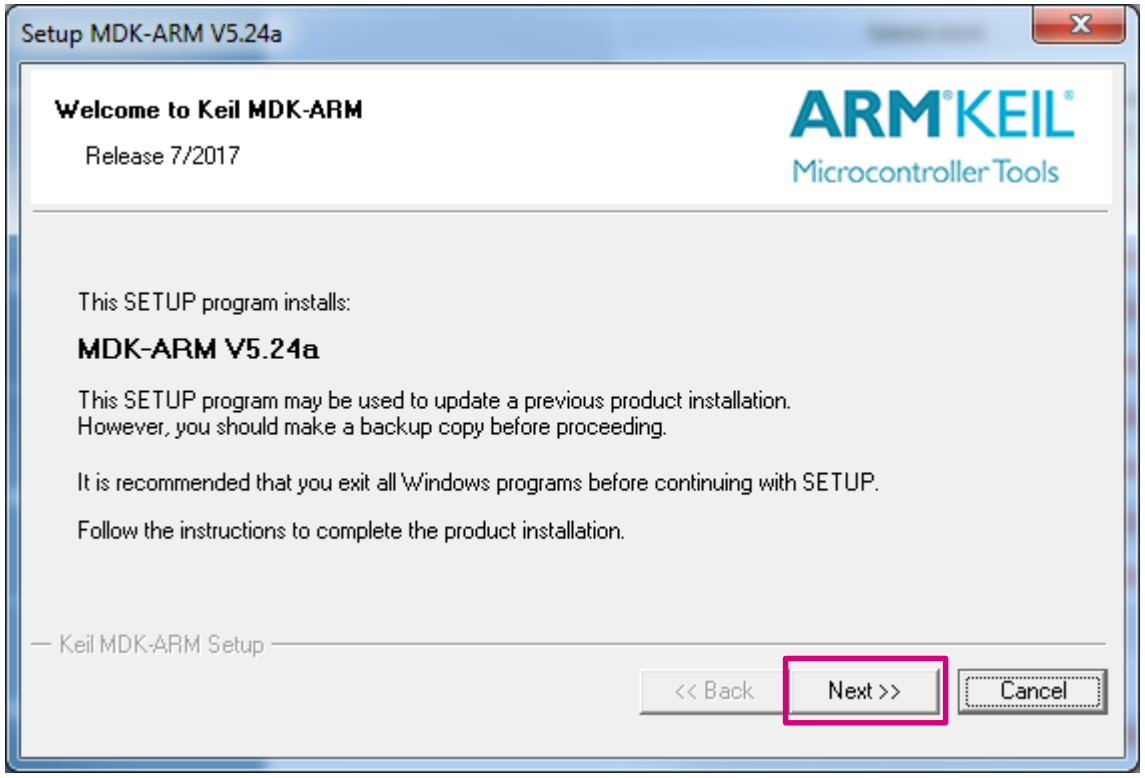
Step 1. Keil Installation

- Click on Run to begin the installation



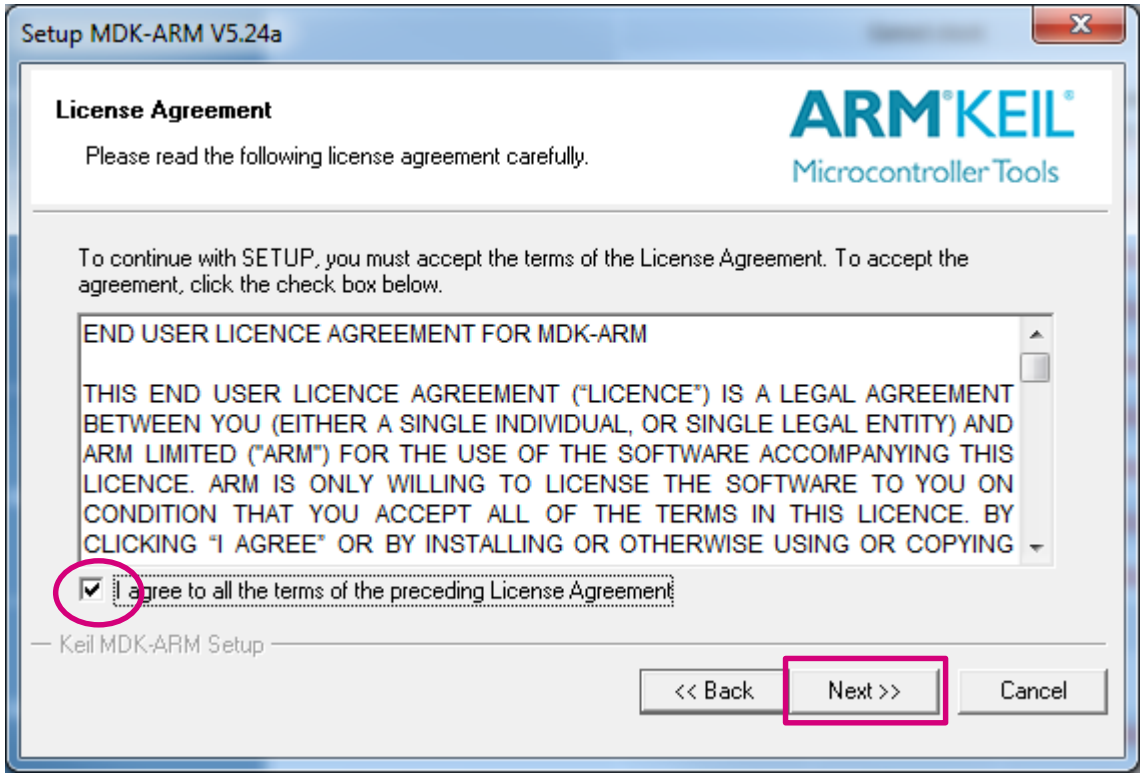
Step 1. Keil Installation

- Click on Next



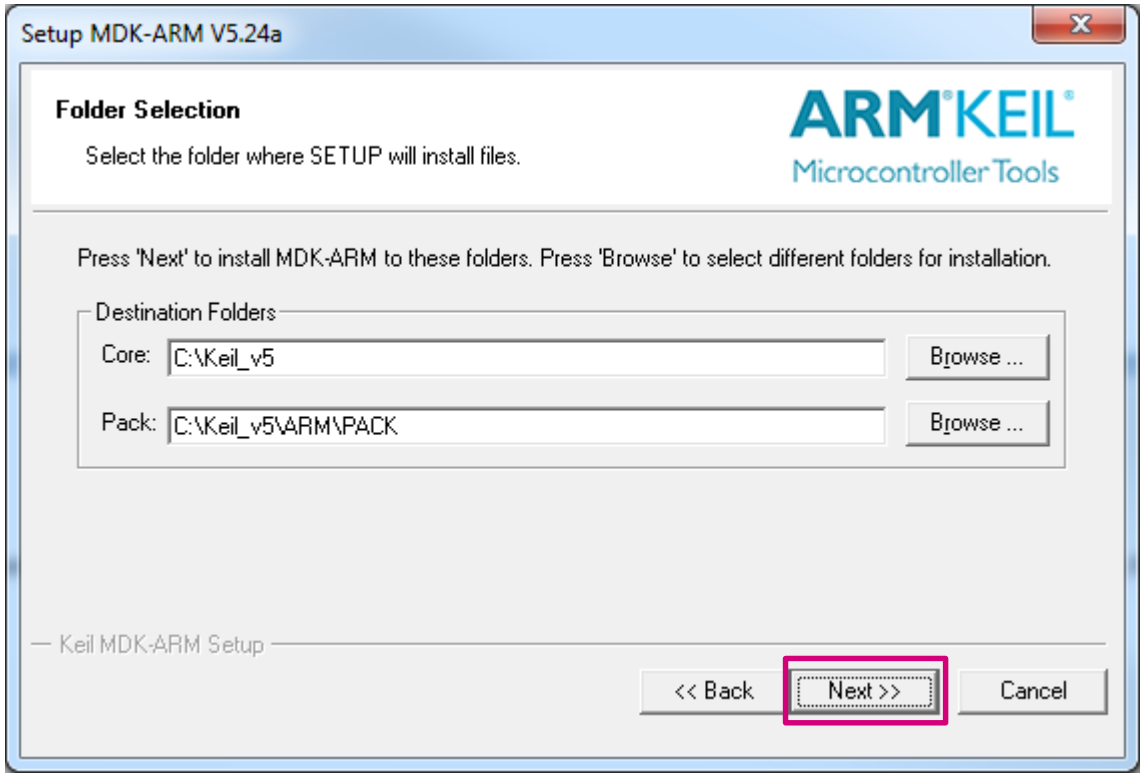
Step 1. Keil Installation

- Accept the license agreement and click on Next



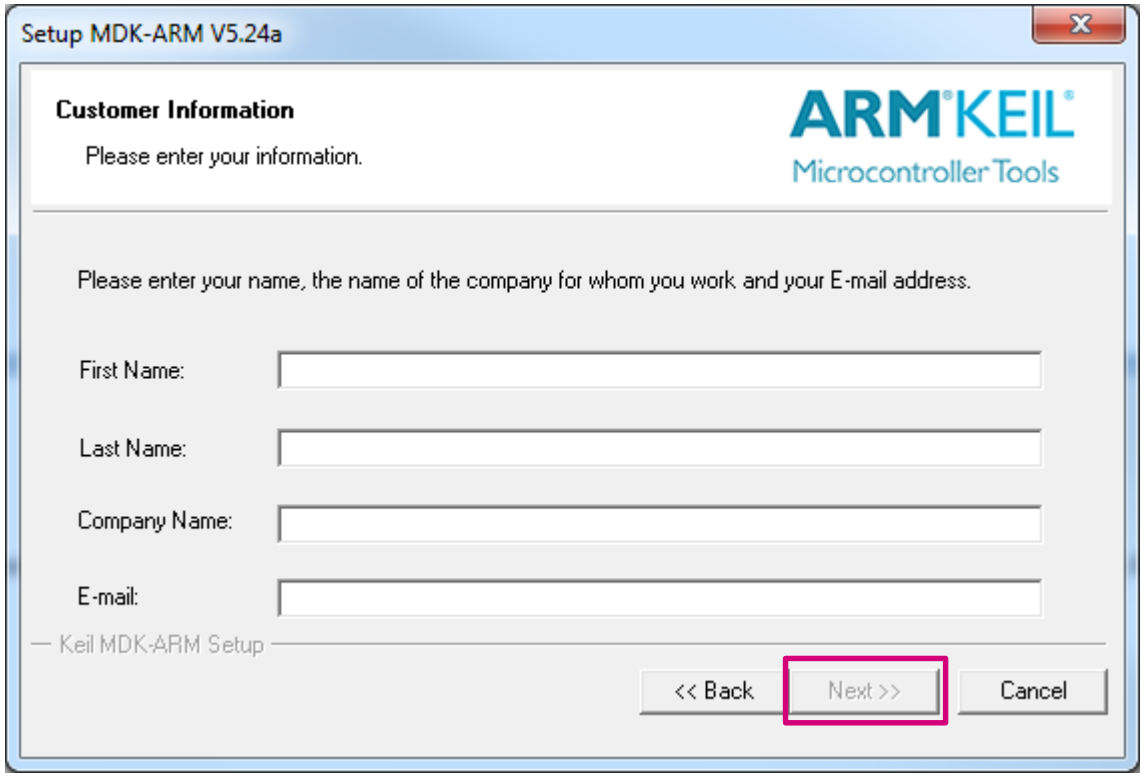
Step 1. Keil Installation

- Please use the default directory path: C:\Keil_v5 and click on Next



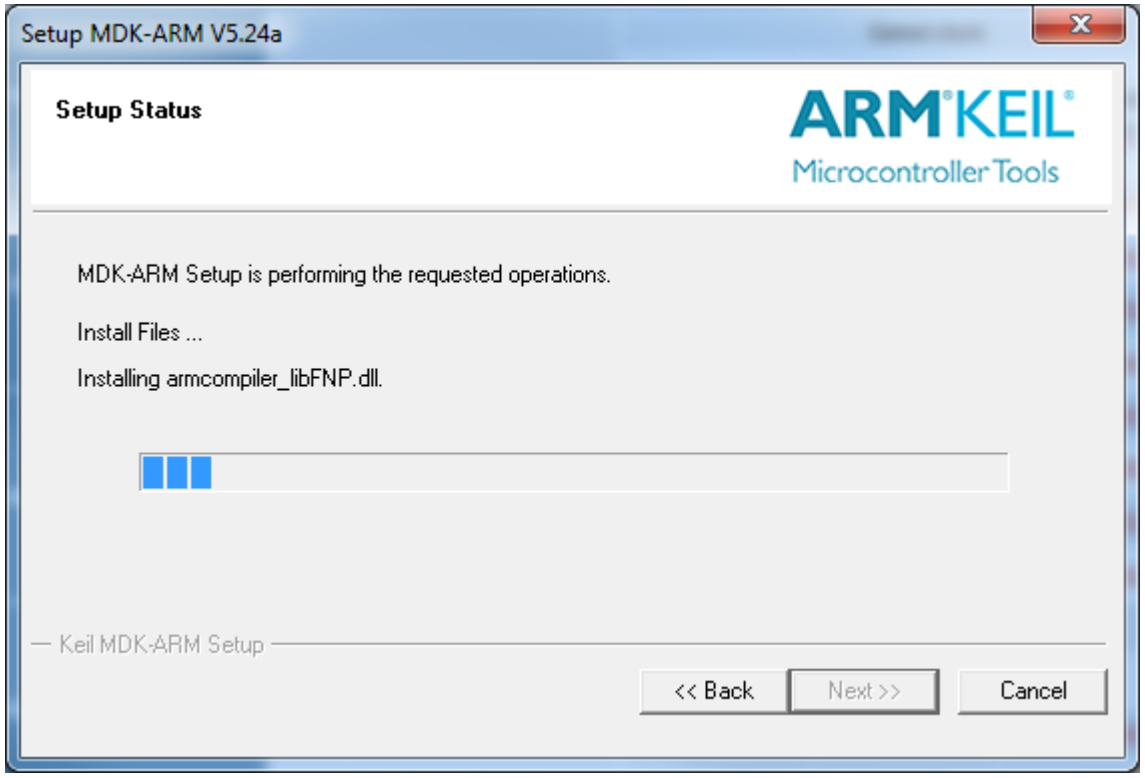
Step 1. Keil Installation

- Enter your contact information and click on Next



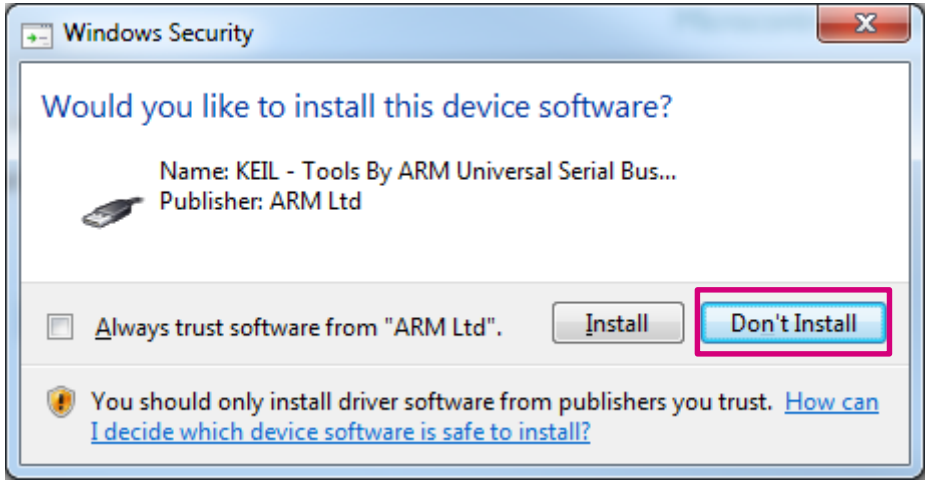
Step 1. Keil Installation

- Installation begins…



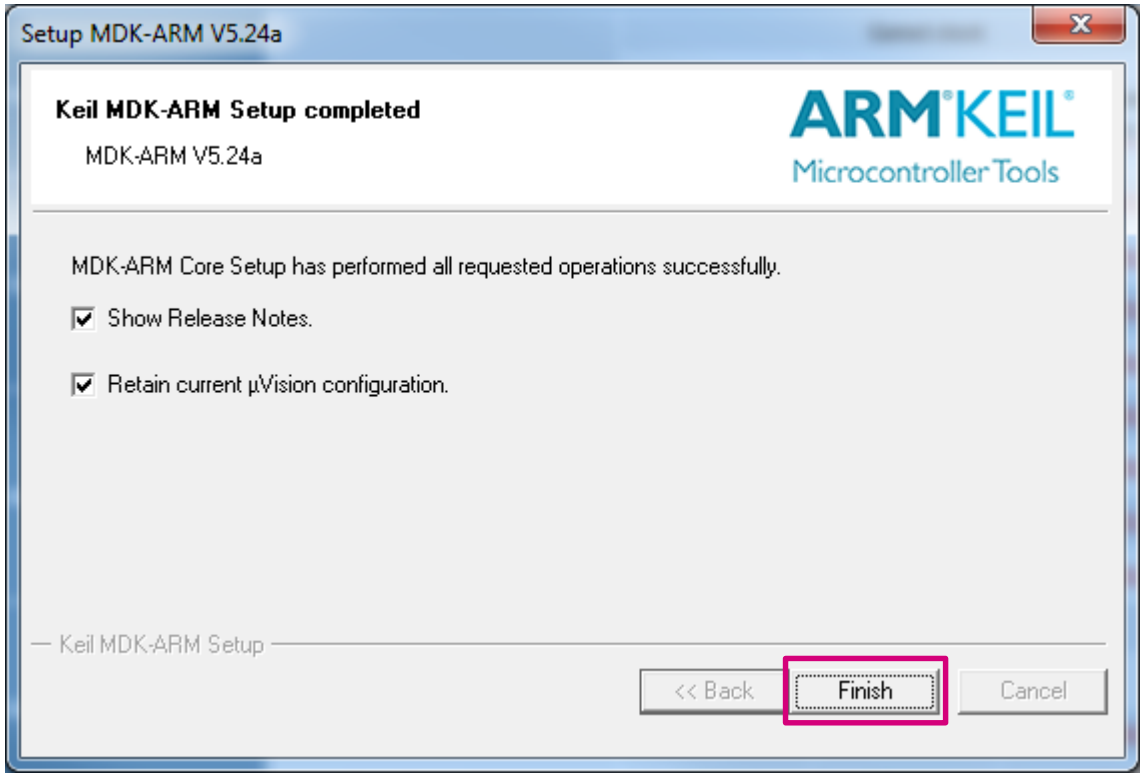
Step 1. Keil Installation

- Before installing ULINK device drivers, Windows will block it and asks for confirmation. We will not use it so just click Don’t Install

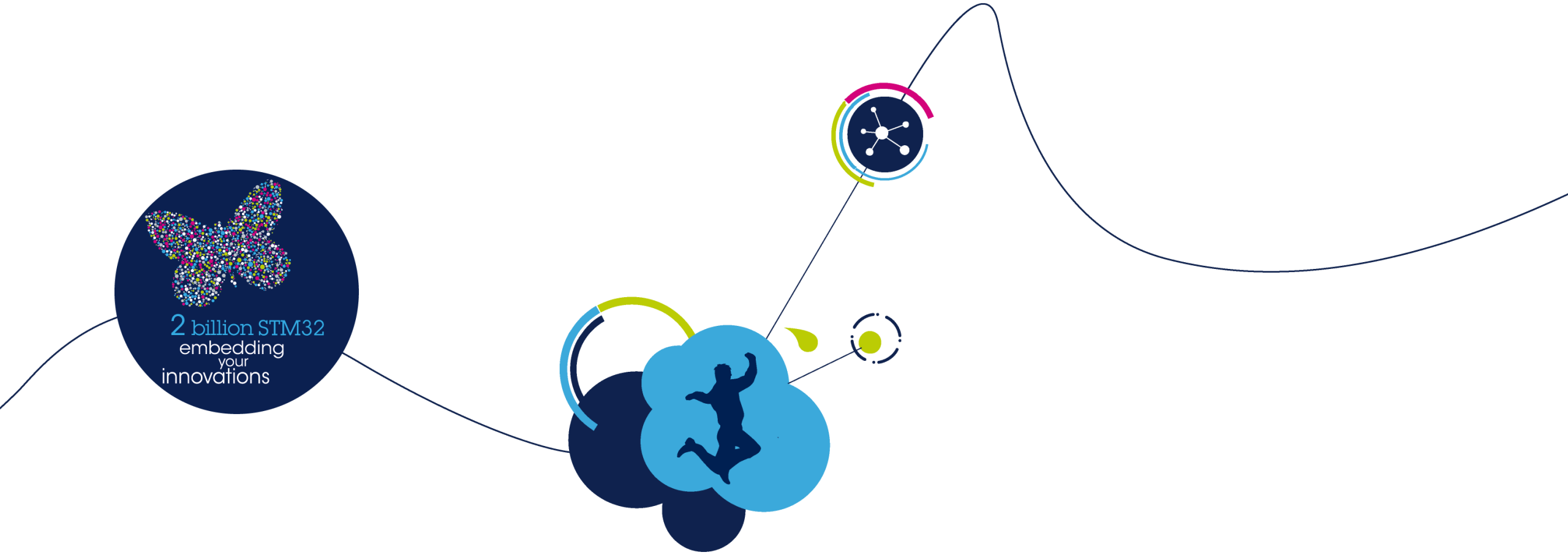


Step 1. Keil Installation

- Setup complete…. Click Finish
- We will manually install “Software Packs” a bit later



If you get any errors for the “Pack Installer”, don’t worry.
Just click OK, and we will get to that in a bit.

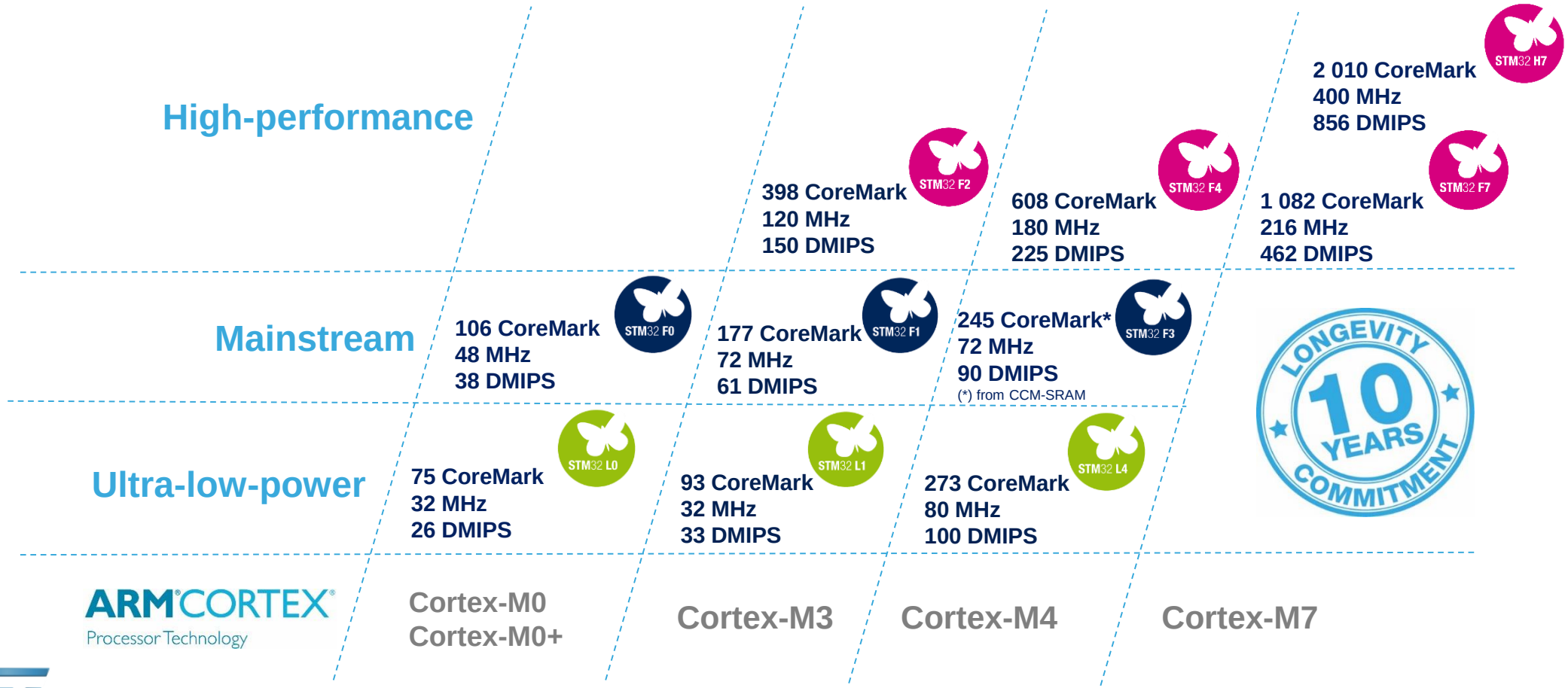


Overview of the STM32 Portfolio

life.augmented

Today - STM32 Portfolio

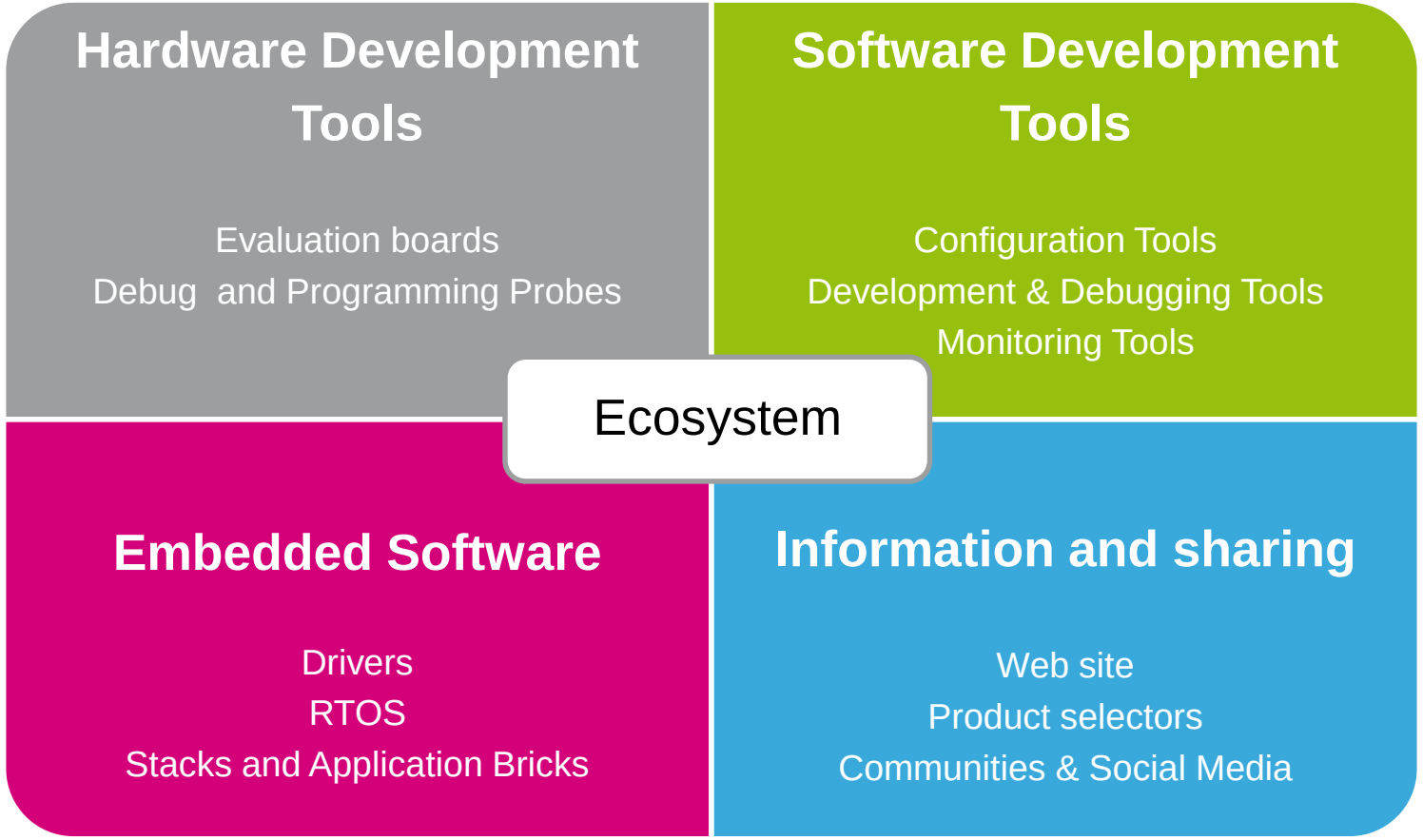
10 product series / More than 40 product lines



life.augmented

What is the MCU Ecosystem?

All collaterals required to develop with an MCU

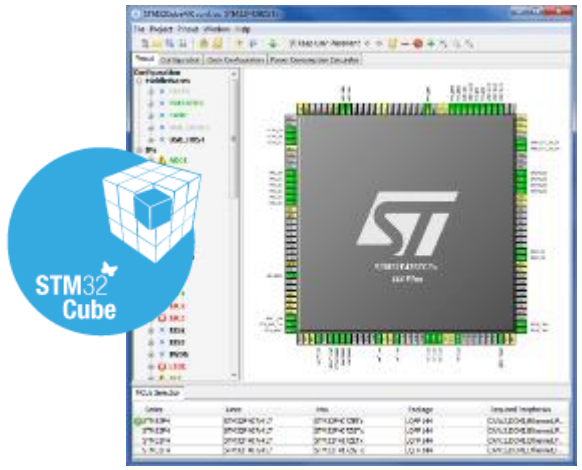


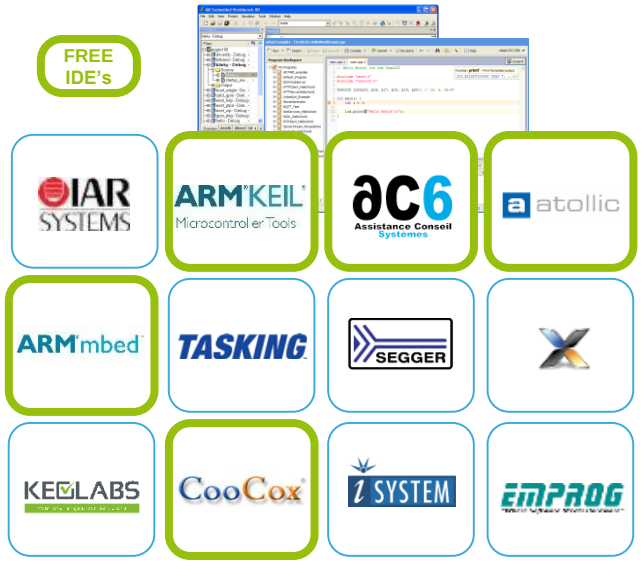
Hardware Development Tools

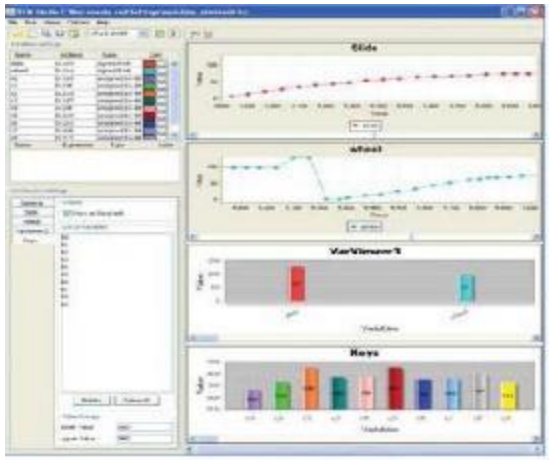
	STM32 Nucleo	Discovery kits	Evaluation boards	3rd parties
Typical use case	Flexible prototyping, Community	Prototyping, Creative demos	Full feature evaluation	From full evaluation to open hardware
Extension possibilities	+ + +	+ +	+	
Connectivity	Arduino™ ST Morpho	ST	ST	

STM32 Ecosystem SW Development Tools

A complete flow, from configuration up to monitoring







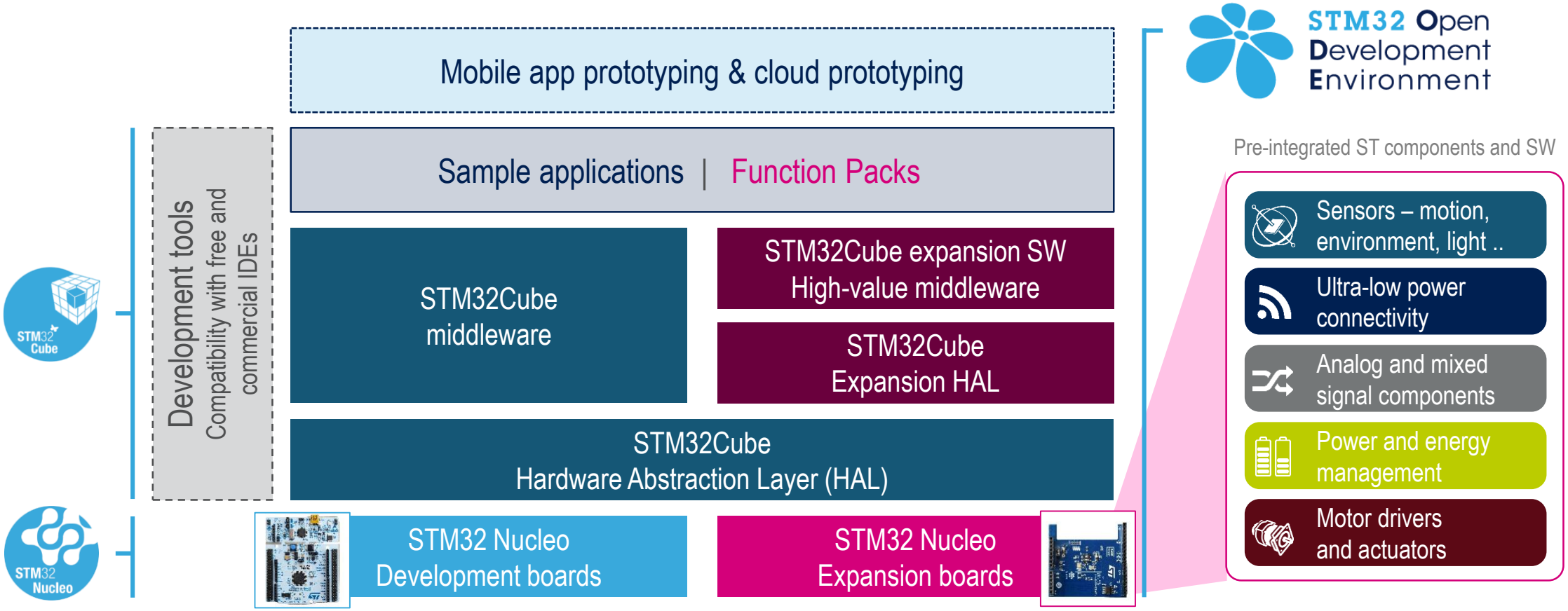
STM32CubeMX
Configure & Generate Code

Partners IDEs
Compile and Debug

STMStudio
Monitor

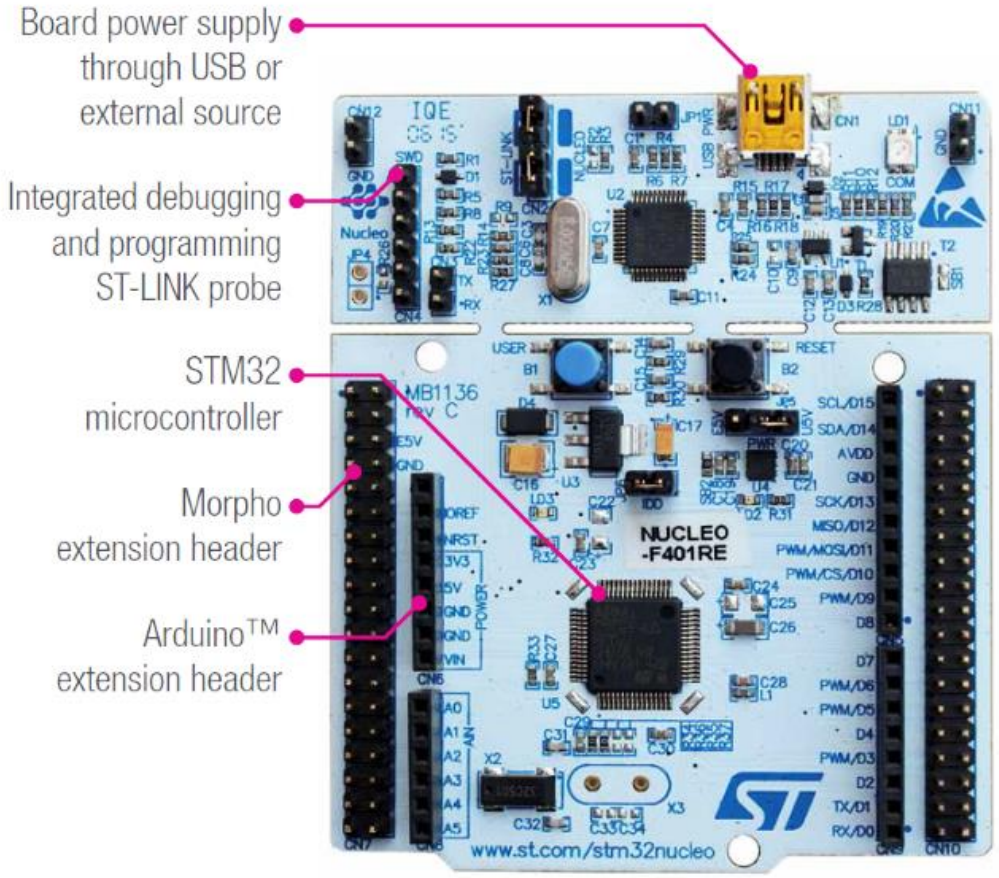
life.augmented

Development Software Architecture



STM32 Nucleo Development Boards

A comprehensive range of affordable development boards for all STM32 microcontroller series, with unlimited unified expansion capability, and with integrated debugger/programmer



STM32 Nucleo

STM32 F0 STM32 F1 STM32 F2 STM32 F3 STM32 F4 STM32 F7

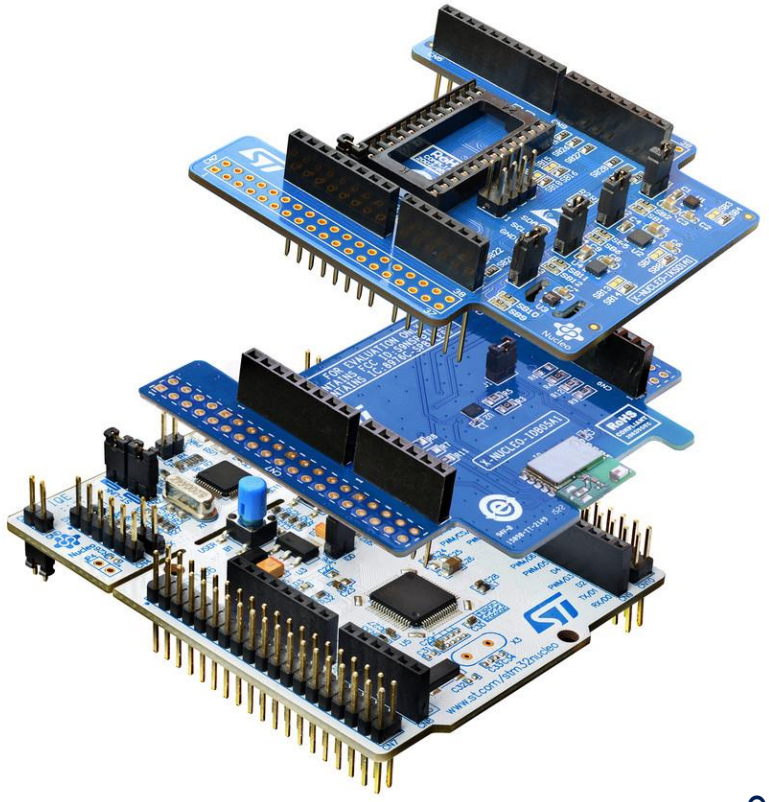
STM32 L0 STM32 L1 STM32 L4

Complete product range
from ultra-low power to high-performance

www.st.com/stm32nucleo

life.augmented

STM32 Nucleo Expansion Boards



http://www.st.com/x-nucleo

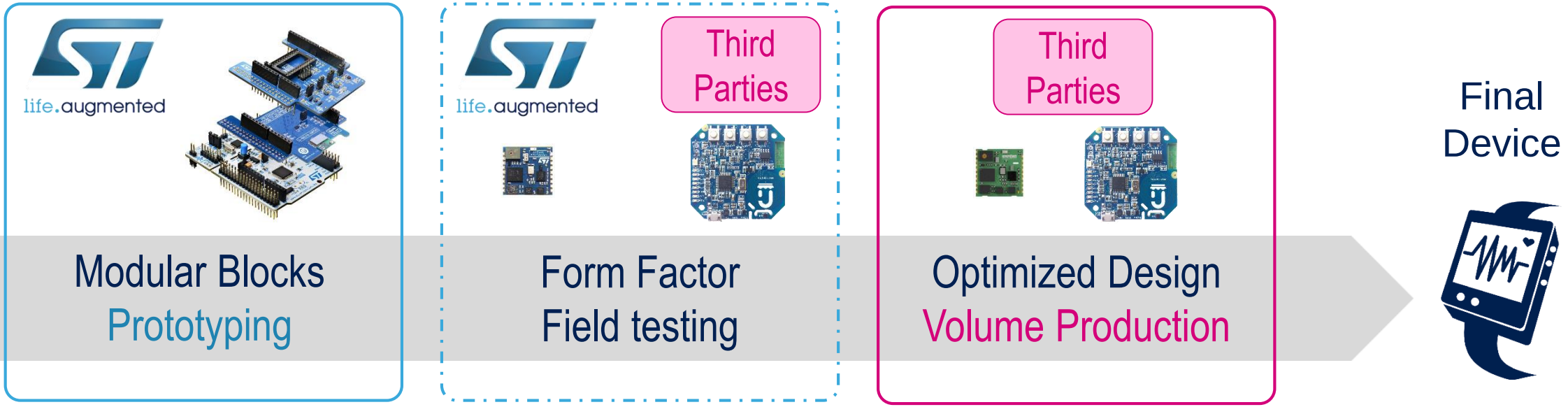
33 expansion boards covering all the key functions

Sense	Connect	Power Drive	Move Actuate	Translate
Motion & environmental sensors Proximity sensor Microphone	BLE Wi-Fi Sub-GHz NFC	Power management LED Boost	Motor driver Actuator	Audio Op-amp

life.augmented

From Prototype to End Device

Partnership with third parties - Form factor devices and customization



Easy Porting of developed software to final Product

HW/SW optimization and support for production

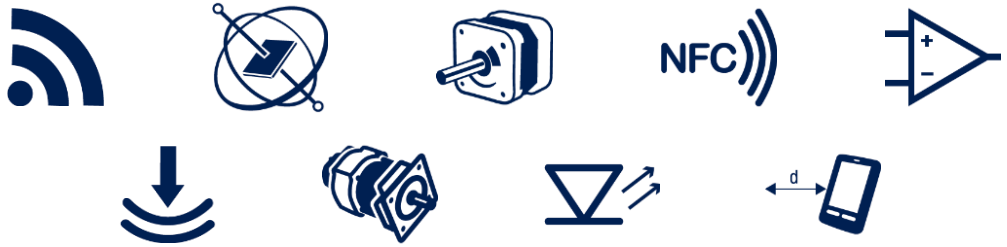
life.augmented

Simple Vs Advanced Use Cases

Expansion SW (X-Cube) vs Function Pack

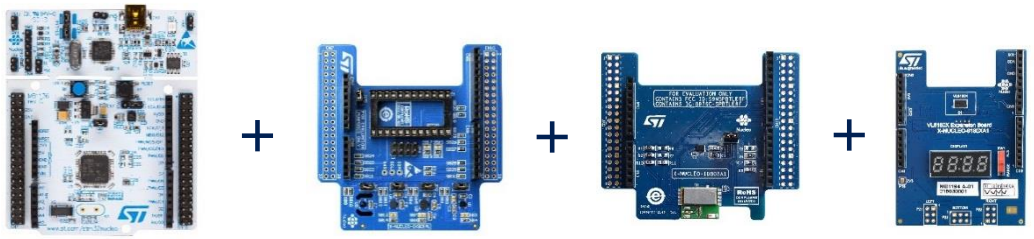
- Prototype with a single expansion board

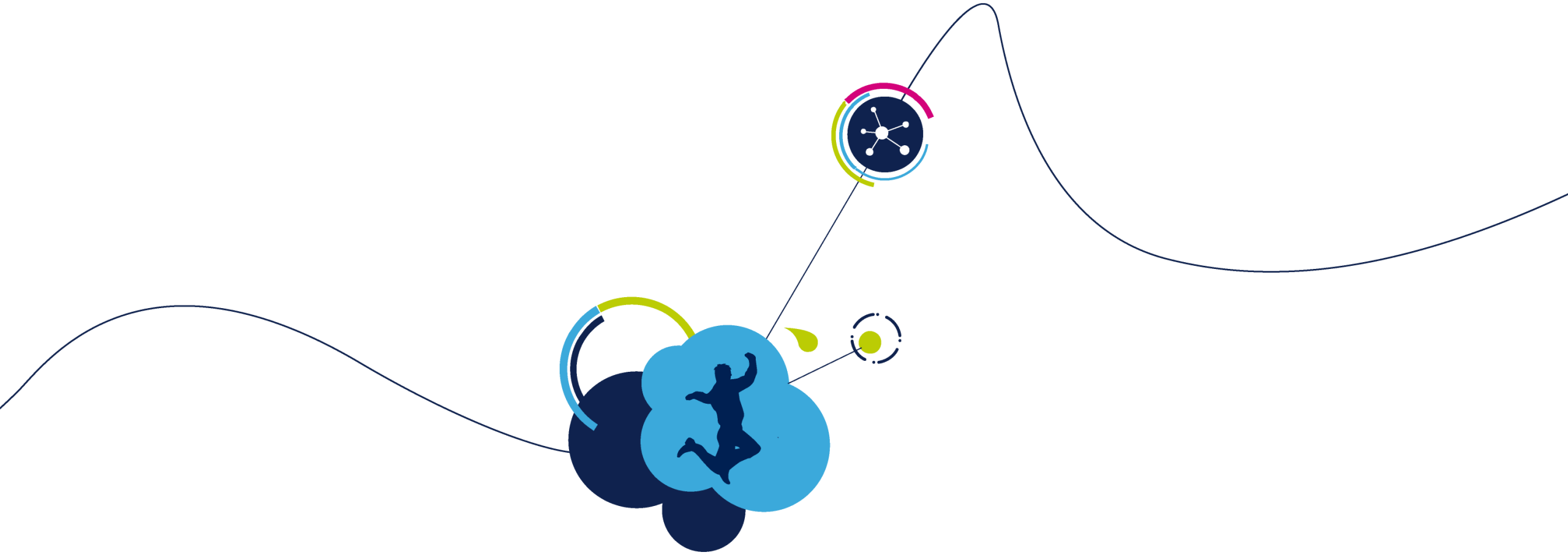
Sample applications



- Create advanced use cases based on multiple expansion boards

Pre-integrated application example





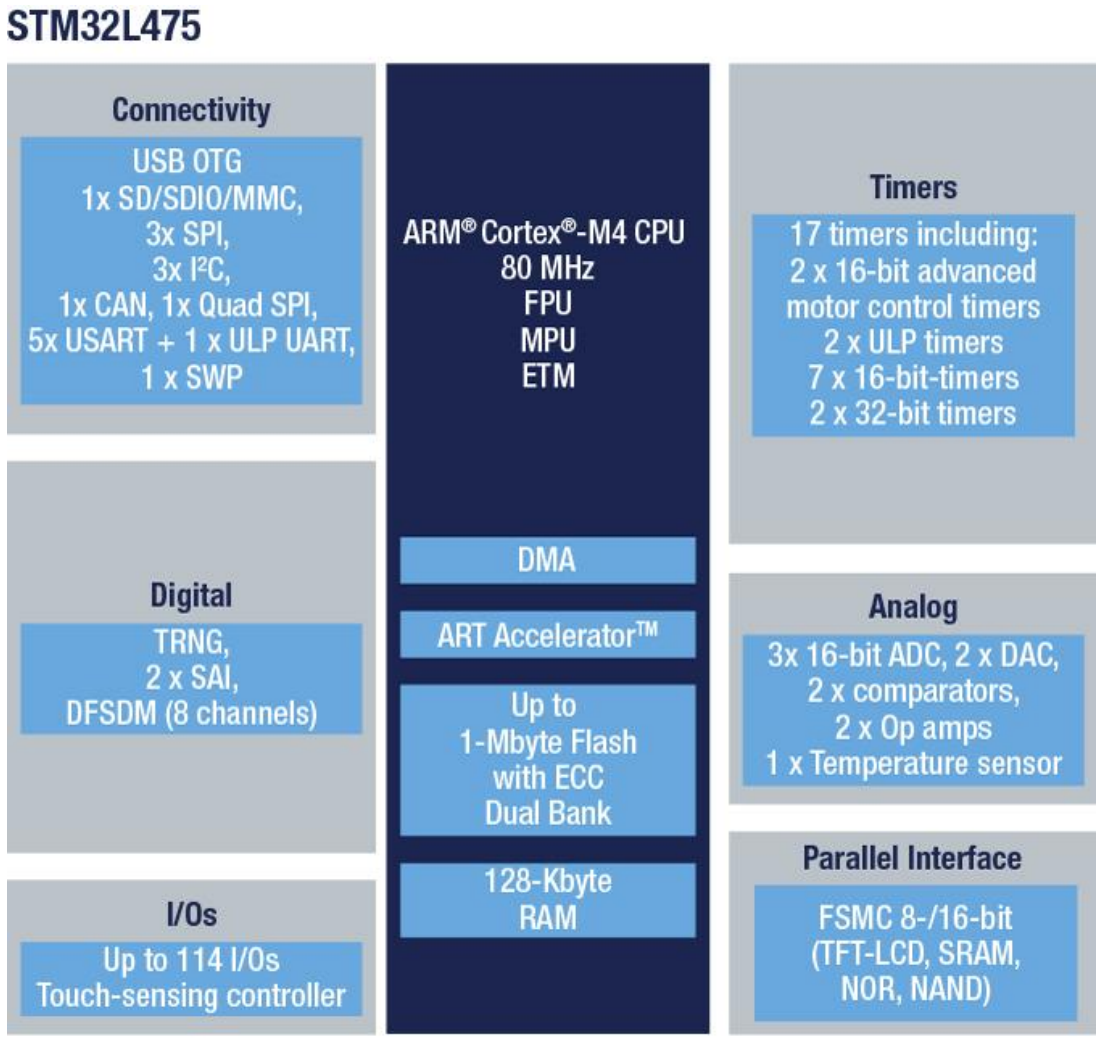
Overview of the STM32L475

life.augmented

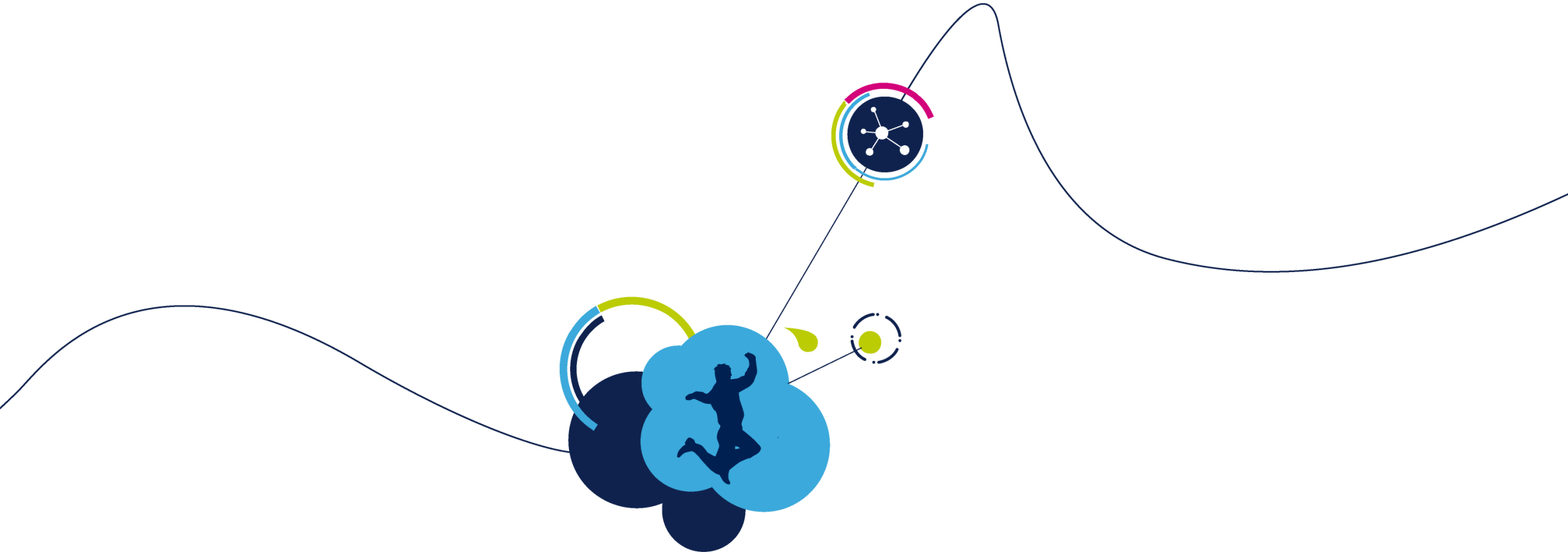
Overview of the STM32L475

Key features

- Cortex M4 @ 80 MHz with DSP, FPU and ART
- 1.71V – 3.6V supply 80 MHz Full functional
- 1MB Flash dual bank / 128KB RAM
- USB OTG FS –LPM Battery Charging Detection
- 3 x Ultra-low-power 12-bit ADC @ 5 MSPS
- Touch-Sensing 24 channels
- Ultra-low power
  - VBAT
  - Down to 160µA/MHz dynamic
- I²C FM+
- SPI: variable data length
- USART
- LP UART & 16-bit Timer
- FSMC, Quad SPI
- CAN, SWPMI, SDMMC, 2x SAI
- Digital filter for Sigma delta modulator
- 17 x timers
- Analog: Op-Amps, comparators, DAC, VREF, Temp
- RNG



life.augmented

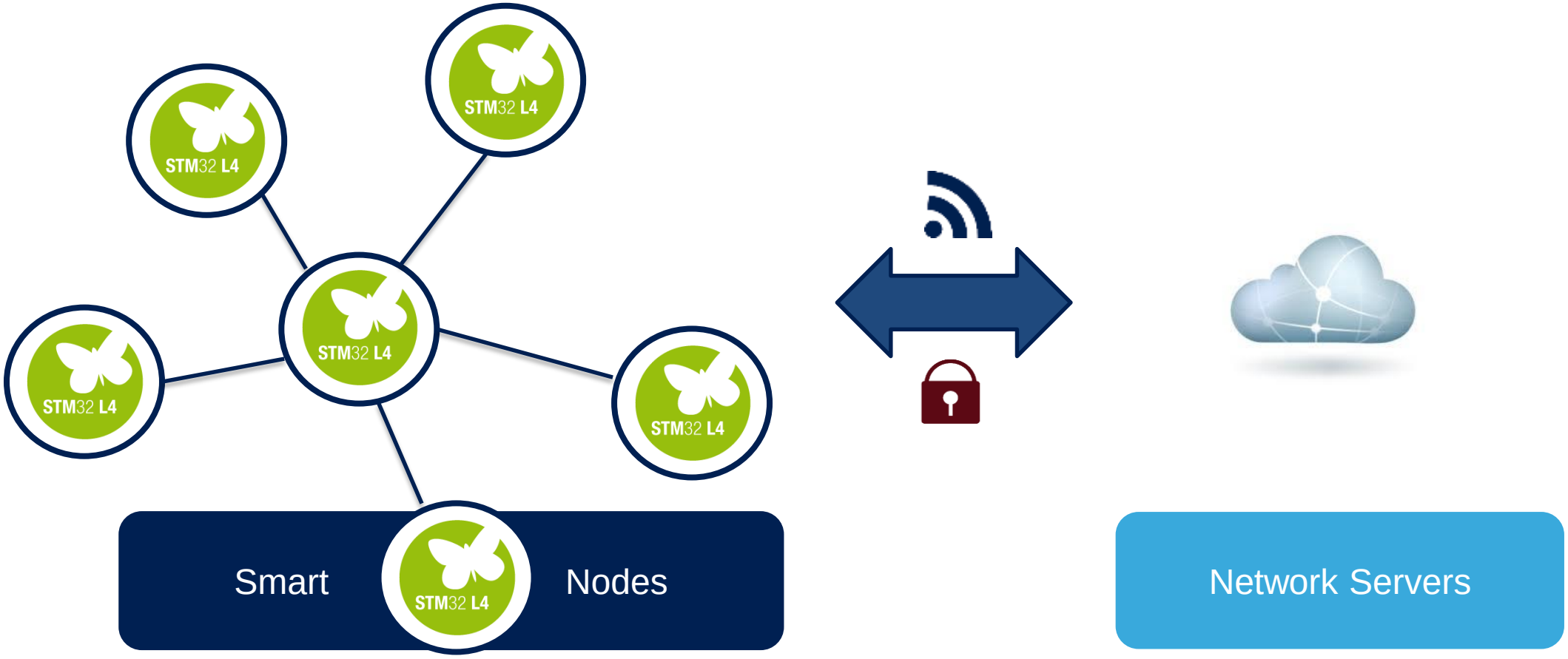


Overview of the STM32L4 Discovery Kit IoT Node

life.augmented

STM32L4 Discovery Kit IoT Node

Get connected seamlessly!



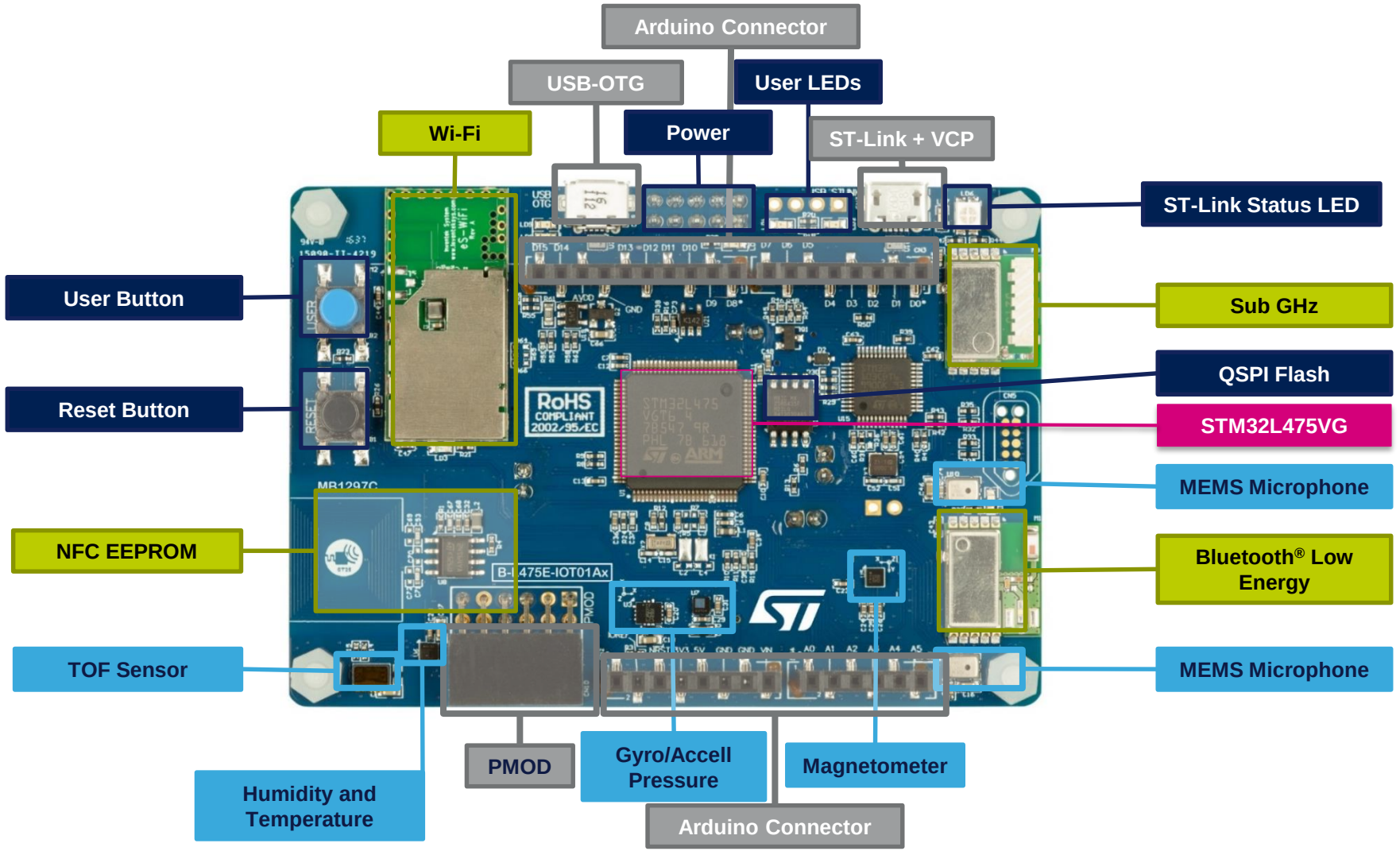
STM32L4 Discovery Kit IoT Node

Open the door to remote services

- Direct connection to cloud servers
- Low-power long-range communication
- Environmental awareness: humidity, pressure, temp
- Detection hub: motion, proximity, audio

STM32L4 Discovery Kit IoT Node

Multi-link communication, multiway sensing

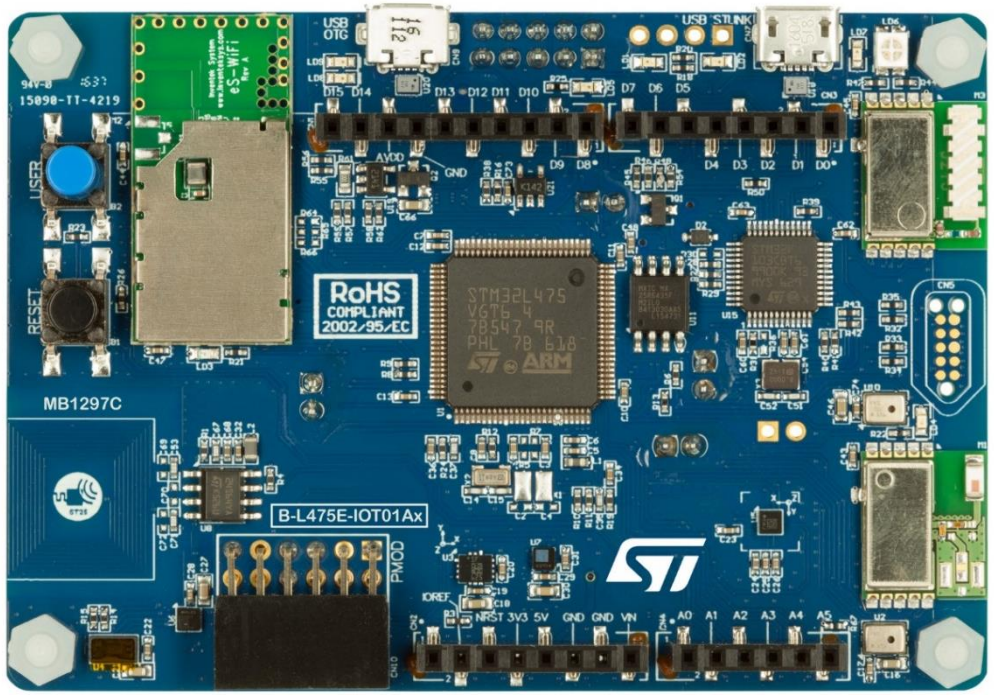


life.augmented

Comprehensive Software Libraries

Instant showcase

- SW Libraries for STM32L4 MCU & sensors
- Connectivity SW protocol stacks
- Cloud service connectors (AWS, Azure)
- Demo examples
  (X-CUBE-AWS, X-CUBE-AZURE, FP-CLD-AZURE1)



Wireless Connectivity – Wi-Fi

- Inventek ISM43362 Wi-Fi Module
  - 802.11 b/g/n compliant Broadcom MAC/Baseband/Radio module
  - Fully contained TCP/IP stack to minimize host CPU requirements
  - FCC and CE certified
  - Secure Wi-Fi authentication supporting WEP-128, WPA-PSK (TKIP), WPA2-PSK

Wireless Connectivity - Bluetooth®

- ST SPBTLE-RF Bluetooth® Low Energy Module
  - Based on our ST BlueNRG-MS Wireless Network Processor
  - Bluetooth® Low Energy 4.1 compliant
  - FCC and BQ certified module with integrated balun & antenna

Wireless Connectivity - SubGHz

- ST SPSGRF-915 Sub-GHz Module (915 MHz - US)
  - FCC and IC certified module with integrated balun & antenna
  - Supports 2-FSK, GFSK, MSK, GMSK, OOK and ASK modulation schemes
  - Long range (100s of meters) with an air data rate from 1 to 500 kbps

Wireless Connectivity - NFC

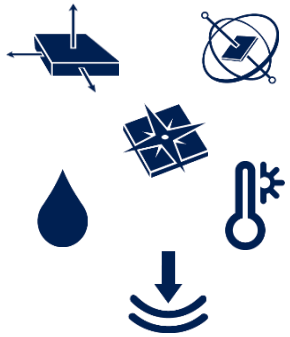
- ST M24SR64-Y Dynamic NFC/RFID Tag
  - ISO14443-A NFC Forum Type 4 RF interface
  - 106 Kbps Data Rate
  - I²C 1MHz interface
  - Up to 64kbit EEPROM memory
  - 128-bit password for data protection
  - 200 years data retention & 1Mcycles erase/write
  - Configurable General Purpose Output signal for MCU wake-up
  - RF disable feature

Wired Connectivity Features

- ST-Link V2
  - STM32 Programming and Debug Interface
- USB OTG FS
  - Full Speed USB On-The-Go Communication Interface
- PMOD
  - Peripheral Module Interface Supporting GSM, GPS, etc…
- Arduino Connectors
  - Arduino Compatible Connectors to interface with additional ST X-NUCLEO or 3rd Party Expansion Board (eg: LoRa)

ST Sensors

- Full Range of Motion & Environmental MEMS Sensors
  - LSM6DSL Accelerometer + Gyroscope Sensor
  - LIS3MDL Magnetometer Sensor
  - HTS221 Humidity + Temperature Sensor
  - LPS22HB Pressure Sensor



- Integrated High Accuracy Proximity/Range Sensor
  - VL53L0X Time-of-Flight Range Sensor

- Digital Microphones
  - MP34DT01 MEMS Digital Microphones
    - Voice & Audio Recognition Functions
    - Acoustic Beam Forming using the ST Open Software Acoustic Beam Forming Library

User Resource Features

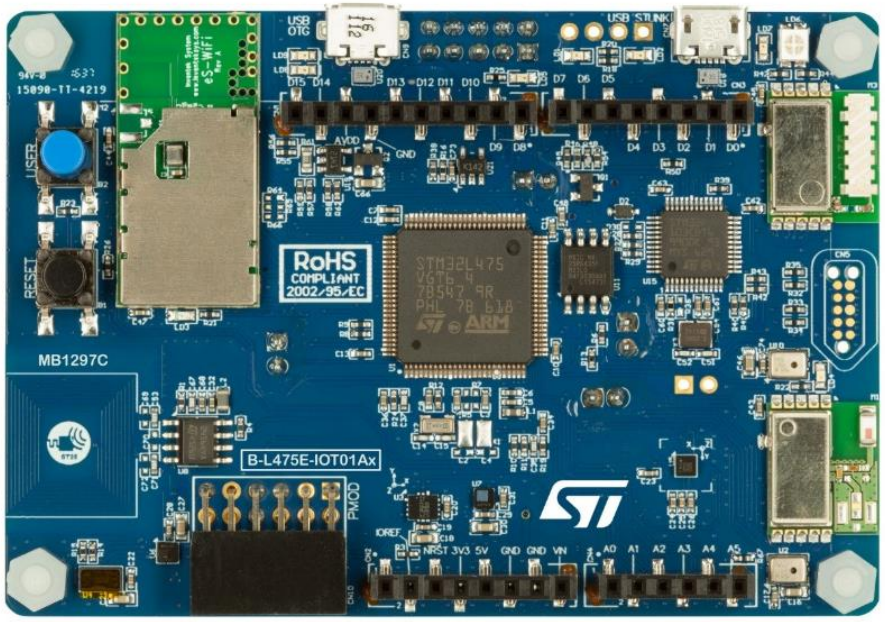
- Reset and User Buttons
  - Board Reset and Programmable Application Buttons
- User LEDs
  - Programmable Application LEDs
- QSPI Flash
  - 64Mbit for Data Storage and Program Execution
- Selectable Power Supply
  - ST-Link, USB-OTG, Arduino or External Power

Advantages of a Single Board

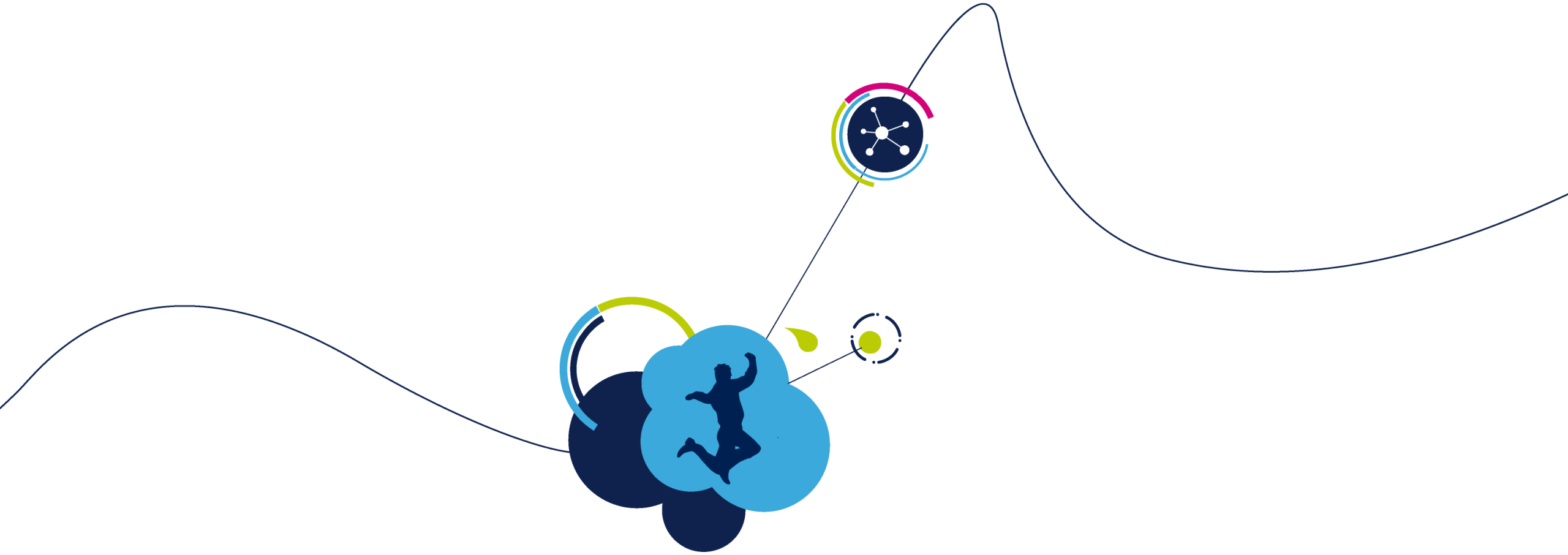
- Easily Debug Hardware Issues.
- Included Collateral is Tightly Coupled
  - BSP Included for all Board Components
  - Cloud Connectivity Reference Solutions Included
- Cost Effective Development Solution (~$60)
- No Need to Manage & Order Multiple Board SKUs.

Availability

Part number	Samples	Mass Market Availability	SubGHz frequency band	Regions with authorized use
B-L475E-IOT01A1	NOW	NOW	915 MHz	US
B-L475E-IOT01A2	NOW	NOW	868 MHz	Rest of the World



life.augmented

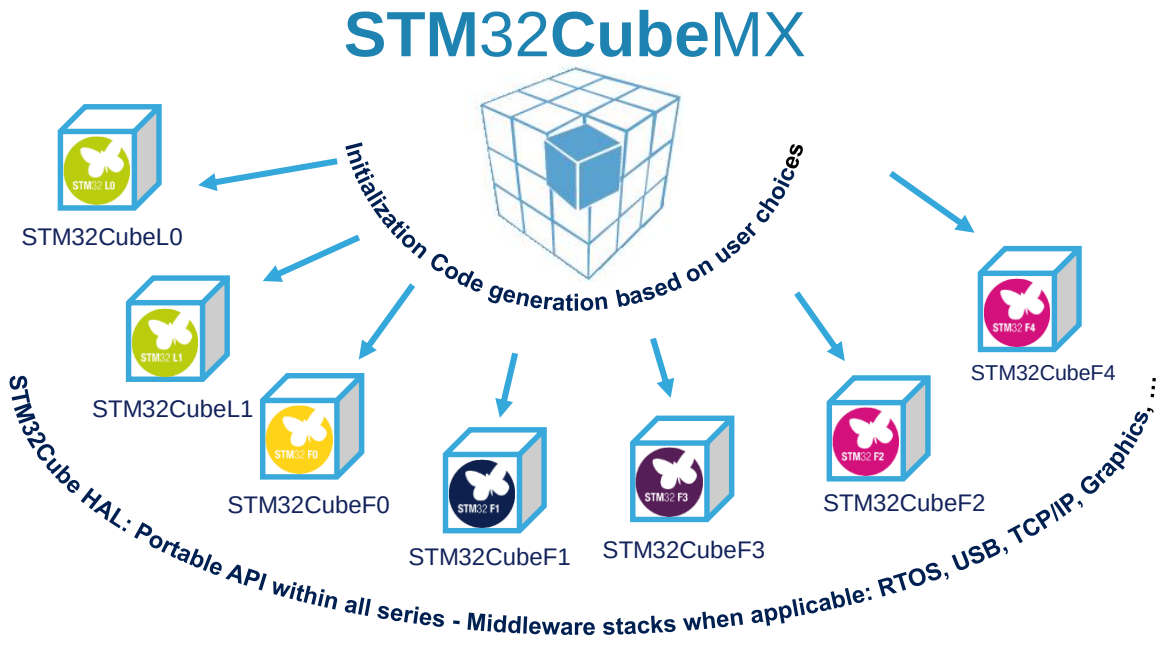


STM32Cube™ Introduction

life.augmented

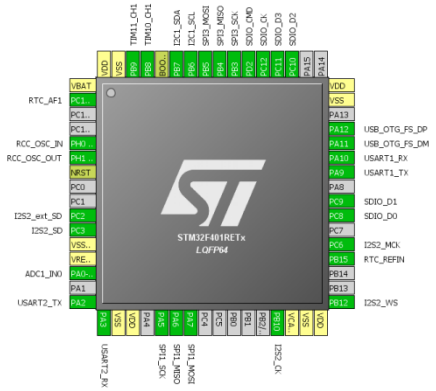
STM32Cube™ Introduction

- STM32Cube™ includes:
  - STM32CubeMX, a configuration tool for generating user driven initialization
  - Firmware collateral, delivered for every STM32 device series (i.e. STM32CubeF4) with:
    - An STM32 Abstraction Layer embedded software: STM32Cube HAL
    - A consistent set of Middleware: RTOS, USB, TCP/IP, Graphics, …



STM32CubeMX : Overview

STM32CubeMX



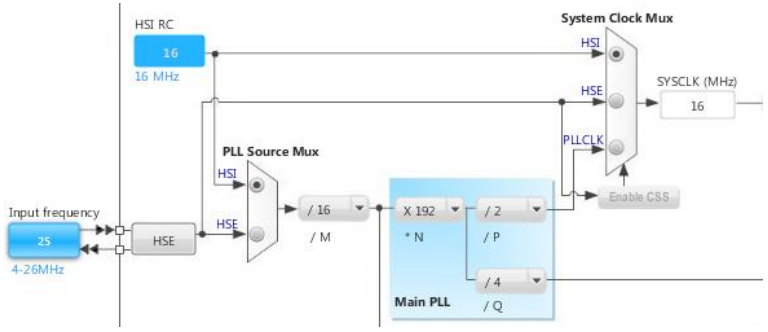
Pinout Wizard

Basic Parameters	
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples

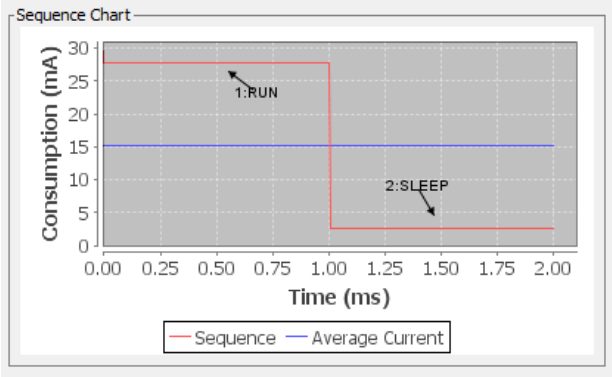
Baud Rate
BaudRate must be between **110 Bits/s** and **10.5 MBits/s**.

Peripherals & Middleware Wizard

Clock Tree wizard

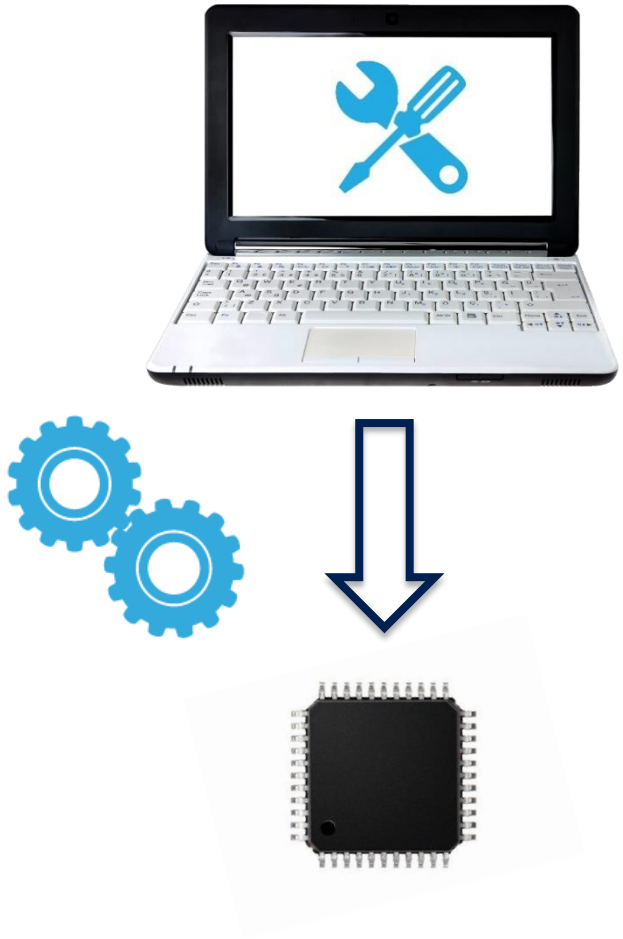


Power Consumption Wizard



life.augmented

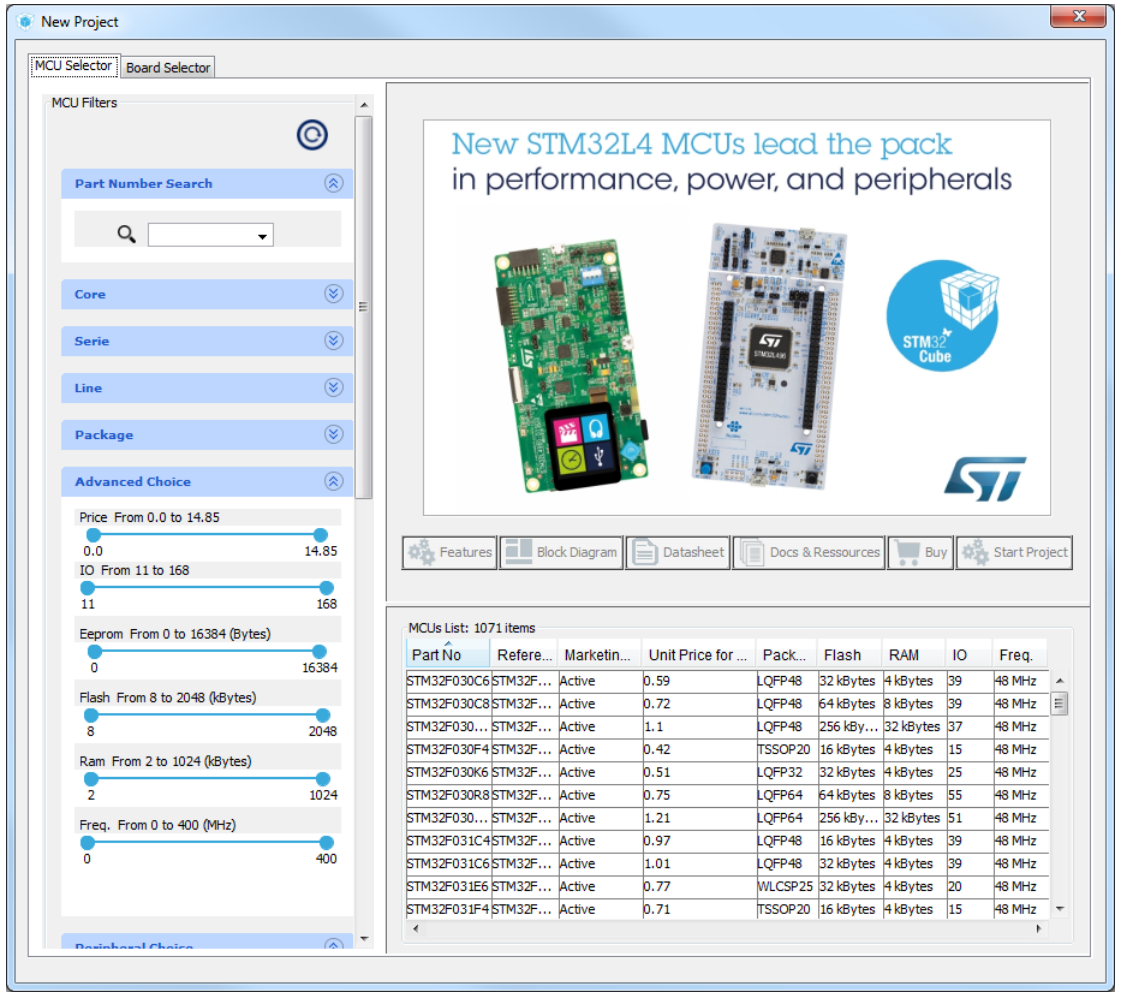
STM32CubeMX : User Code Configuration



Generates Initialization C Code based on user driven configuration!

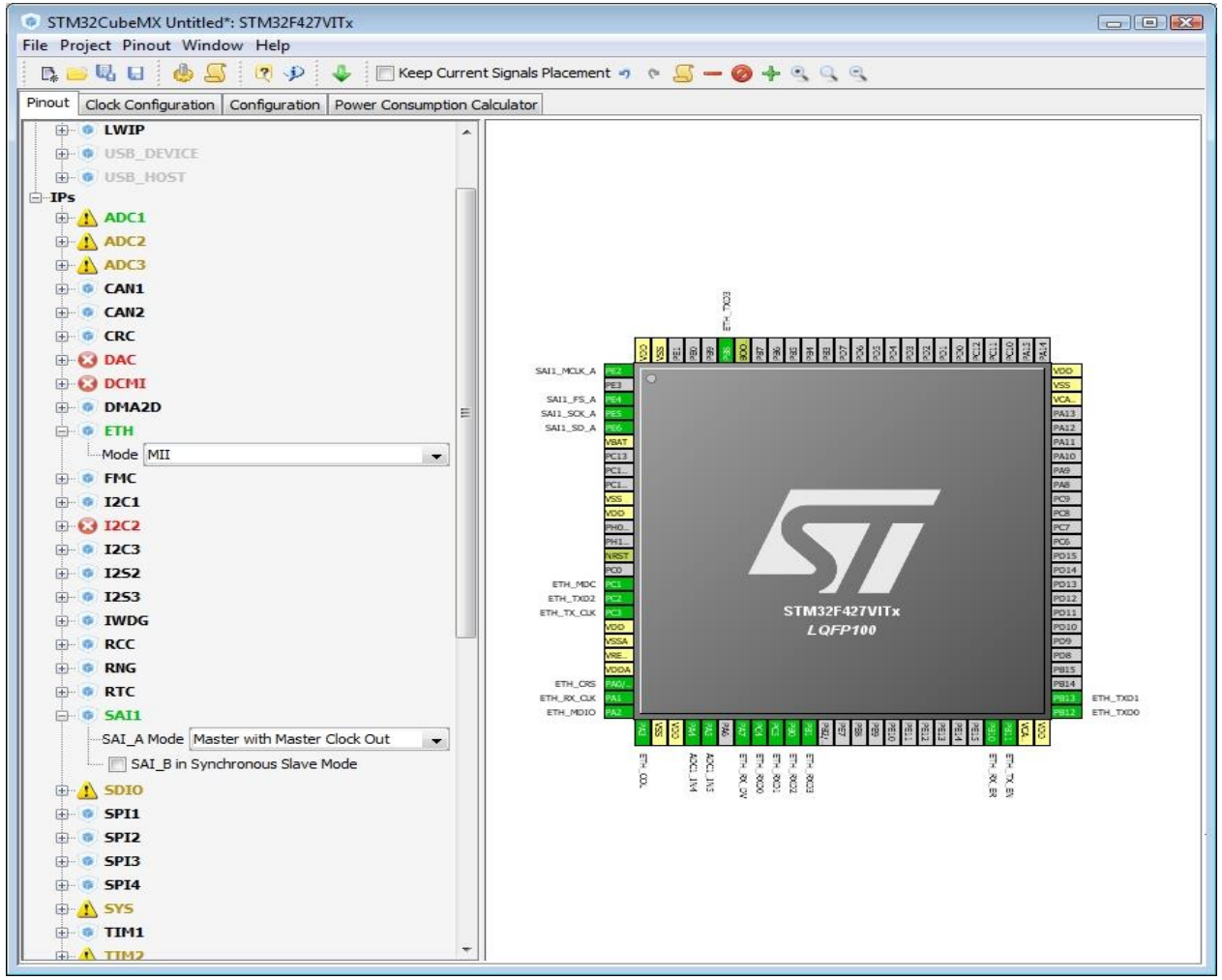
STM32CubeMX : MCU Selector

- Filter by:
  - Series
  - Line
  - Package
  - Peripherals



STM32CubeMX : Pin-out Configuration

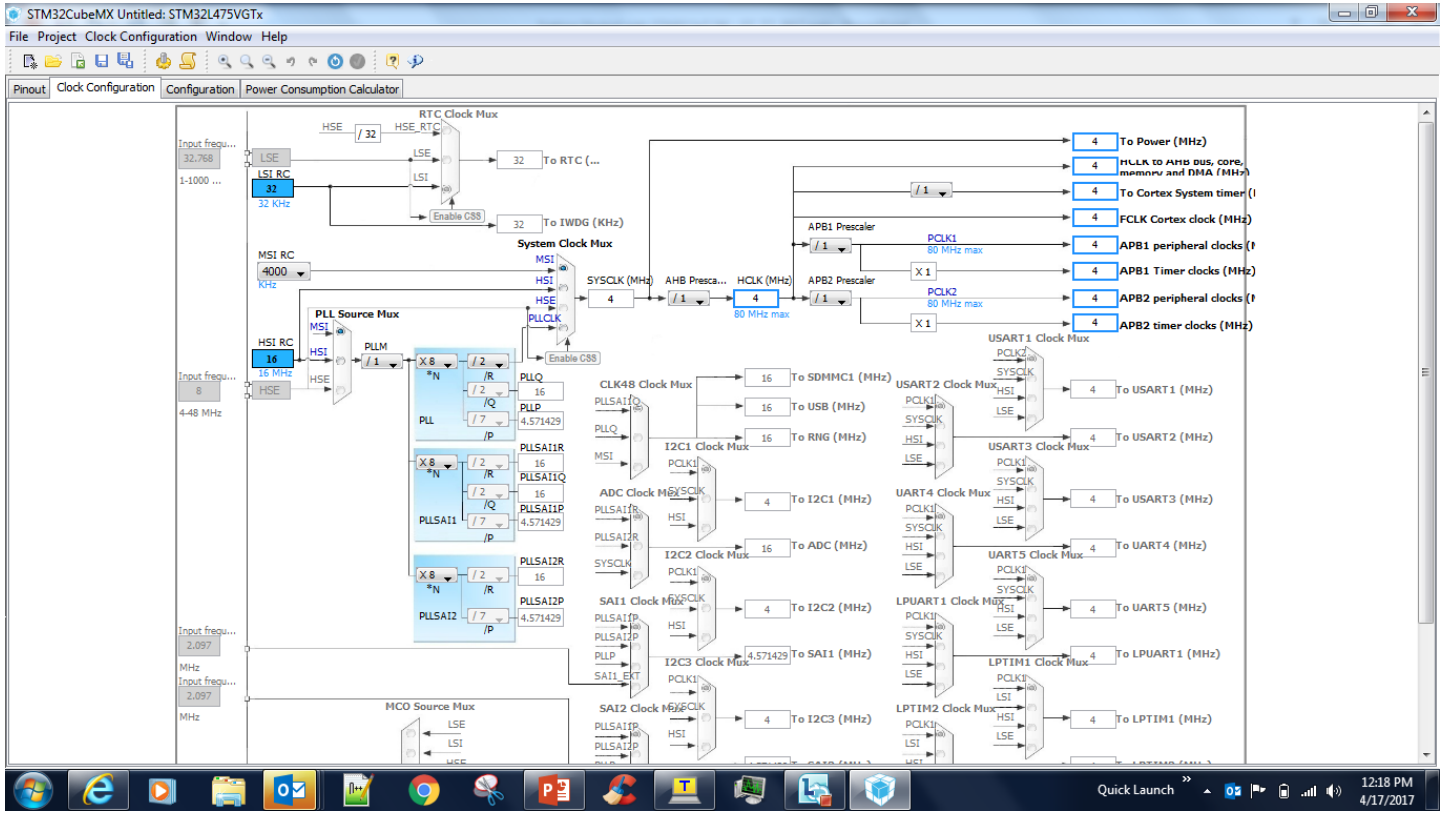
- Pinout from:
  - Peripheral tree
  - Manually
- Automatic signal remapping
- Management of dependencies between peripherals



life.augmented

STM32CubeMX : Clock Tree

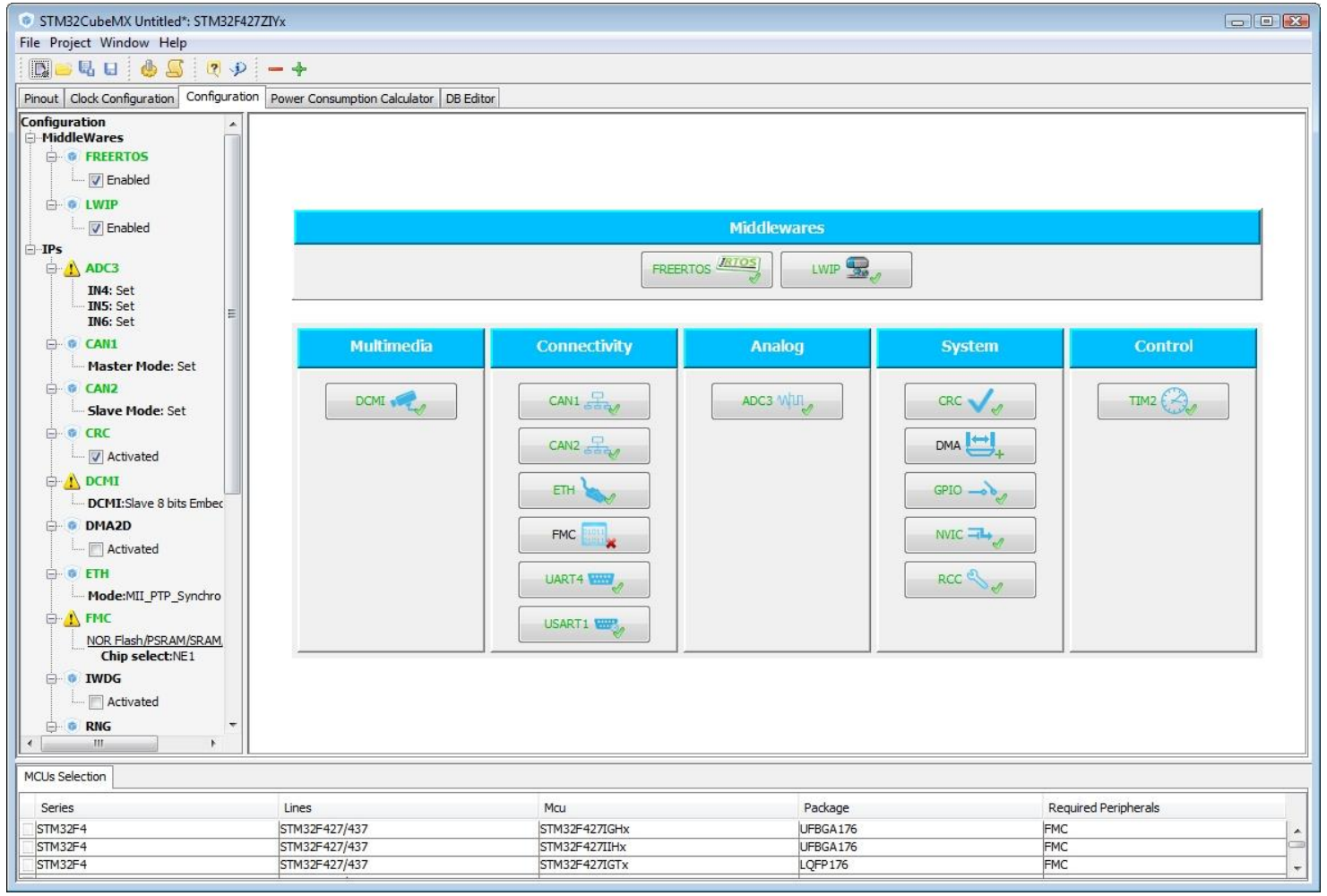
- Immediate display of all clock values
- Management of all clock constraints
- Highlight of errors



life.augmented

STM32CubeMX : Peripheral Configuration

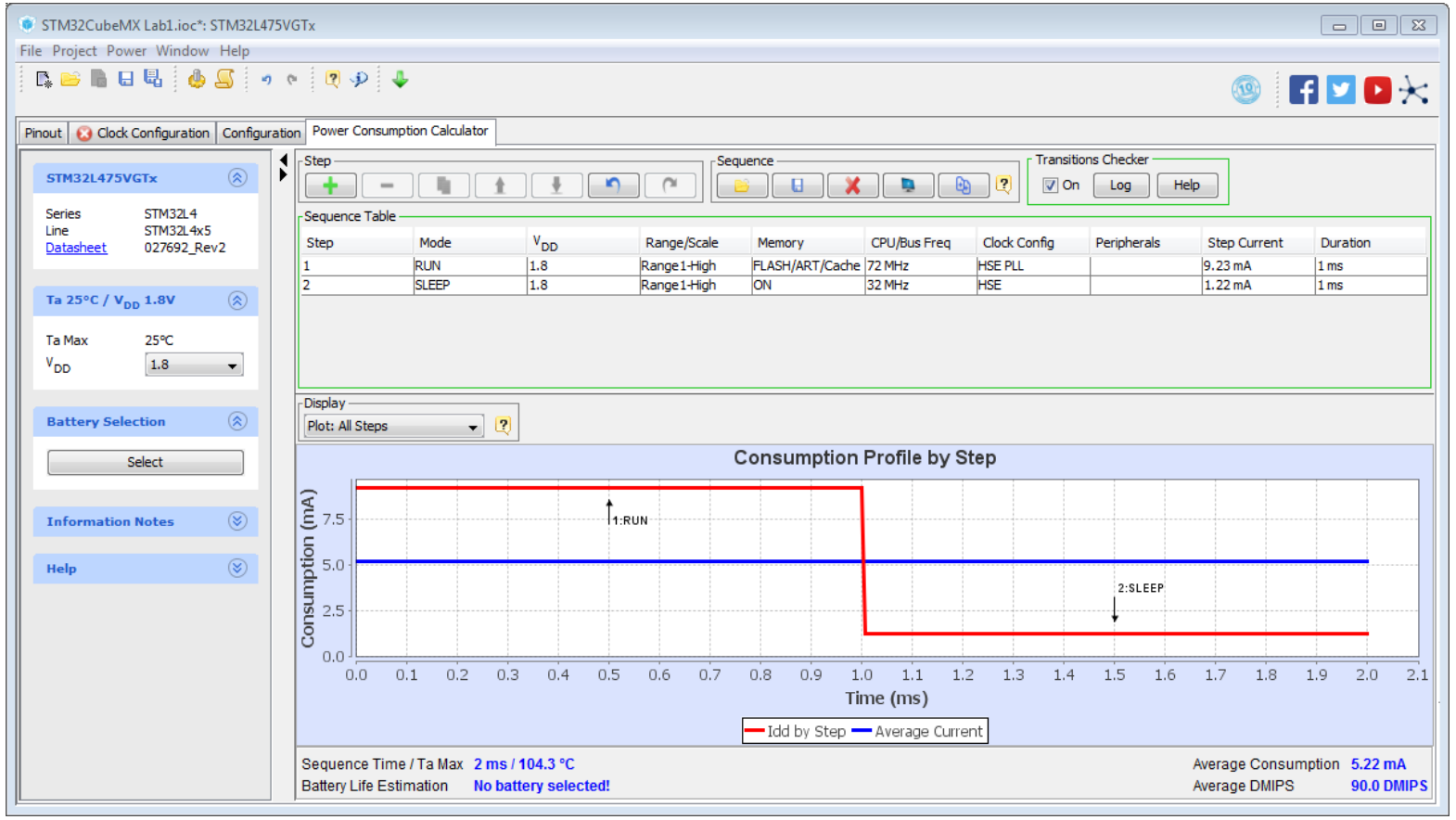
- Global view of used peripherals and middleware
- Highlight of configuration errors
- Manage:
  - GPIO
  - Interrupts
  - DMA



life.augmented

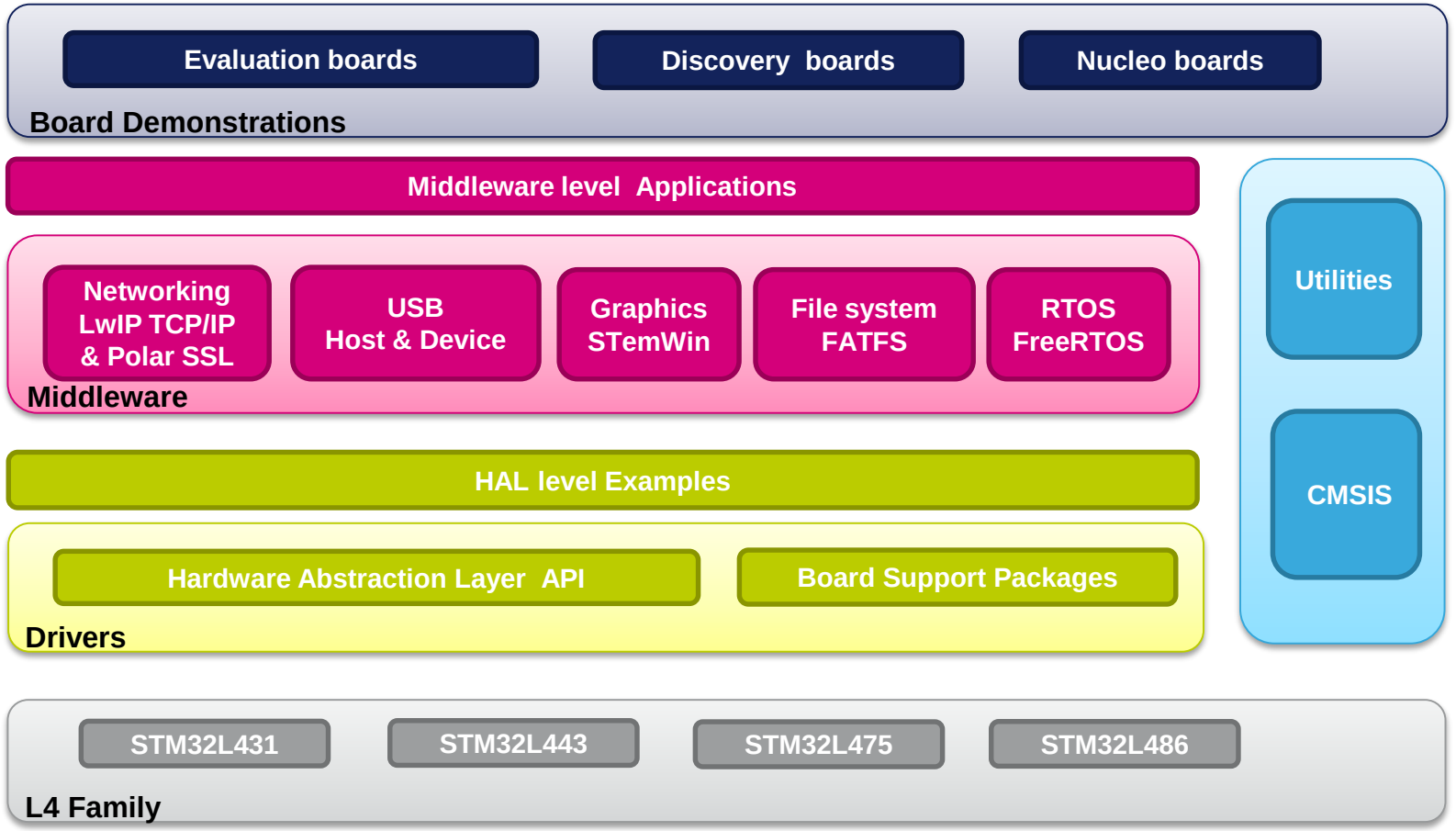
STM32CubeMX : Power Consumption Calculator

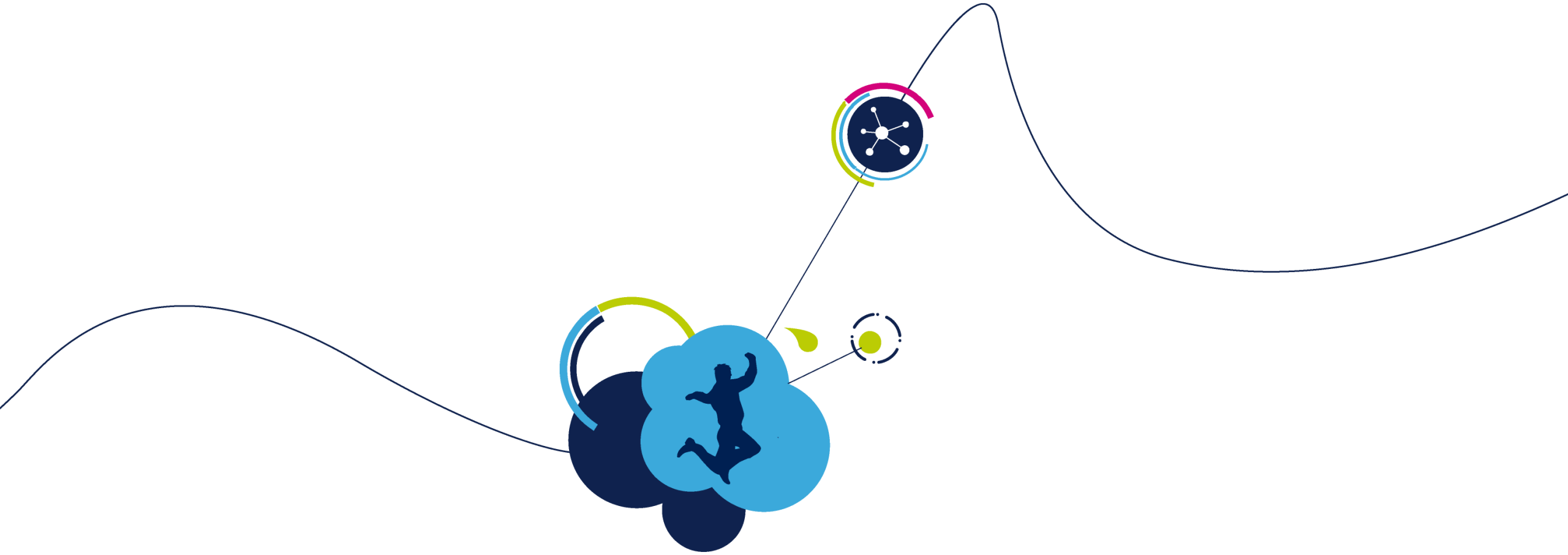
- Power step definitions
- Battery selection
- Creation of consumption graph
- Display of
  - Average consumption
  - Average DMIPS
  - Battery lifetime



life.augmented

STM32Cube : Firmware Components





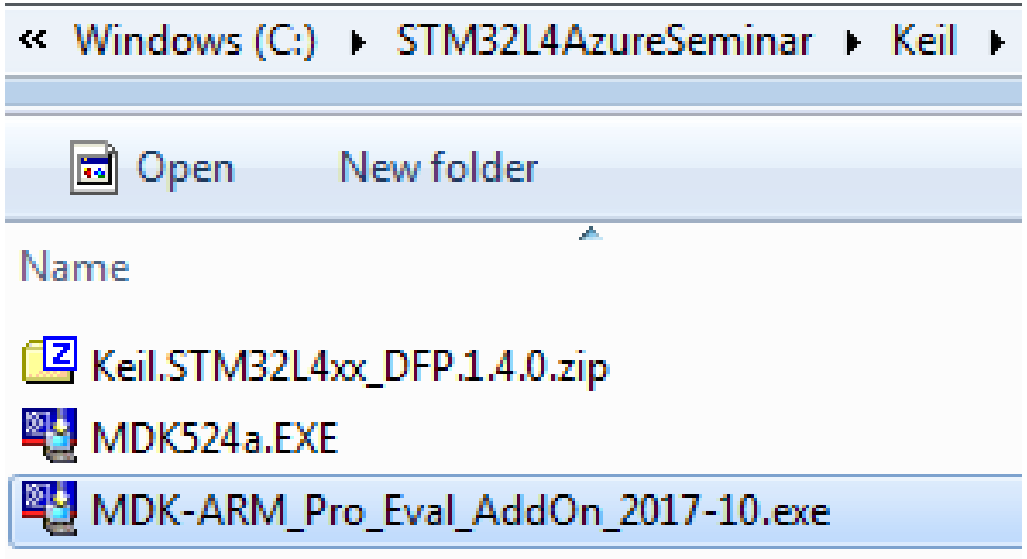
Keil License Installation

life.augmented

Step 2. Keil License Installation

- Browse to **C:\STM32L4AzureSeminar\Keil** and double-click on the file

MDK-ARM_Pro_Eval_AddOn_2017-10.exe

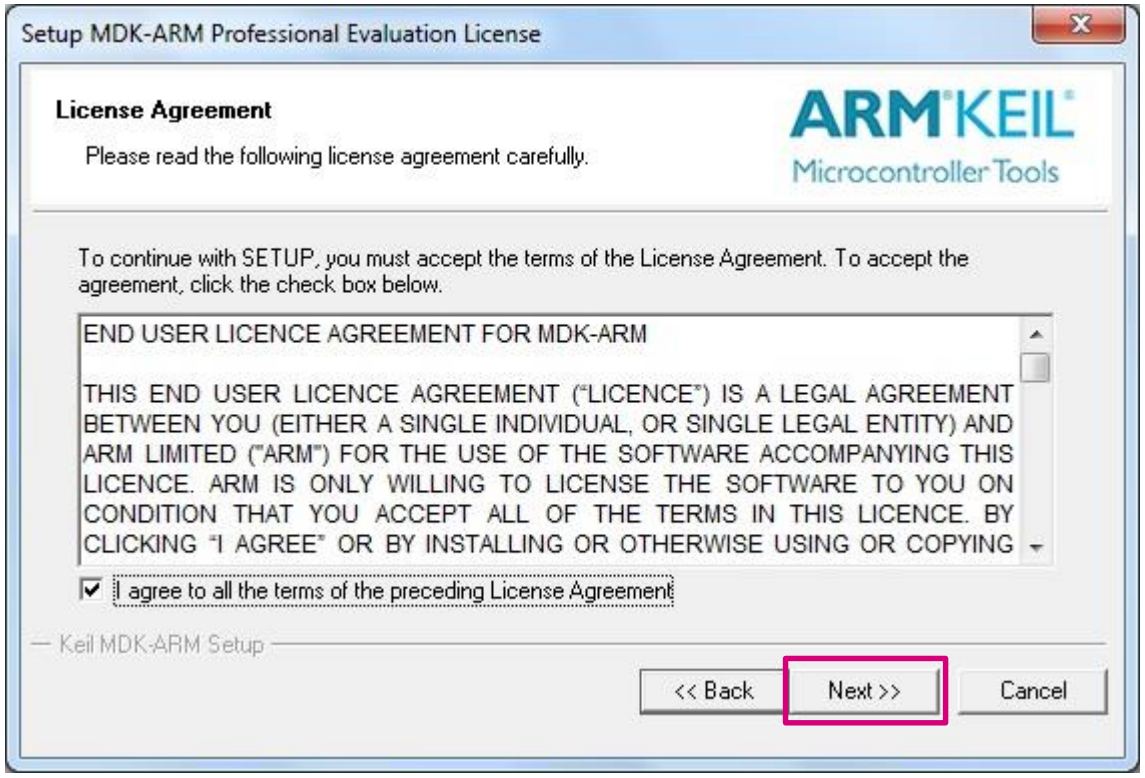


Step 2. Keil License Installation

- Click on Next

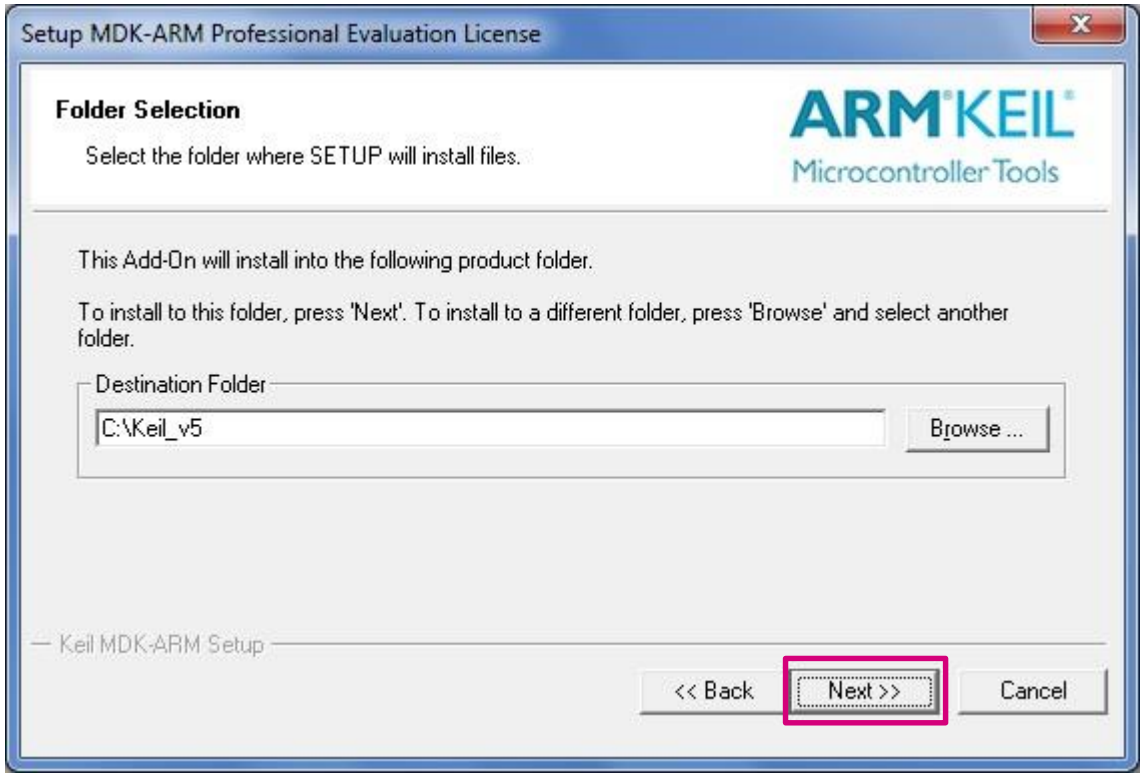
Step 2. Keil License Installation

- Accept the license agreement and click on Next



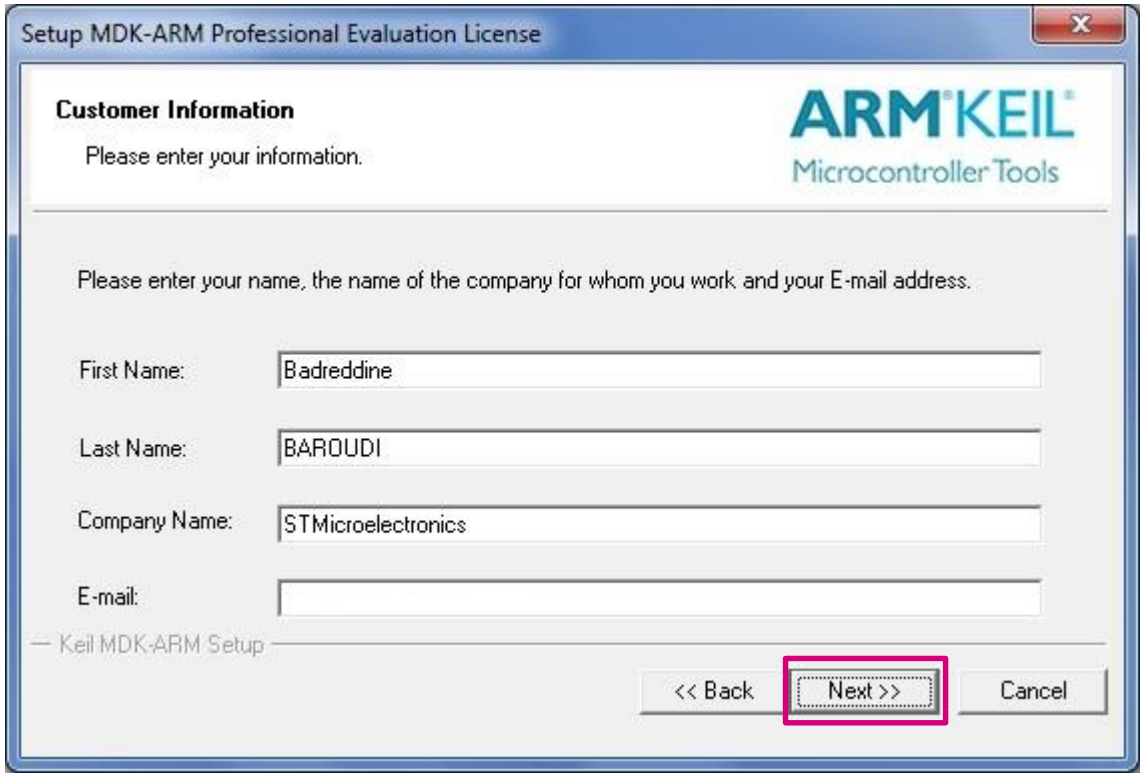
Step 2. Keil License Installation

- Click on Next



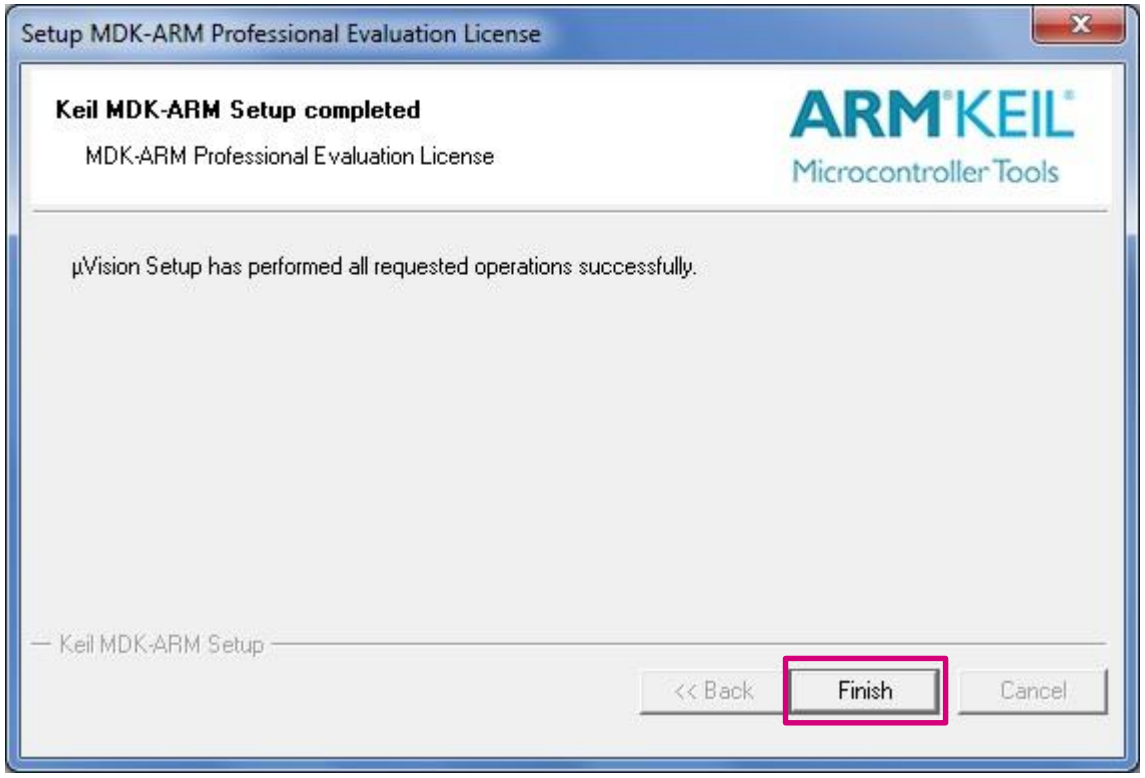
Step 2. Keil License Installation

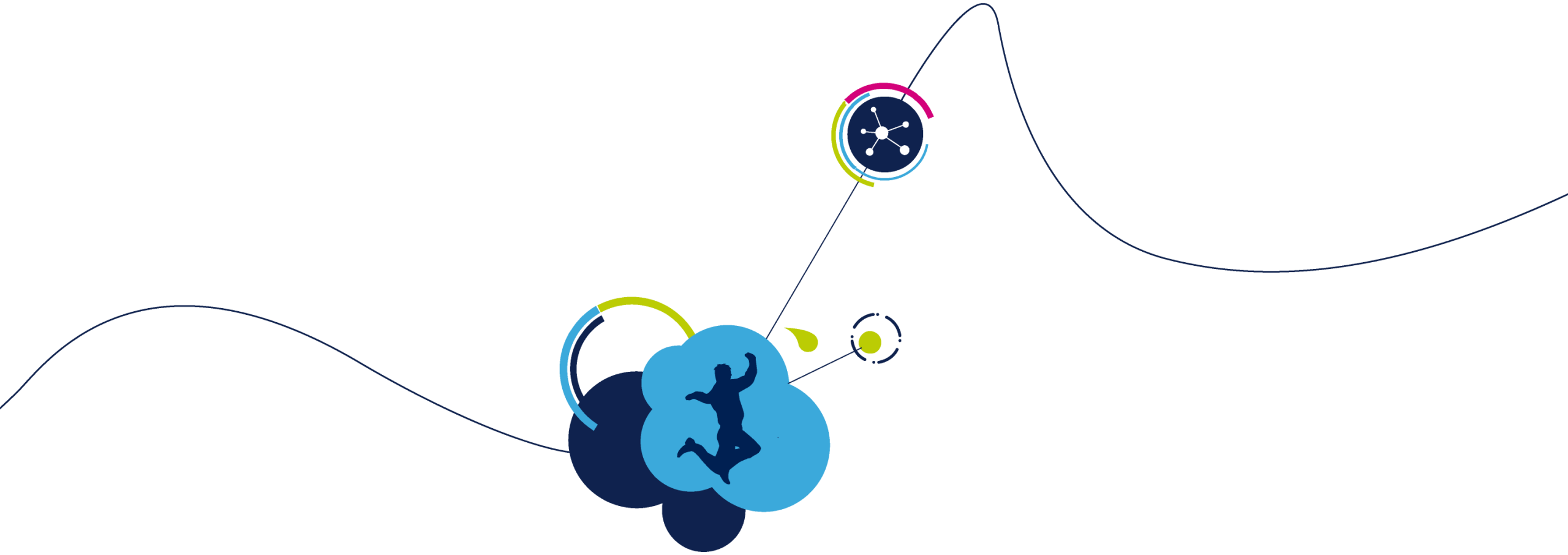
- Enter your contact information and click on Next



Step 2. Keil License Installation

- Click on Finish



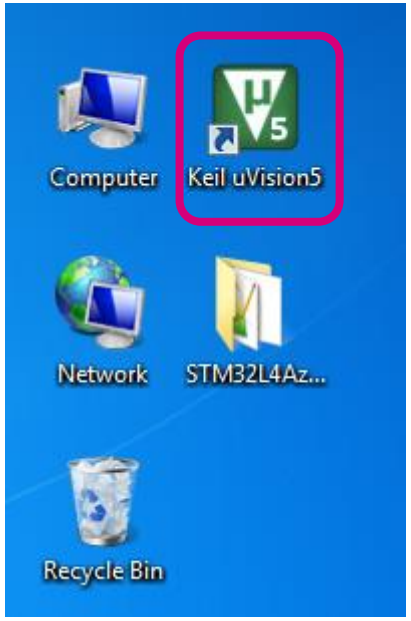


Keil Software Pack Installation

life.augmented

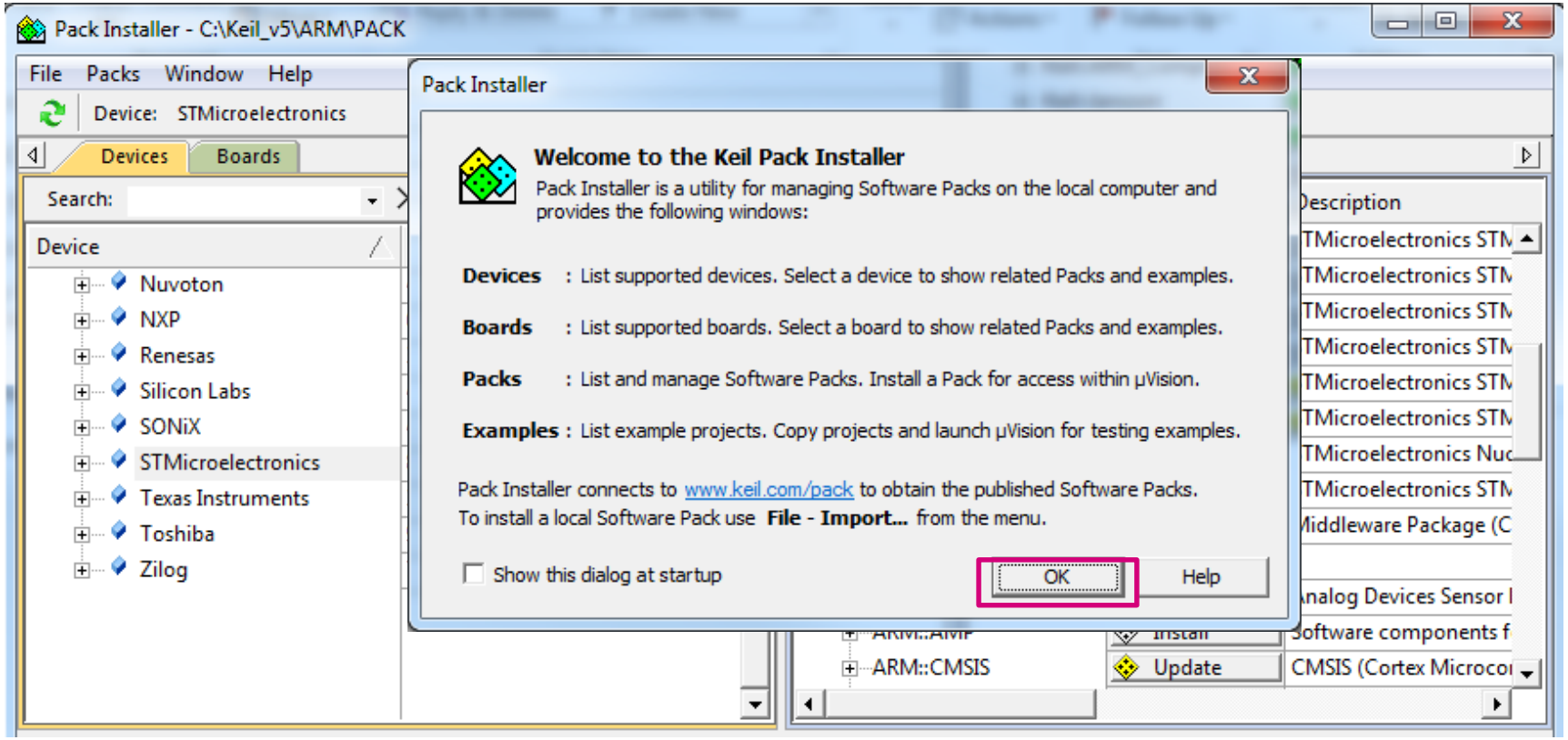
Step 3. Keil Software Packs

- Browse to your Desktop and double-click on Keil uVision5



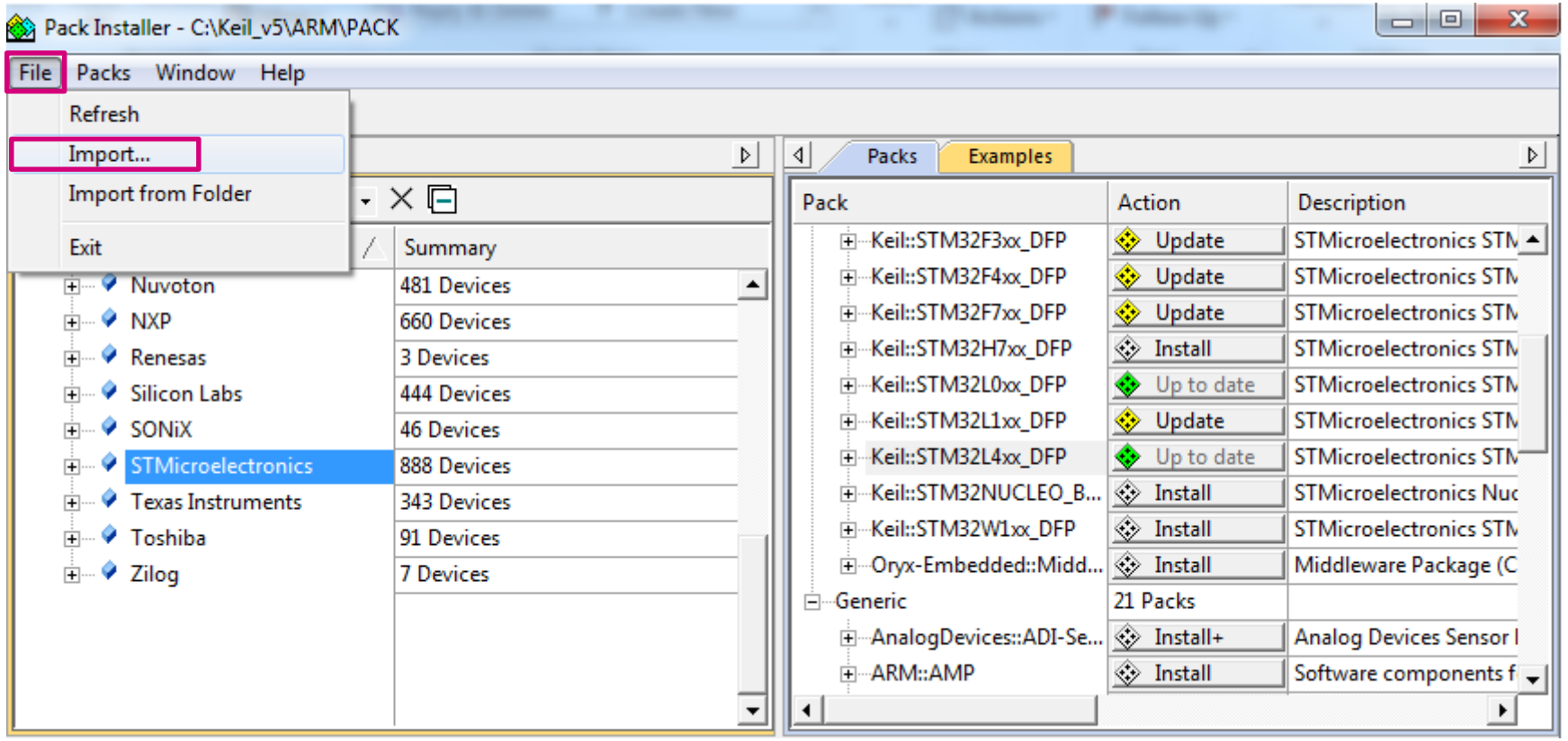
Step 3. Keil Software Packs

- Keil will automatically prompt you to install the Device Family Packs.
- Click on OK
- If the Pack Installer doesn't show up, click on Pack Installer icon 



Step 3. Keil Software Packs

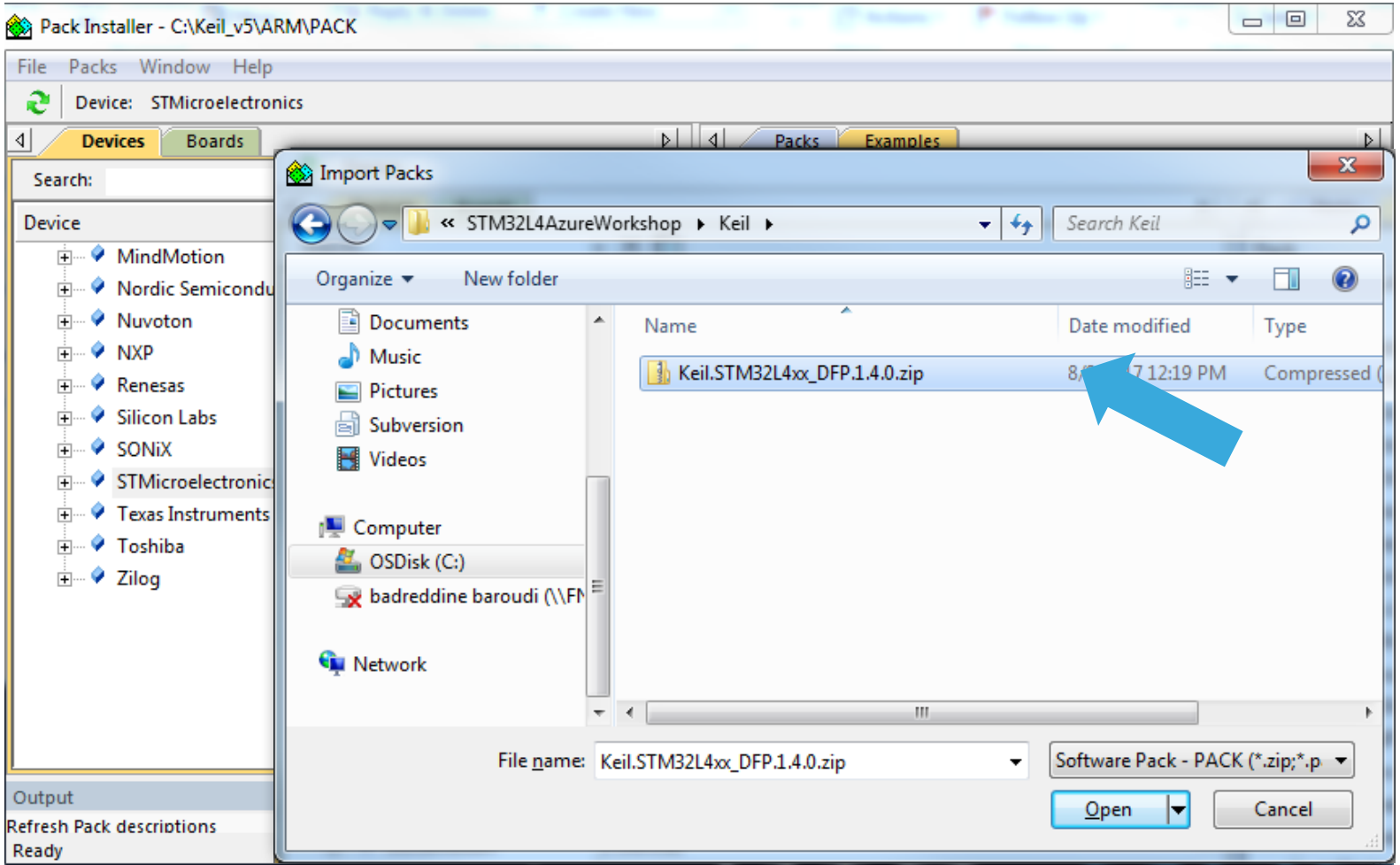
- We will manually import the STM32L4xx_DFP software pack…
- In Pack Installer, Select File and then Import



Step 3. Keil Software Packs

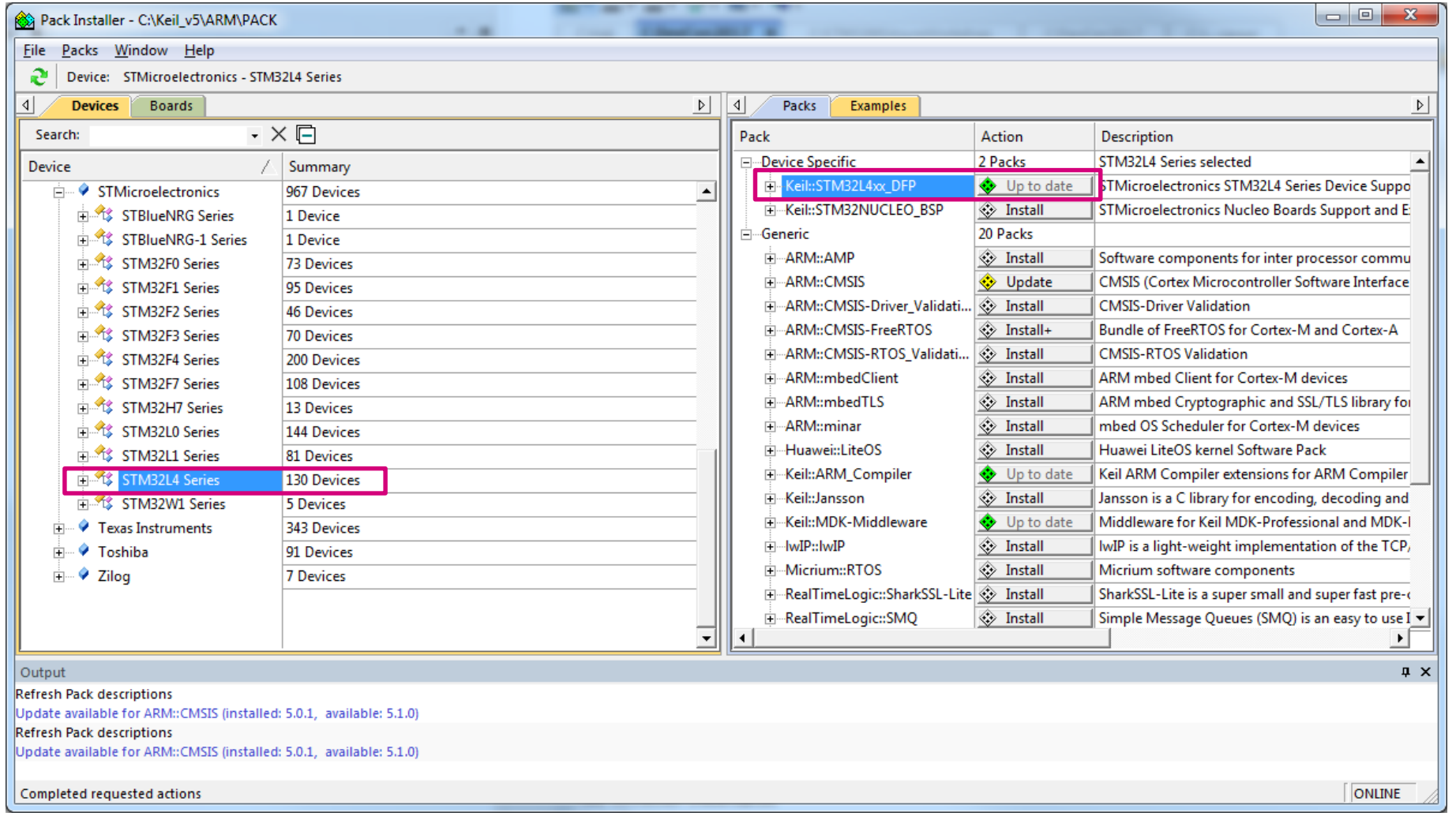
Browse to **`C:\STM32L4AzureSeminar\Keil`**

And then choose Keil.STM32L4xx_DFP.1.4.0.zip



Step 3. Keil Software Packs

After the STM32L4 Pack is installed, the Pack installer can be closed



Board Distribution

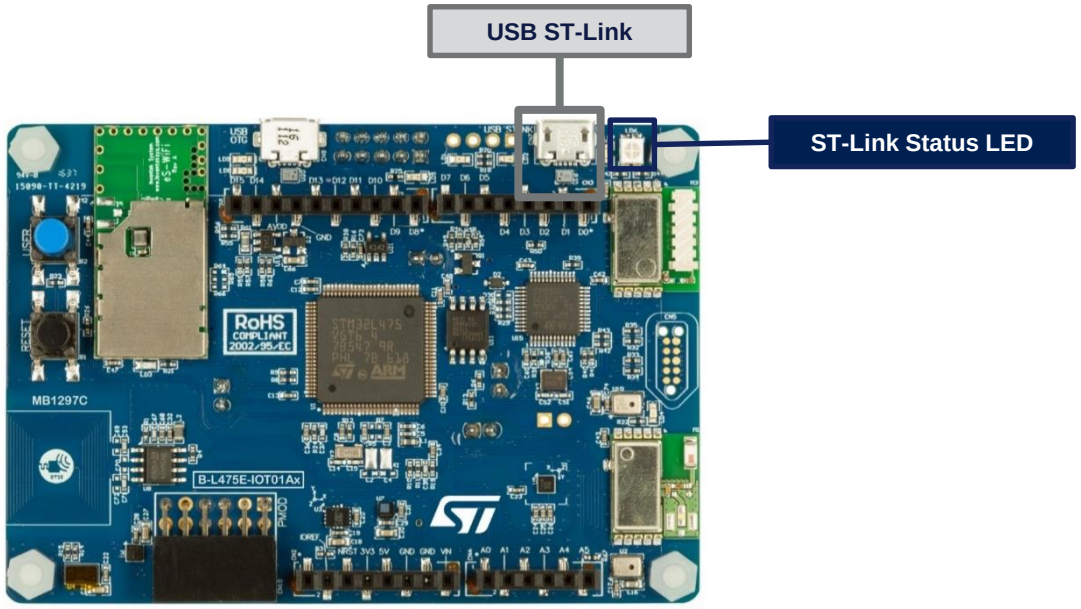
Participant Number
(2 characters)

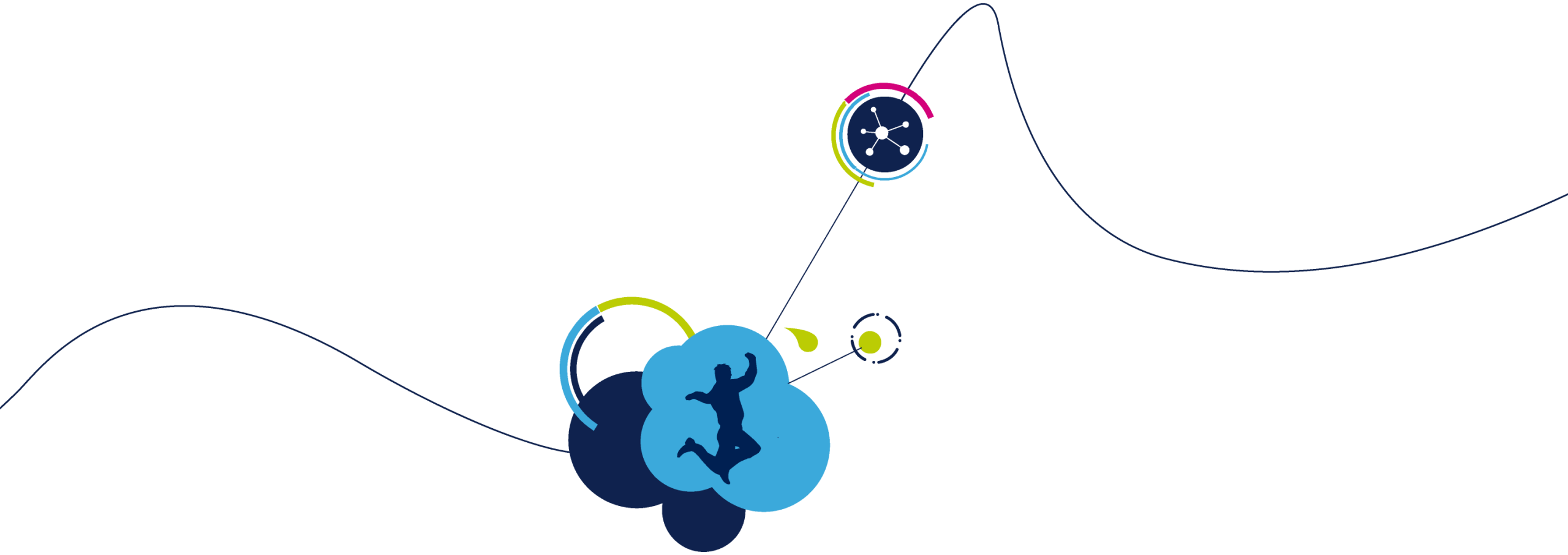
Each board package will have a label with a unique number.
During the lab sessions, this number will be referred to as your Participant Number.

life.augmented

ST-Link Driver Installation

- Connect your PC to the USB ST-Link.
- The board will be powered through the ST-Link connection.
- The ST-Link Status LED will be steady when ST-Link is recognized.

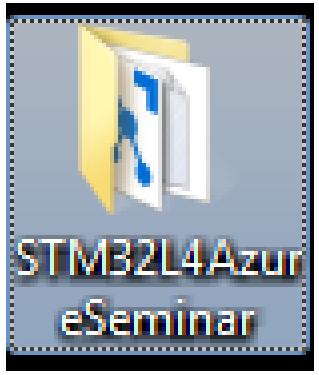


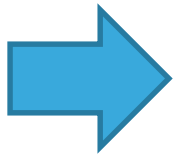


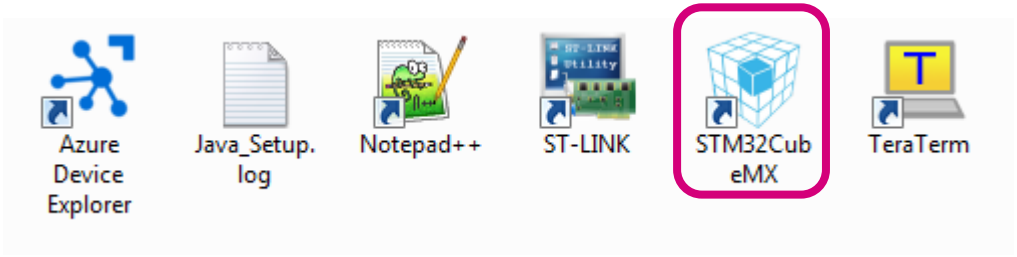
Lab 1: Getting Started with STM32CubeMX – Blinking LED

life.augmented

Launch STM32CubeMX



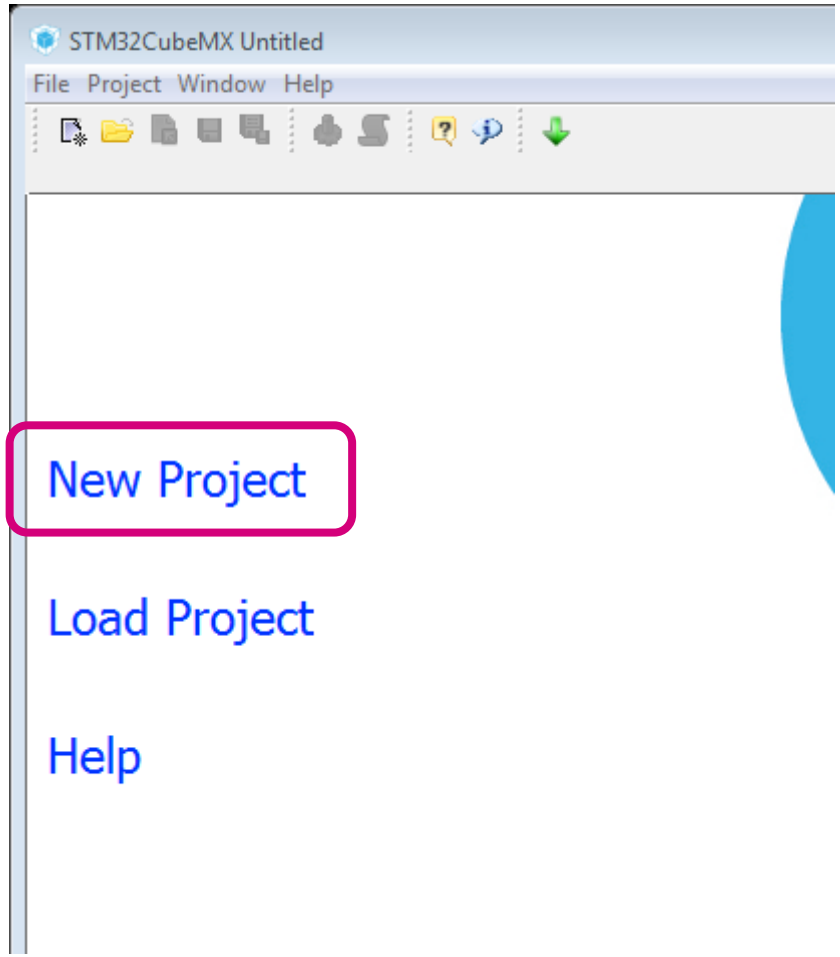




- Open the STM32L4AzureSeminar folder on your Desktop.
- Double click on the **STM32CubeMX** icon.

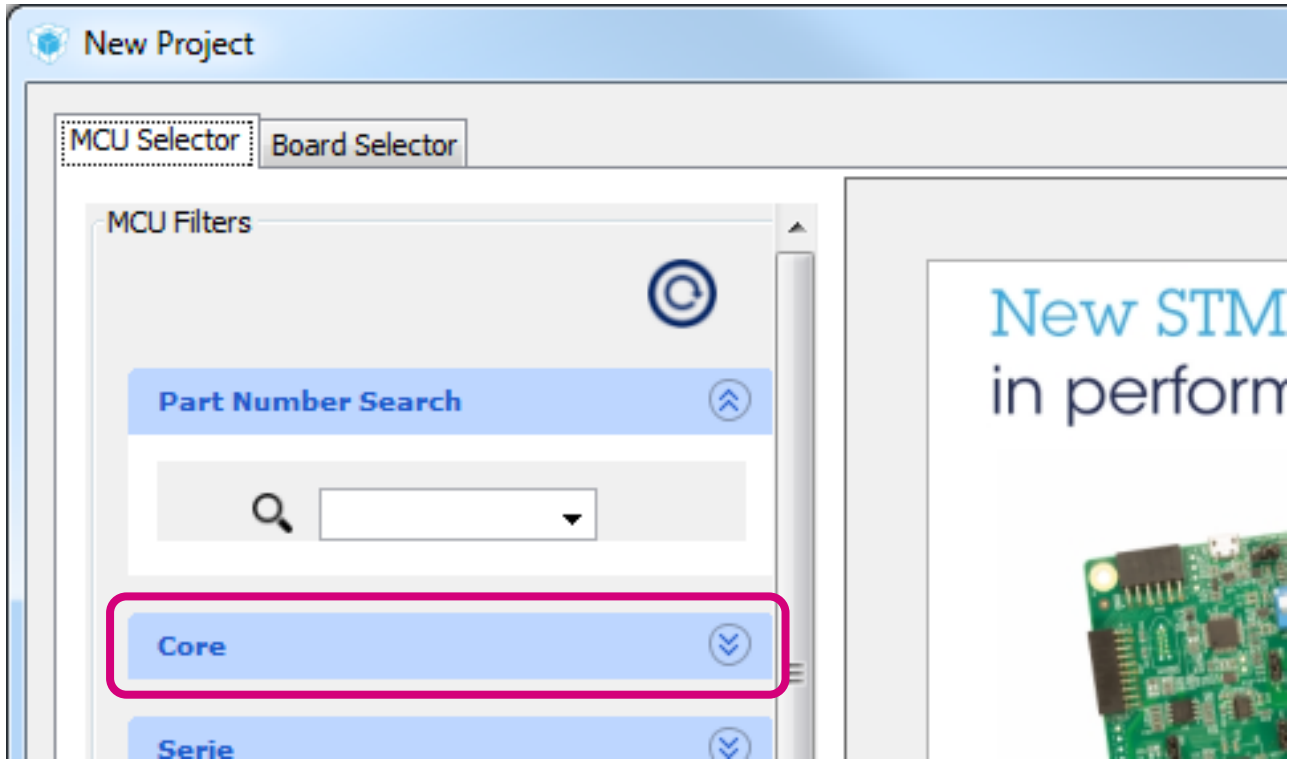
Create New STM32CubeMX Project

- Click on **New Project**



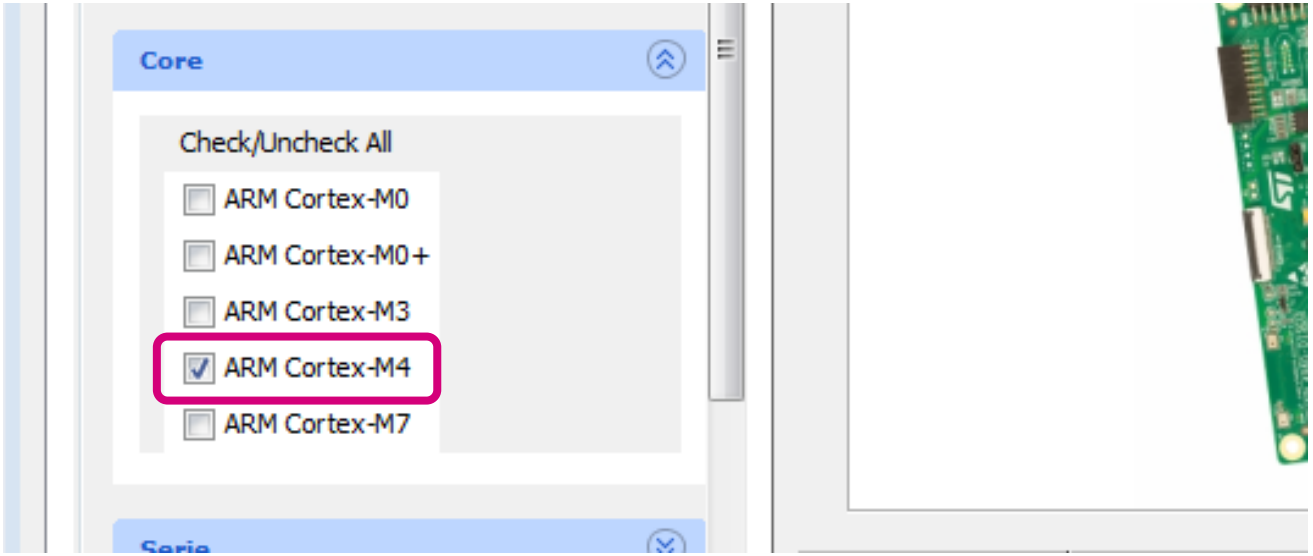
Select the Microcontroller

- Expand **Core**



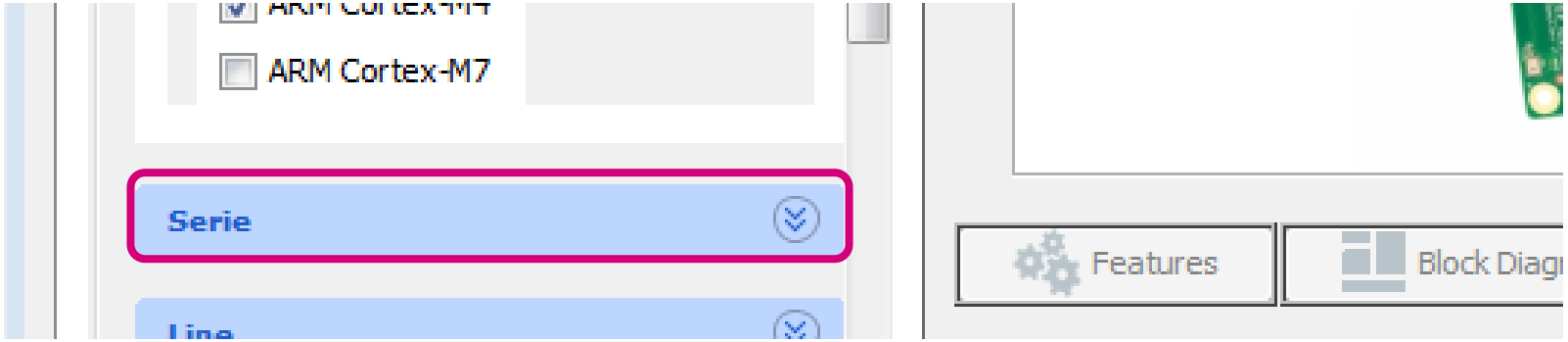
Select the Microcontroller

- Select **ARM Cortex-M4**



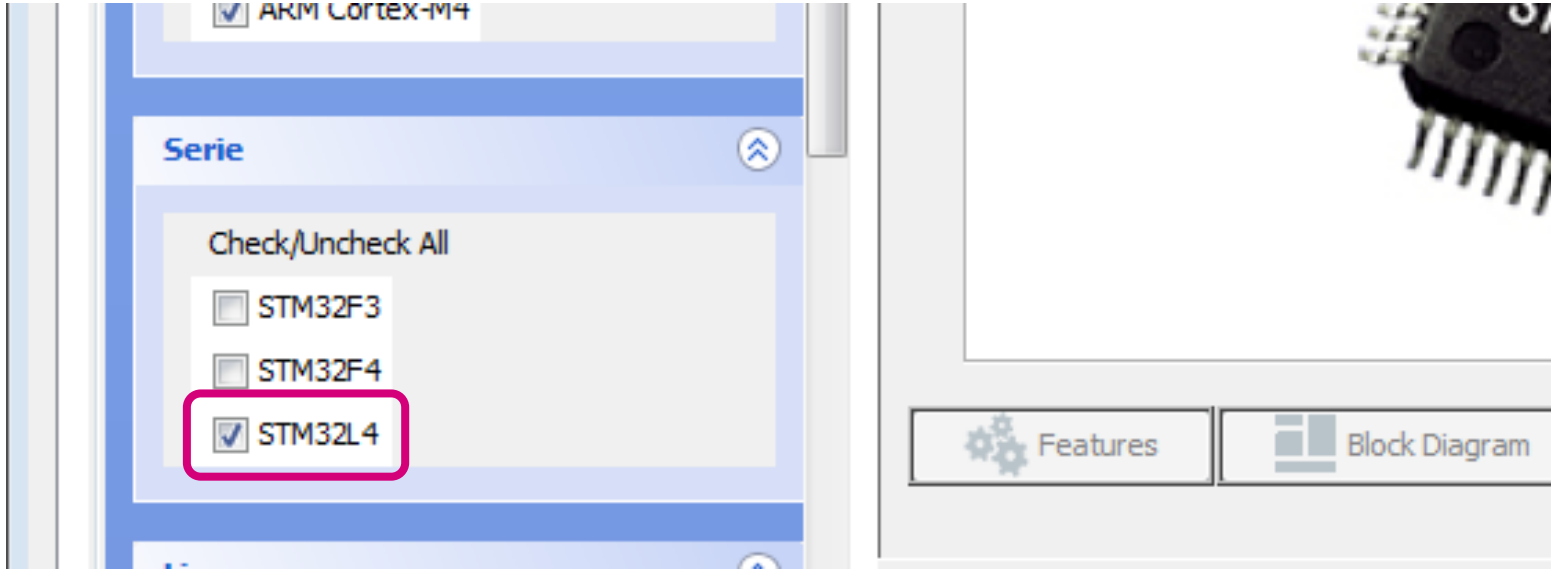
Select the Microcontroller

- Expand **Series**



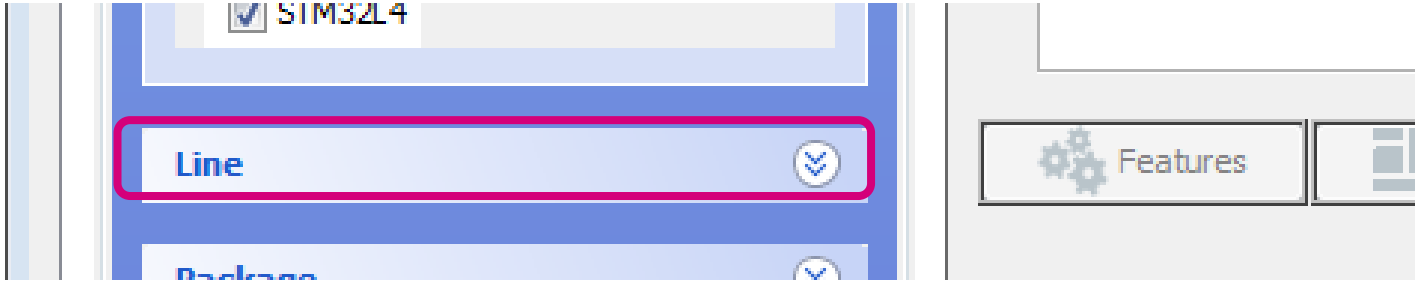
Select the Microcontroller

- Select **STM32L4**



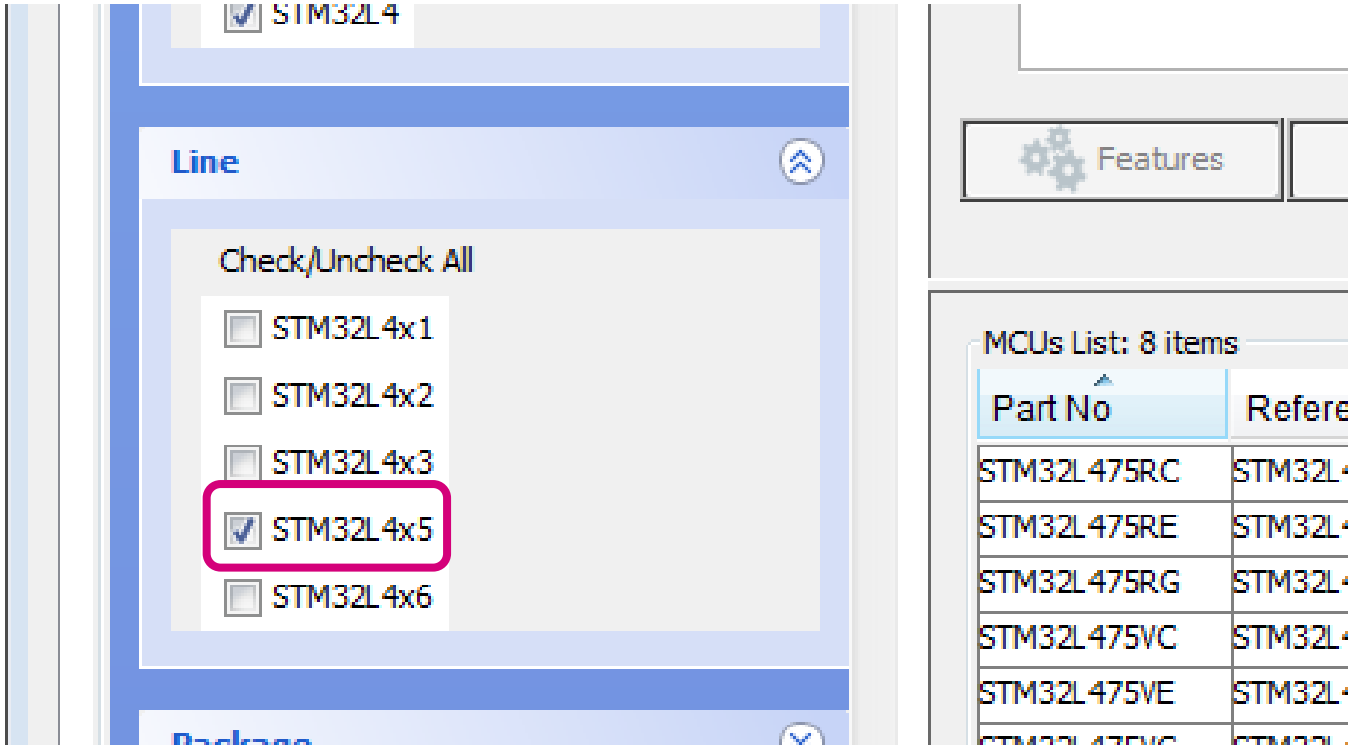
Select the Microcontroller

- Expand **Line**



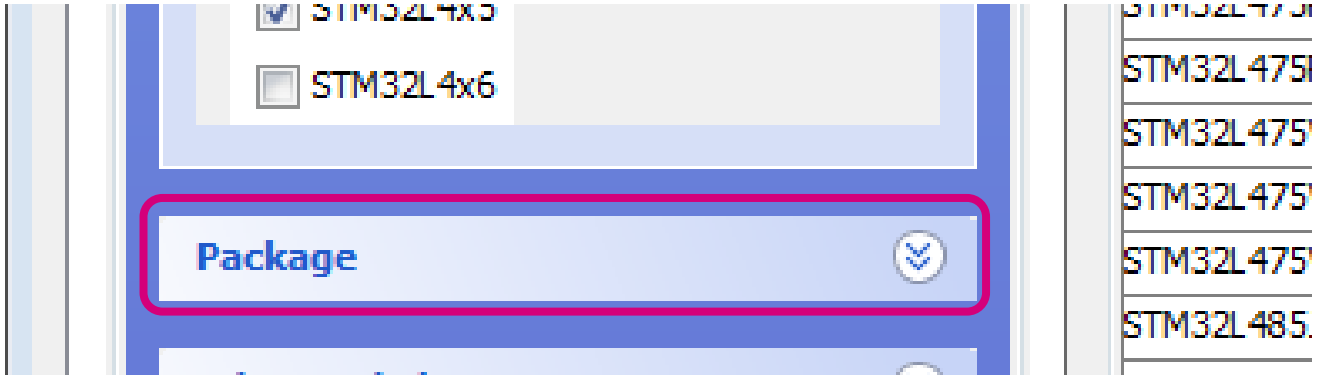
Select the Microcontroller

- Select **STM32L4x5**

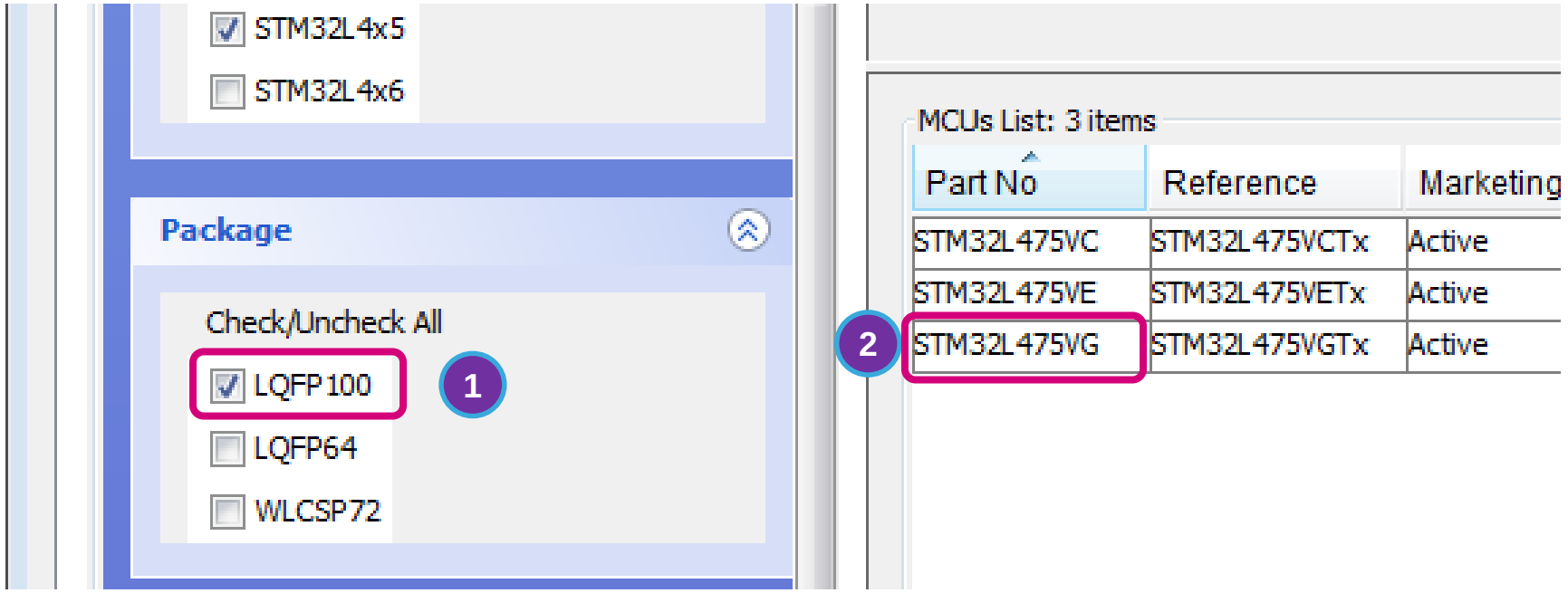


Select the Microcontroller

- Expand **Package**



Select the Microcontroller



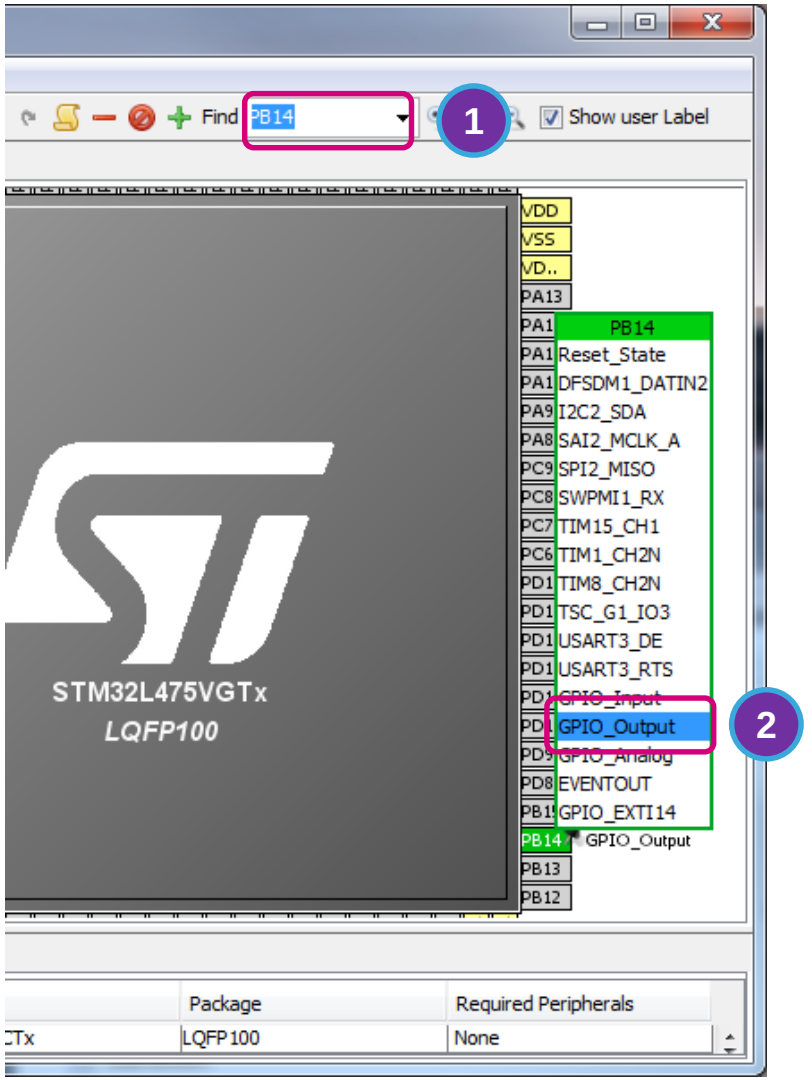
1. Select **LQFP100**
2. Double-click on **STM32L475VG**

GPIO Selection

- This example will use **LED2** on the DK IoT board.

1. Use the find toolbar and type **PB14**
2. Select **PB14** and set it to **GPIO_Output** mode.

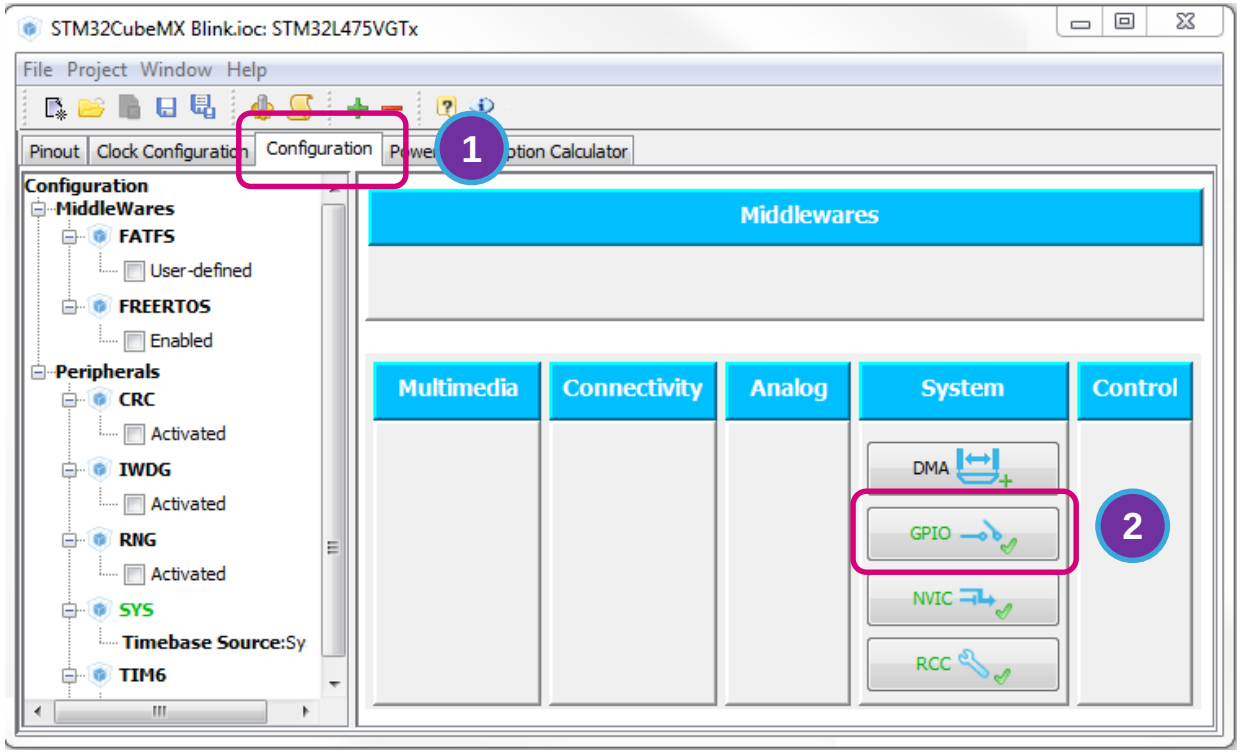
Note: Board schematics can be found in the IoT board User Manual (UM2153)



life.augmented

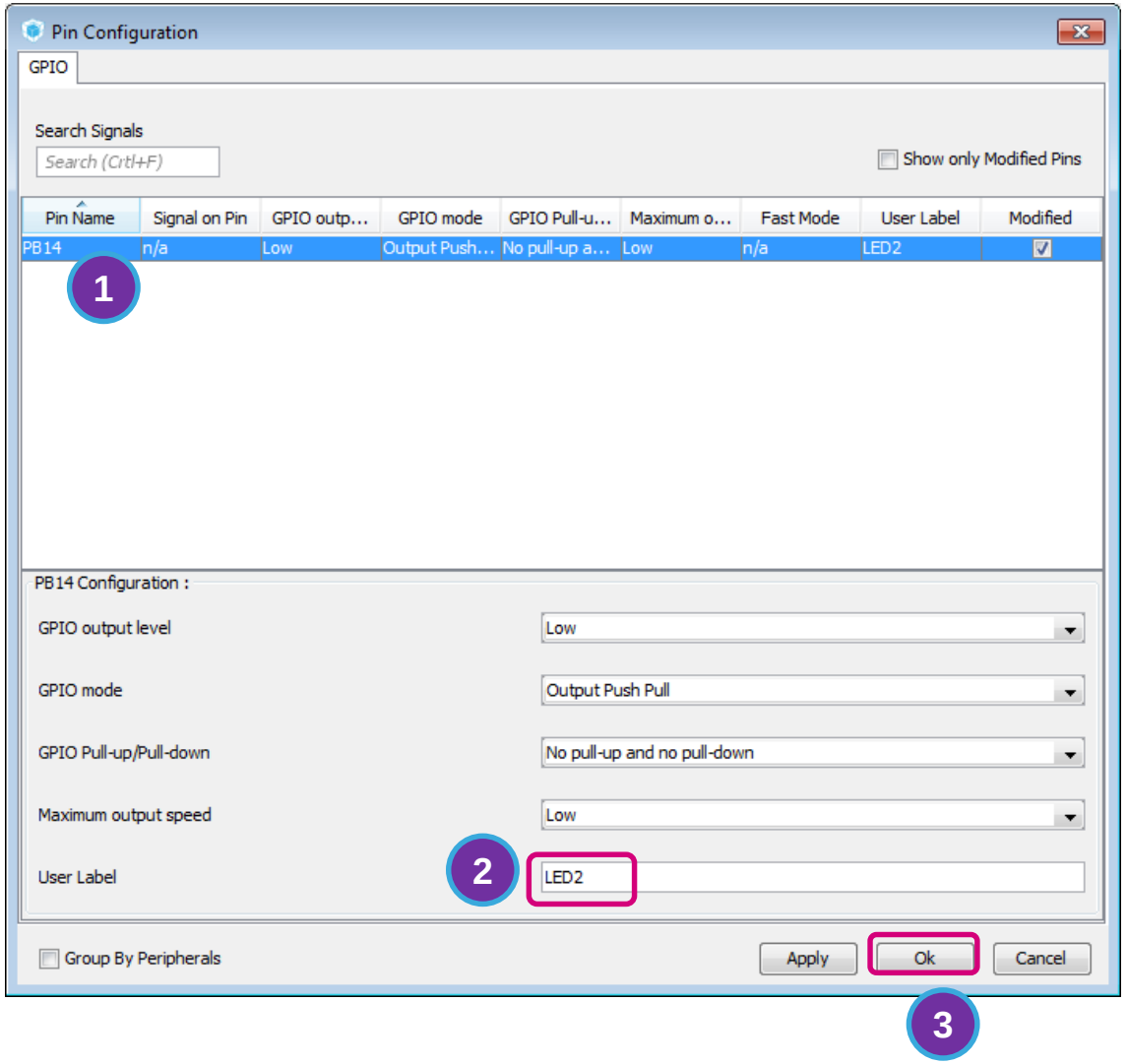
GPIO Configuration

1. Select the **Configuration** tab
2. Select **GPIO** under System.



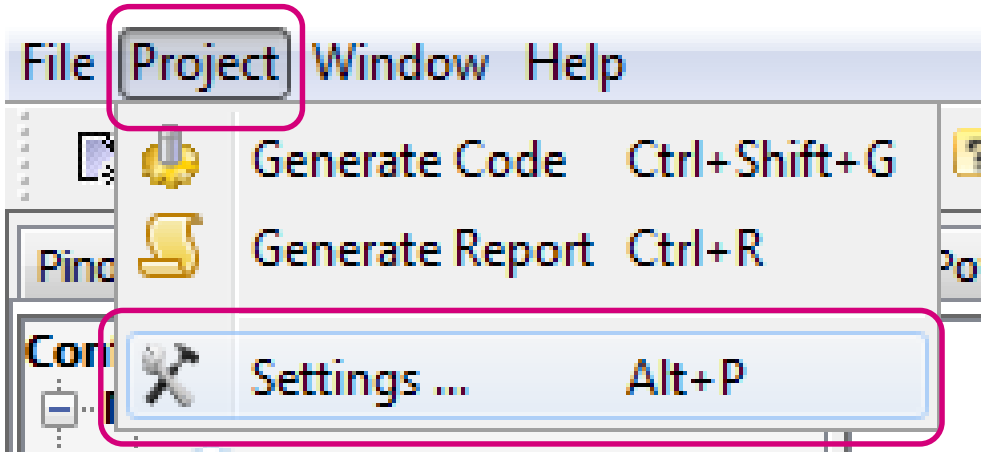
GPIO Configuration

1. Select **PB14**.
2. Set the User Label to **LED2.**
3. Click OK



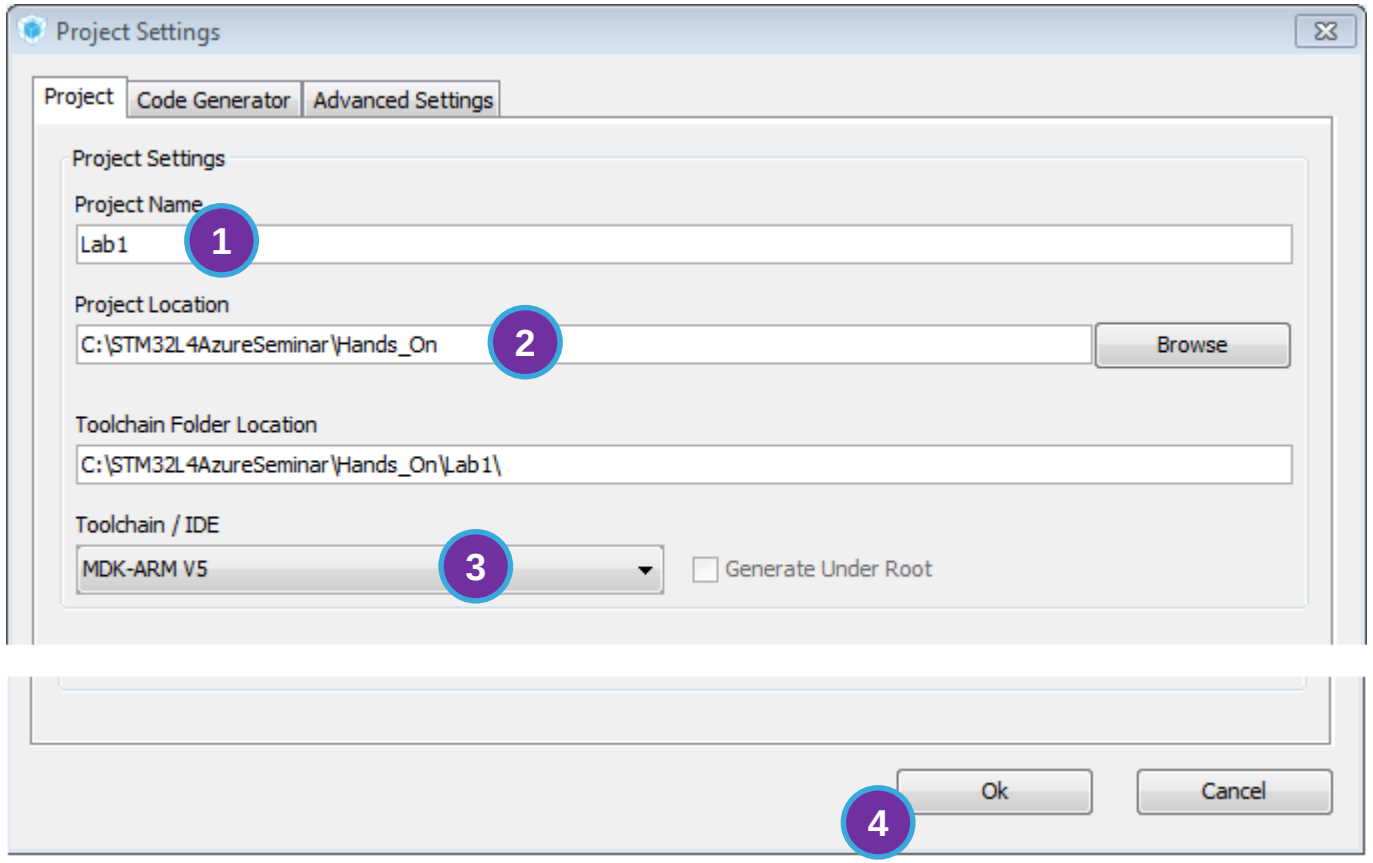
Project Settings

1. Select **Project** -> **Settings** (Alt + P)

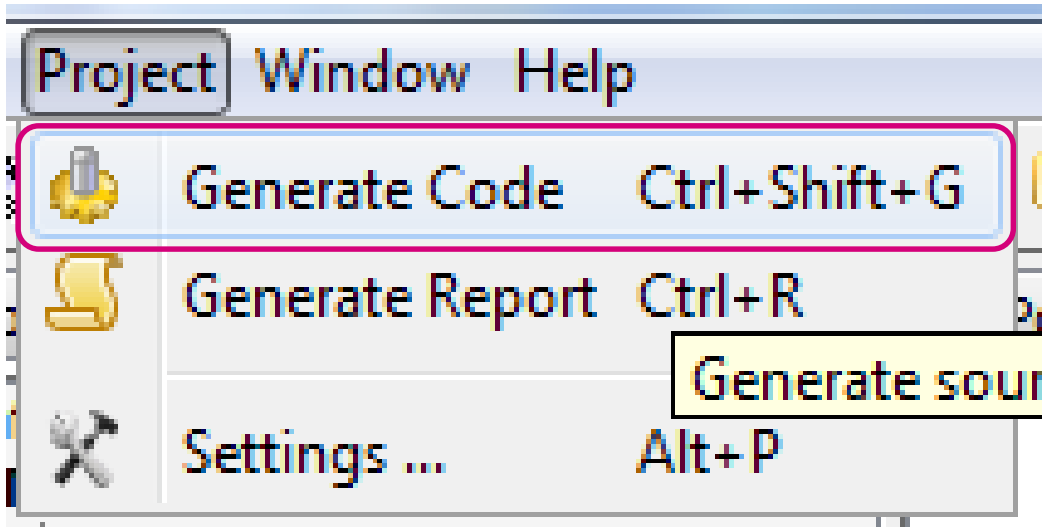


Project Settings

1. Set the project name to **Lab1**.
2. Set the project location:
   C:\STM32L4AzureSeminar\Hands_On
3. Set the IDE Toolchain to **MDK-ARM V5**.
4. Click **OK**.

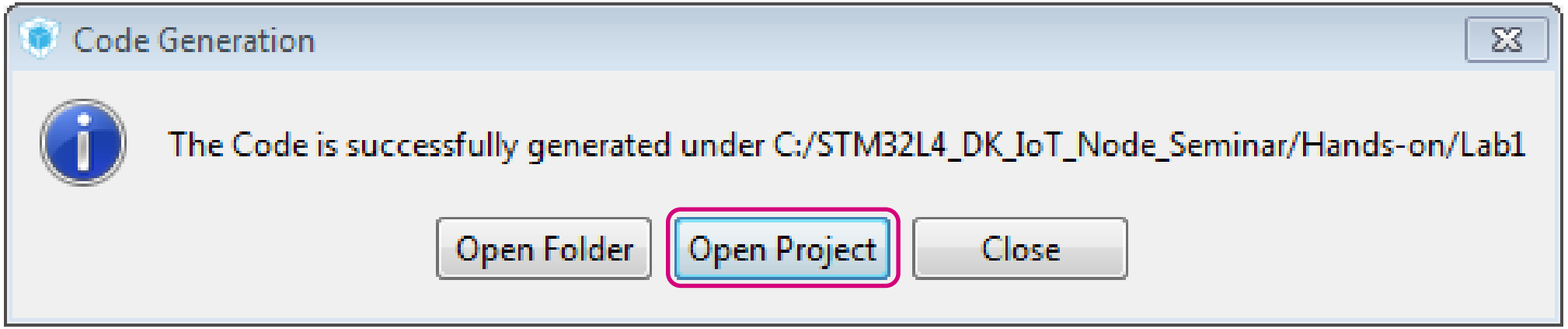


Generate the Project



- Click **Generate Code** (Ctrl + Shift + G)

Open the Project



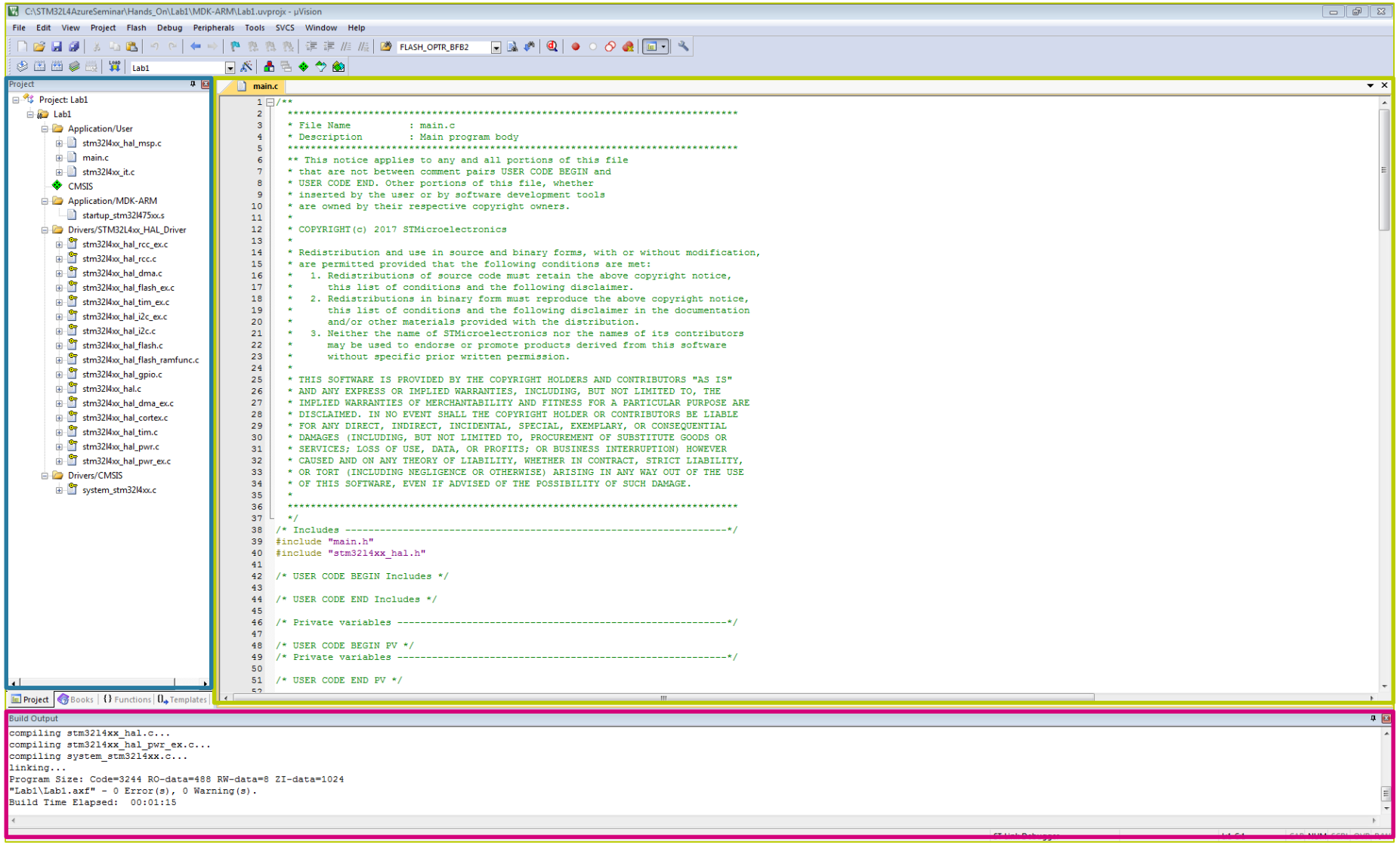
- Click **Open Project**

Inside Keil MDK-ARM

Project Window

File Editor Window

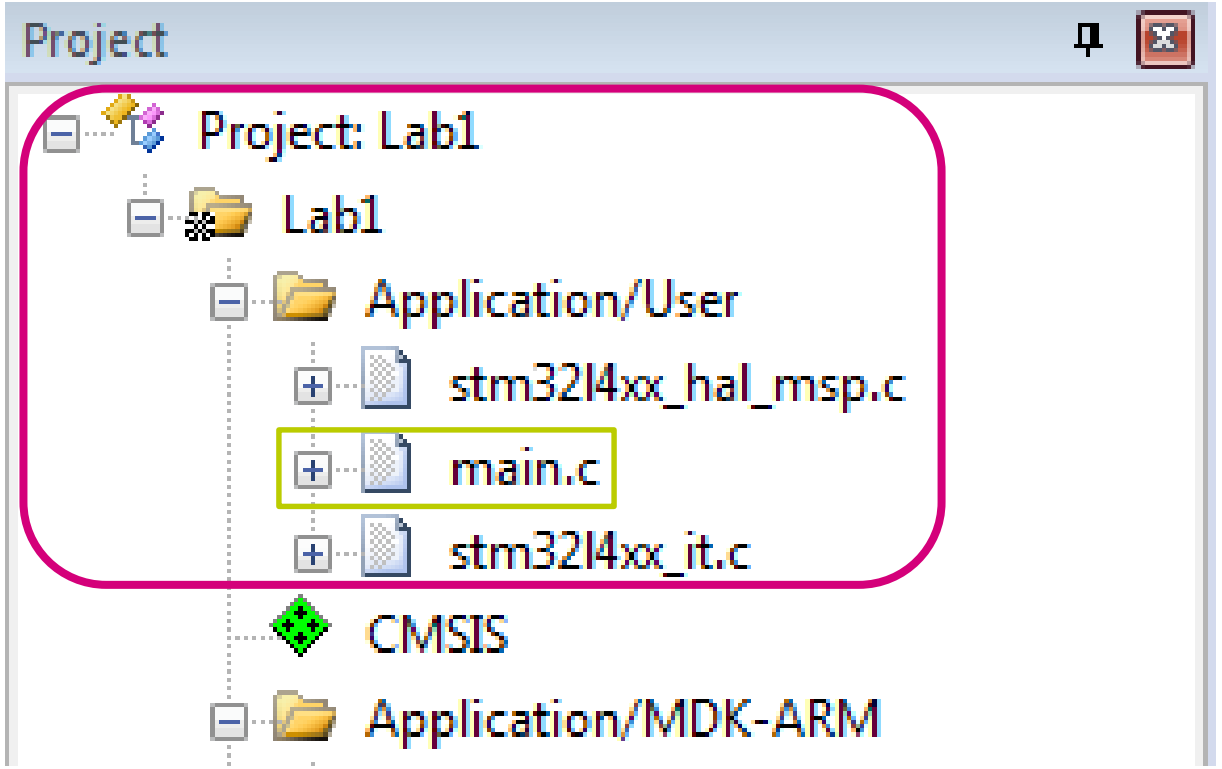
Build Output Window



life.augmented

Edit main.c

- Expand the file tree:
  1. Expand **Lab1**
  2. Expand **Application/User**
  3. Double-click **main.c**



Edit main.c

main.c*

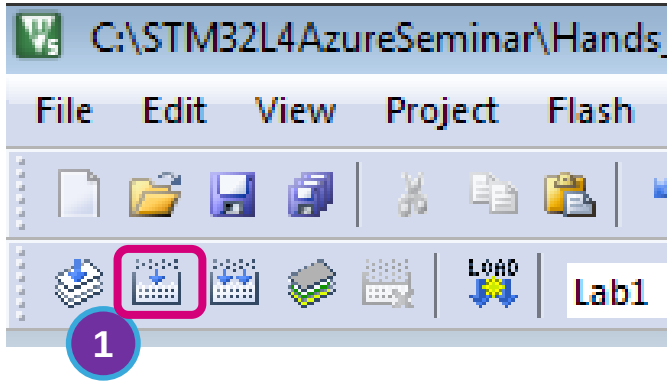
```
96    /* Infinite loop */
97    /* USER CODE BEGIN WHILE */
98    while (1)
99    {
100     HAL_Delay(100);
101     HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
102
103   /* USER CODE END WHILE */
104
105   /* USER CODE BEGIN 3 */
106
107   }
108   /* USER CODE END 3 */
109
110 }
```

Add the following code inside the **while(1)** loop (line 100):

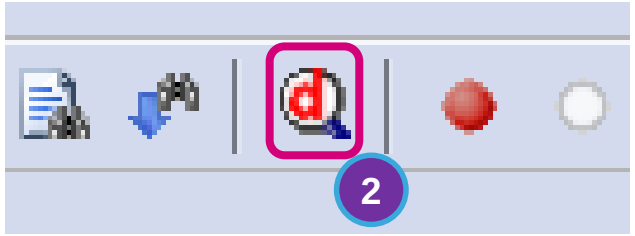
```
HAL_Delay(100);
HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
```

Load and Run

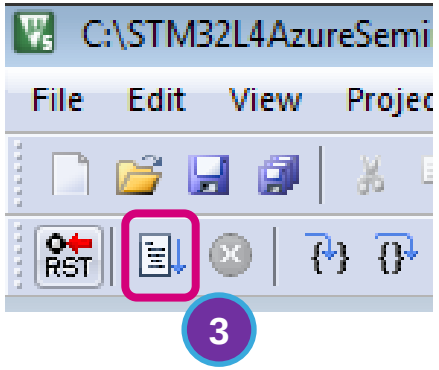
1. Click on the Build button or choose Project > Build Target



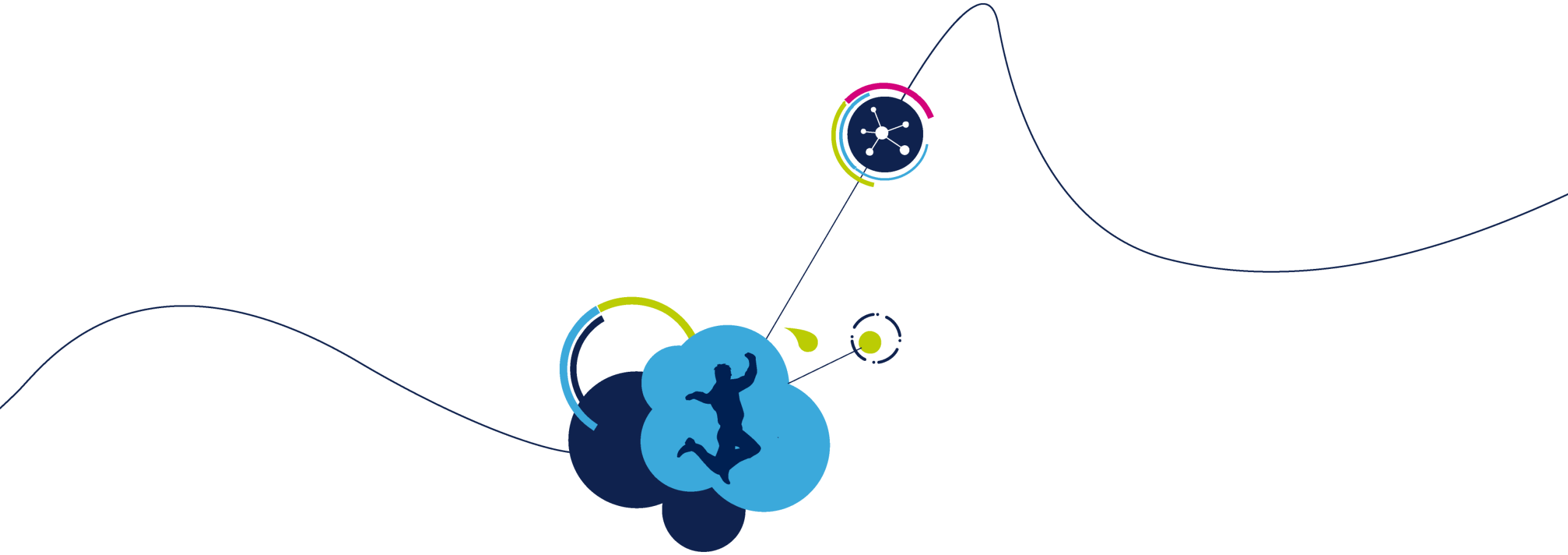
2. Click the Start/Stop Debug Session button



3. Click the Run button



The board LED should now start blinking!



Microsoft Azure IoT Overview

life.augmented

What is Microsoft Azure IoT?

- Microsoft Azure Internet of Things (IoT) services enable secure, bidirectional communication between IoT devices and the cloud.

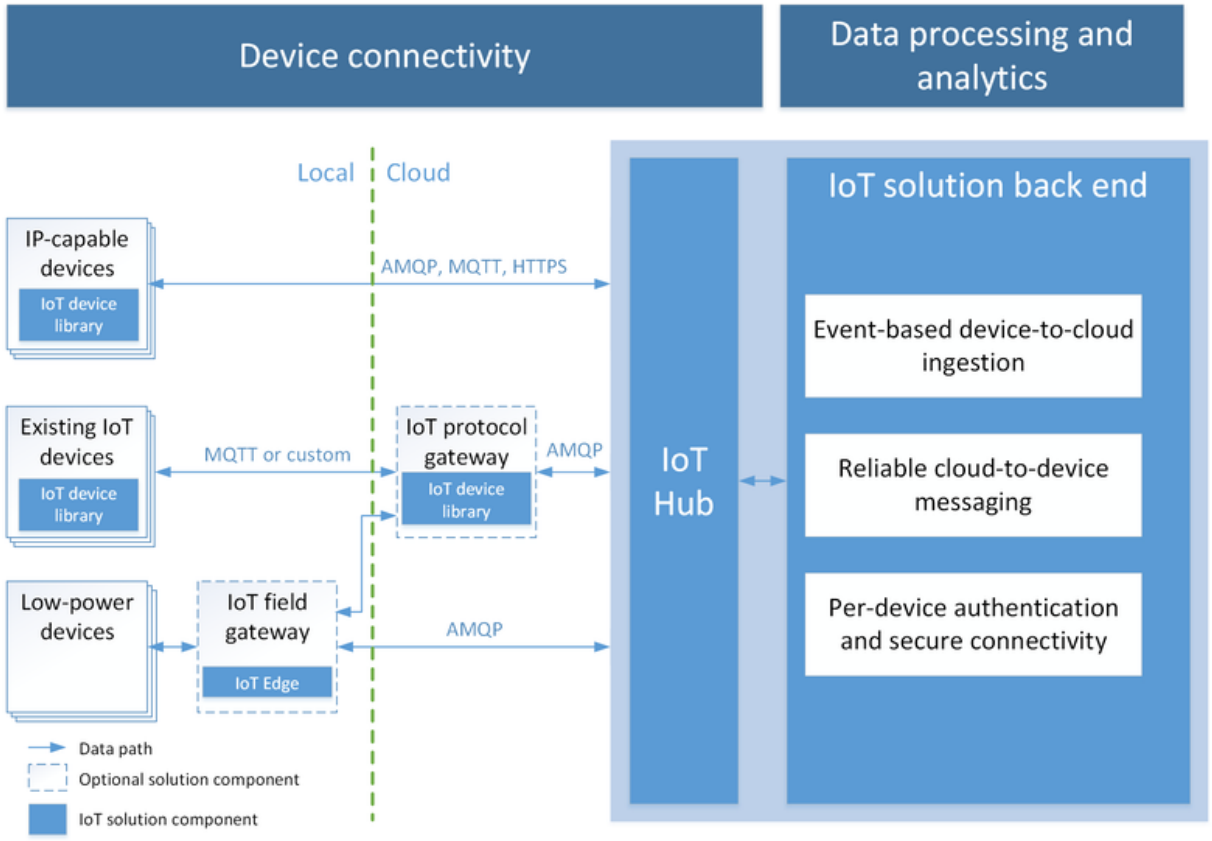


Image Source: Microsoft

What Is JSON

- JSON (JavaScript Object Notation) is an open standard lightweight data-interchange format. As a text document, it is easy for users to read and write, and for machines to parse and generate.

- JSON is completely language independent, but uses conventions that are familiar to C-family programmers including C, C++, C#, Java, JavaScript, Perl, Python and many others.

```
{
  "state": {
    "desired": {
        "LED_value": "On"
    }
  }
}
```

What is MQTT

- **MQTT** stands for **MQ Telemetry Transport**. It is an extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks.

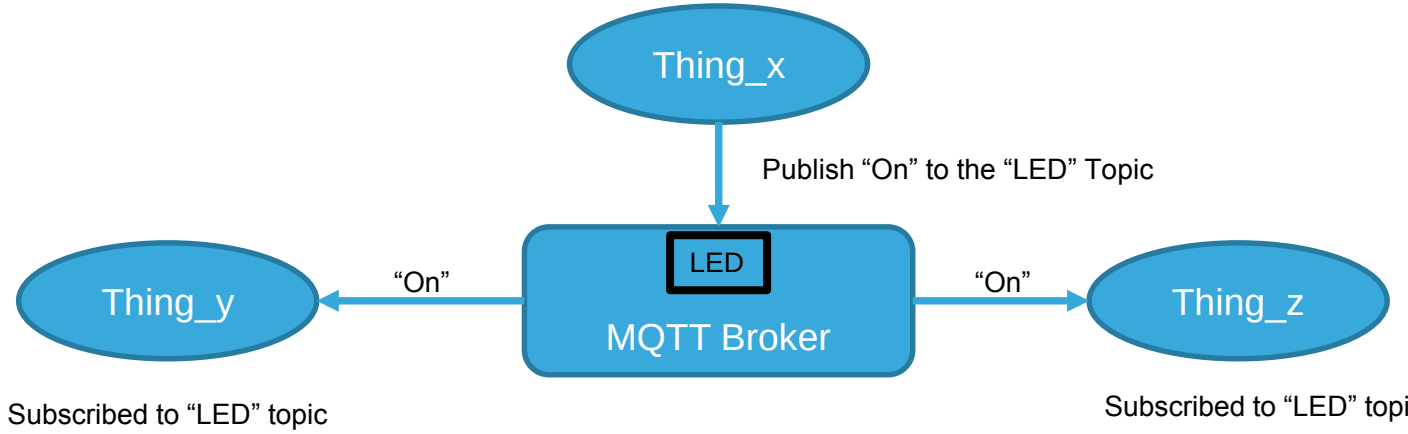
- Designers use **MQTT** to minimize network bandwidth and device resource requirements, while ensuring reliability and some degree of delivery assurance. These principles also turn out to make the protocol ideal in the emerging “machine-to-machine” (M2M) or “Internet of Things” world of connected devices, and for mobile applications where bandwidth and battery power are at a premium.

Source: http://mqtt.org/

How MQTT Works

- MQTT uses a client/server model where every IoT device is a client and is connected to a server, called an MQTT broker (example: Azure IoT Hub).

- The clients send messages to an address, called a topic. The MQTT broker will forward that message to all the clients subscribed to that topic.



Azure IoT Device Twin

- A Device Twin is a JSON document that is used to store and retrieve current state information for a IoT node.
- For each new device, the IoT Hub creates a Device Twin and uses it to maintain a persistent state of the device regardless of whether the device is connected to the Internet or not.
- The Device Twin JSON document includes:
  - Tags
  - Desired Properties
  - Reported Properties

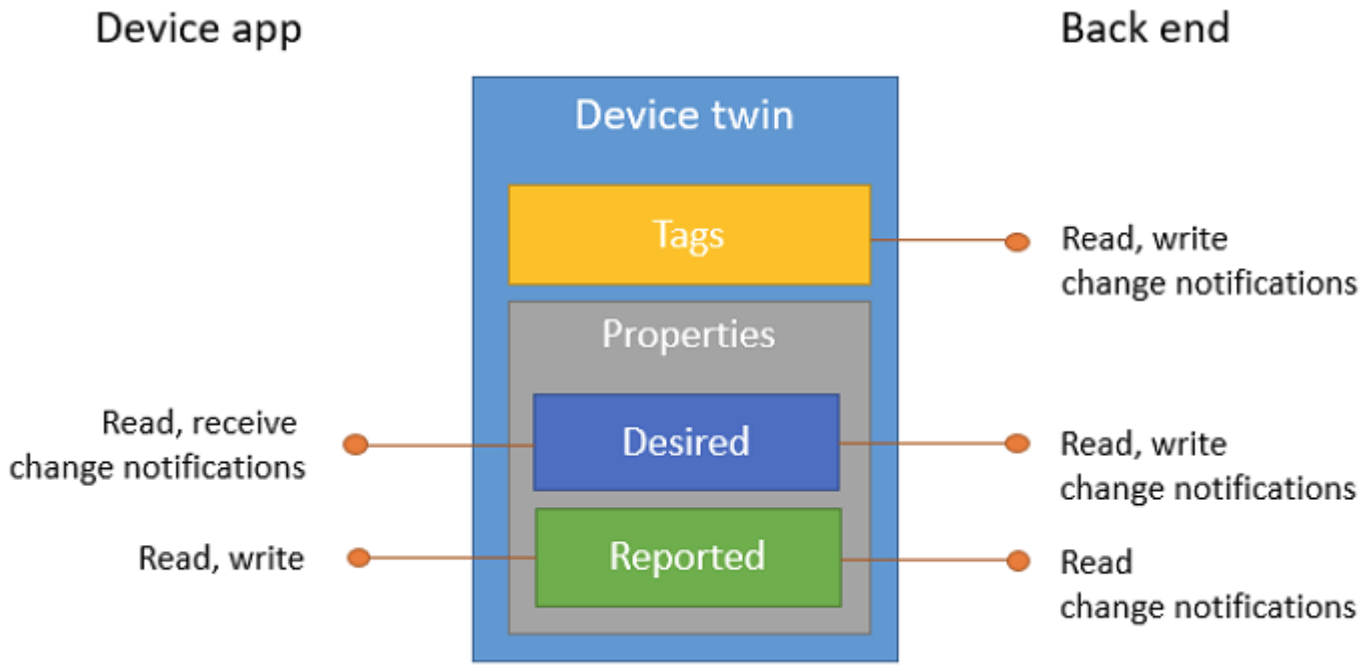


Image Source: Microsoft

Device Management

- IoT Hub enabled a set of device management patterns that can be adapted or extended to fit your needs
  - Reboot
  - Factory reset
  - Configuration
  - Firmware update
  - Report progress and status

Configuration

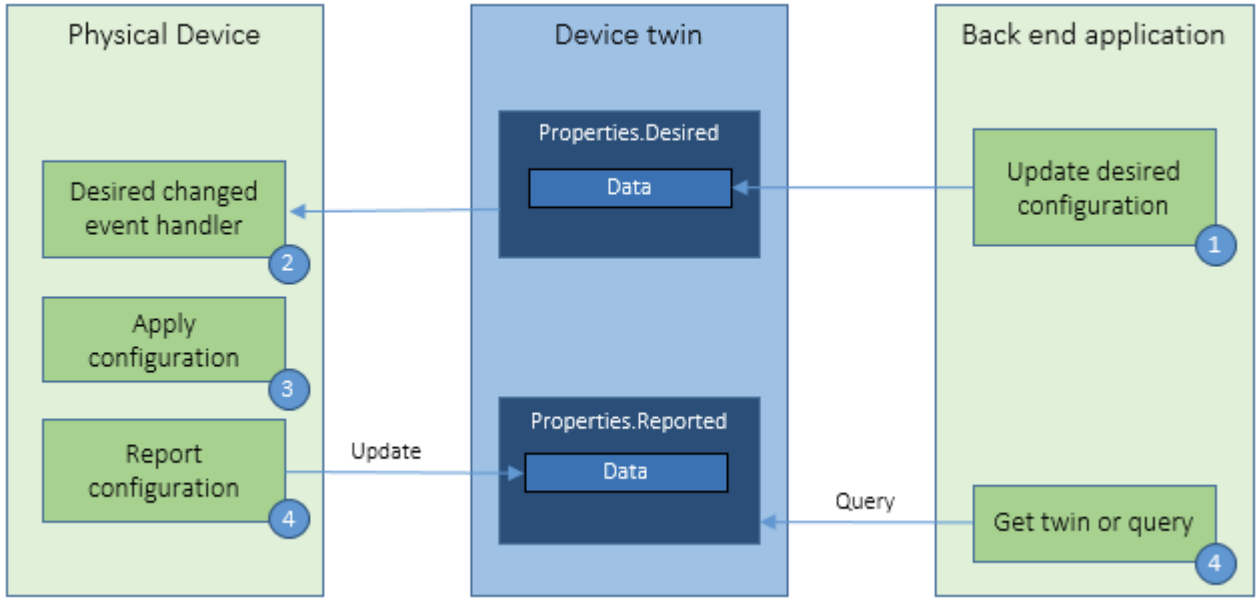
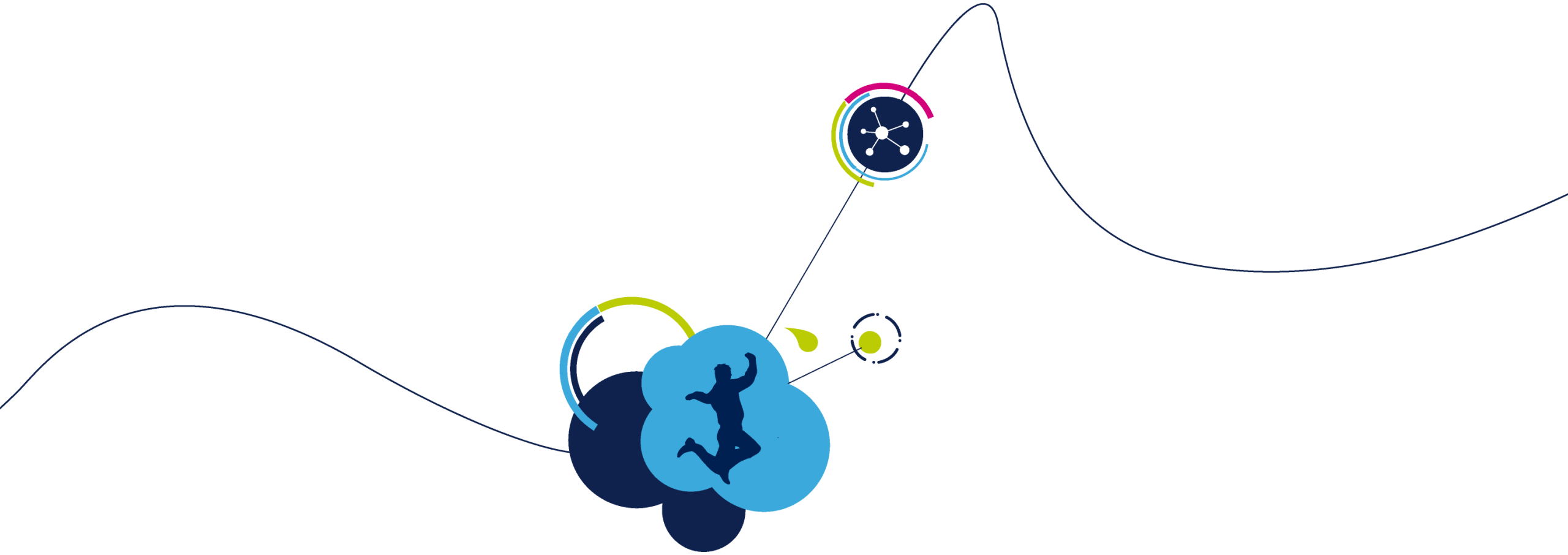


Image Source: Microsoft

The pattern shown here is for Configuration: the back-end app uses the desired properties to configure software running on the device. The device uses reported properties to update the configuration status of the device.

Azure IoT Security

- IoT Hub uses security tokens to authenticate devices and services
- Devices and services use a shared access or symmetric key to sign and send security tokens to the IoT Hub to get access
- IoT Hub also allows using X.509 certificates to authenticate a device



Lab 2: Getting Started with Microsoft Azure IoT in 5 minutes!

life.augmented

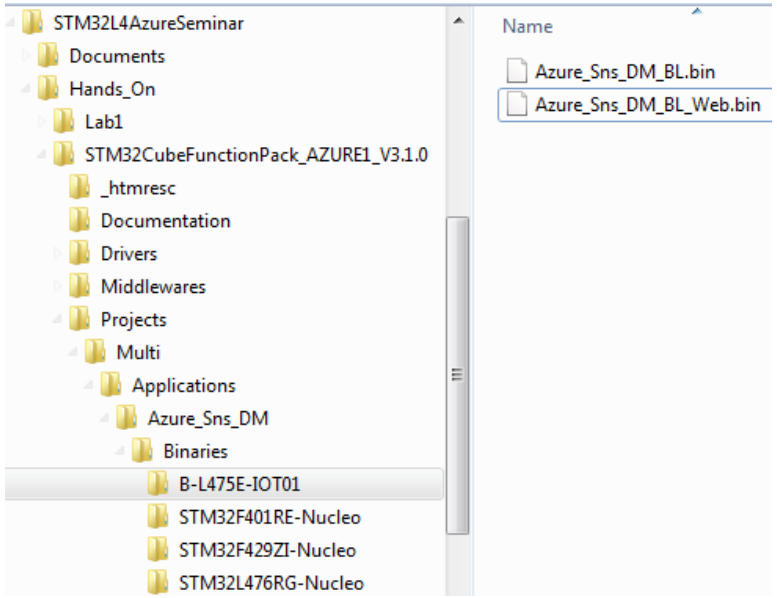
Lab Overview

- In this lab, we will:
  - Program the STM32L4 Discovery Kit IoT node with precompiled binaries
  - Input WiFi credentials through a terminal program
  - Establish connection to Azure IoT
  - View sensor data from your node on a web application
  - Explore various Azure features available through the web application

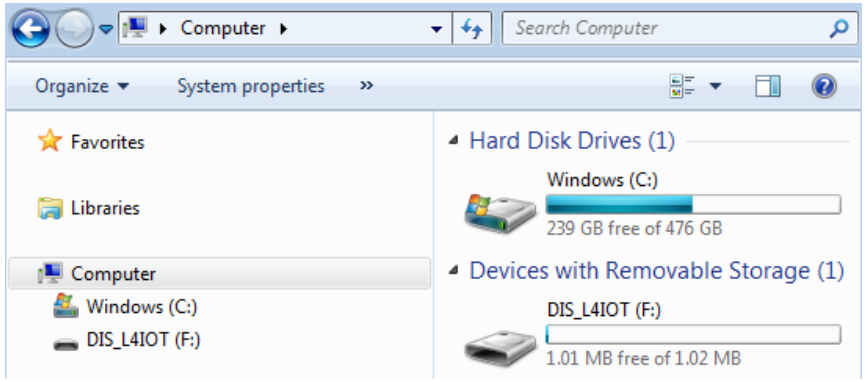
Program your board

- In Windows Explorer, navigate to the following folder:

C:\STM32L4AzureSeminar\Hands_On\STM32CubeFunctionPack_AZURE1_V3.1.0\Projects\Multi\Applications\Azure_Sns_DM\Binaries\B-L475E-IOT01\



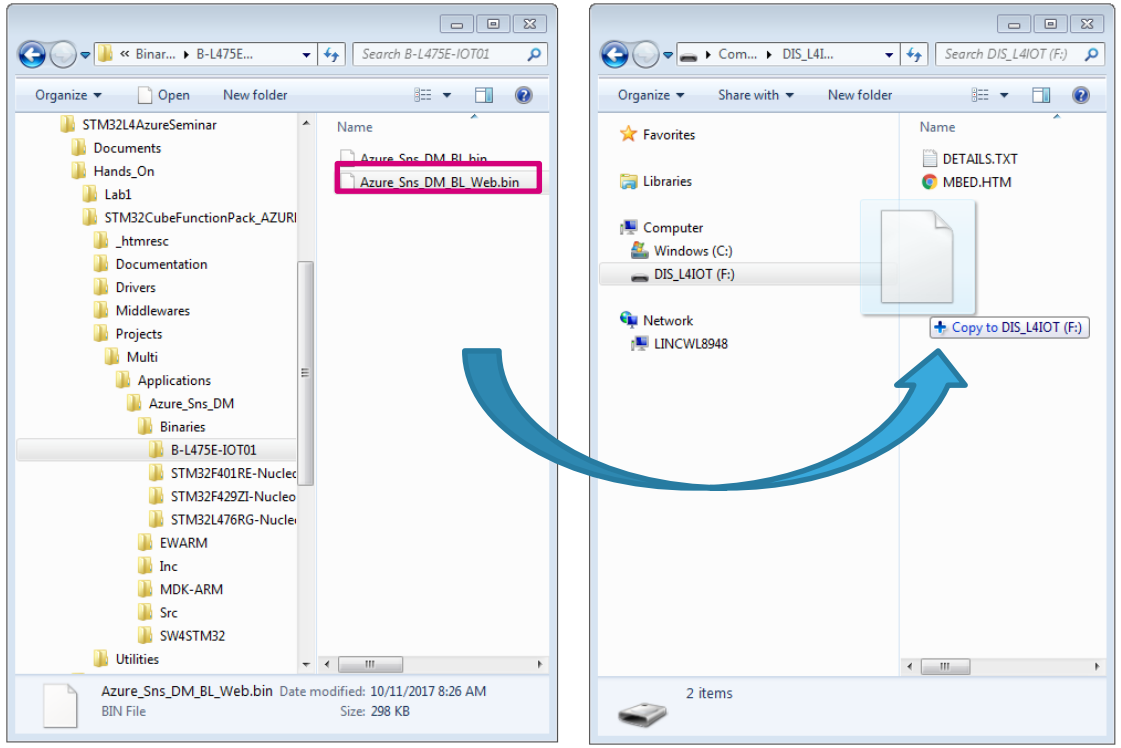
- Open another Windows Explorer instance, and Click “Computer” and open the “DIS_L4IOT” drive; this could be assigned some drive letter such as D: or E: on your PC.



life.augmented

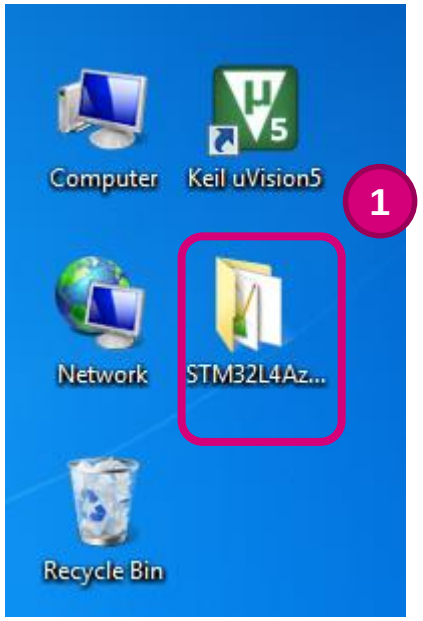
Program your board

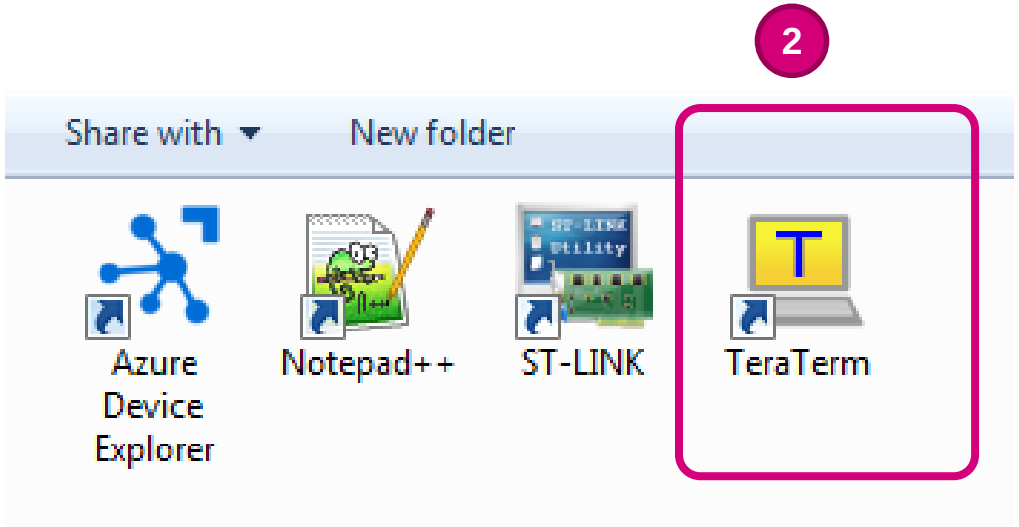
- We will use simple drag-and-drop programming that is enabled by the enhanced ST-LINK/V2-1 debugger built into the STM32L4 Discovery Kit
- Drag and drop the "**Azure_Sns_DM_BL_Web.bin**" file to the Discovery kit "DIS_L4IOT" drive
- Programming is complete when the LD6 LED stops flashing and the "DIS_L4IOT" drive disappears and reappears.



Note: If you prefer, you can also copy and paste the file to the target drive

Open and Configure Serial Terminal



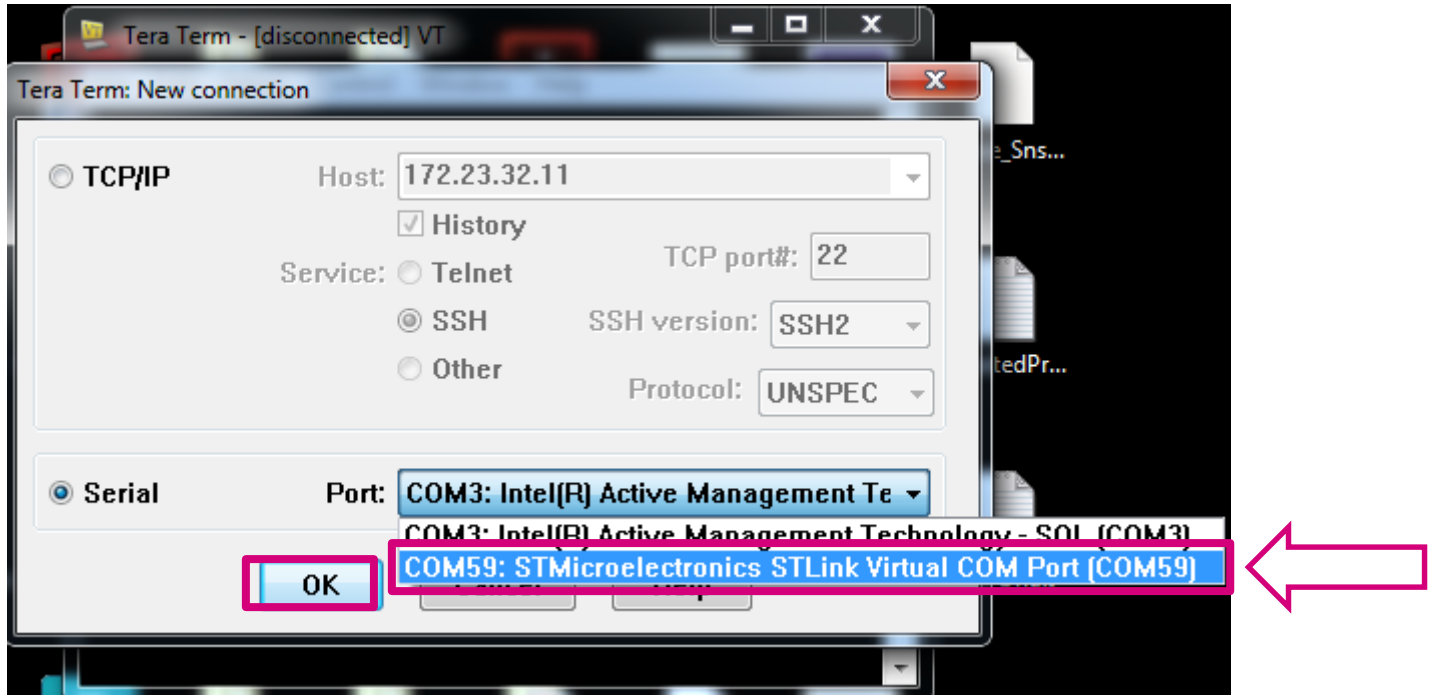


1. Open the STM32L4AzureSeminar folder on your Desktop.

2. Double-click on the **TeraTerm** shortcut

Open and Configure Serial Terminal (Cont'd)

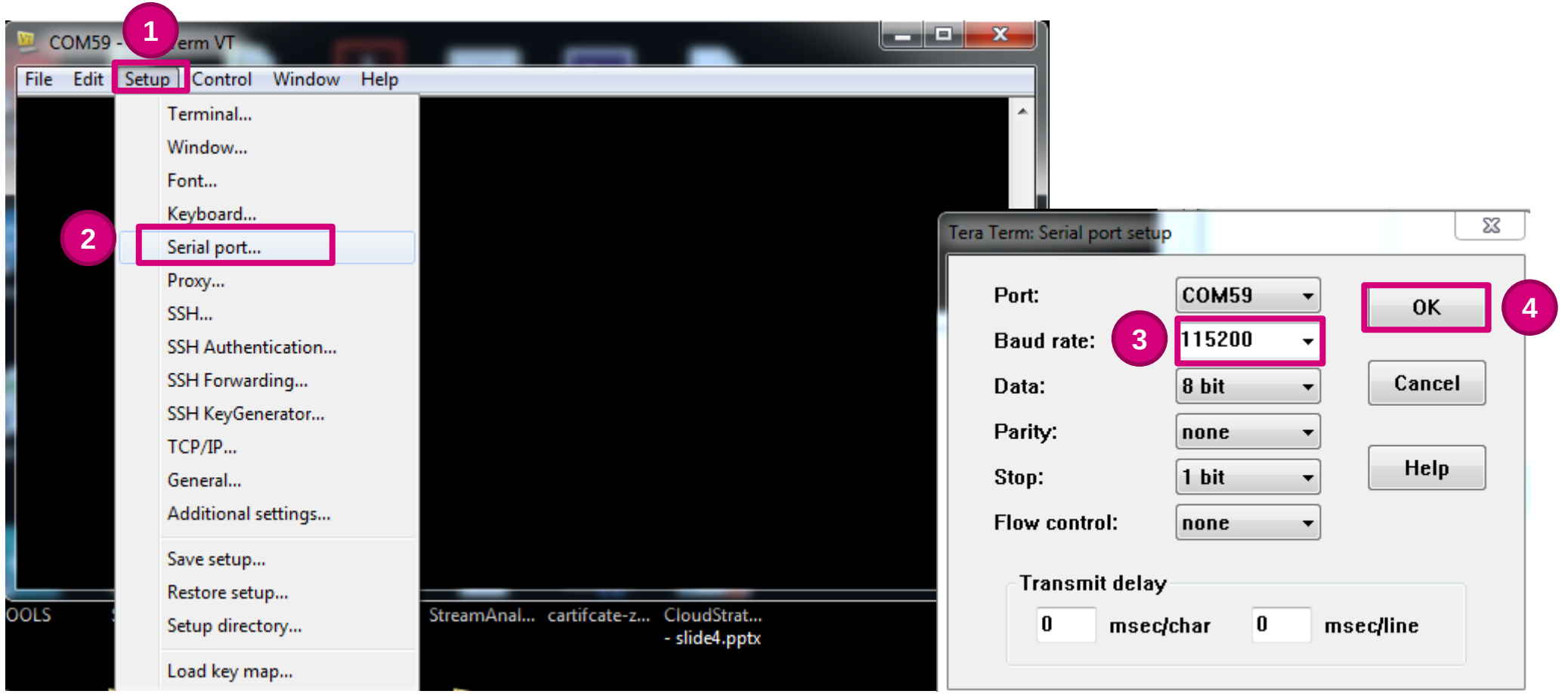
1. Select the STMicroelectronics STLink Virtual COM Port
2. Click OK



life.augmented

Open and Configure Serial Terminal

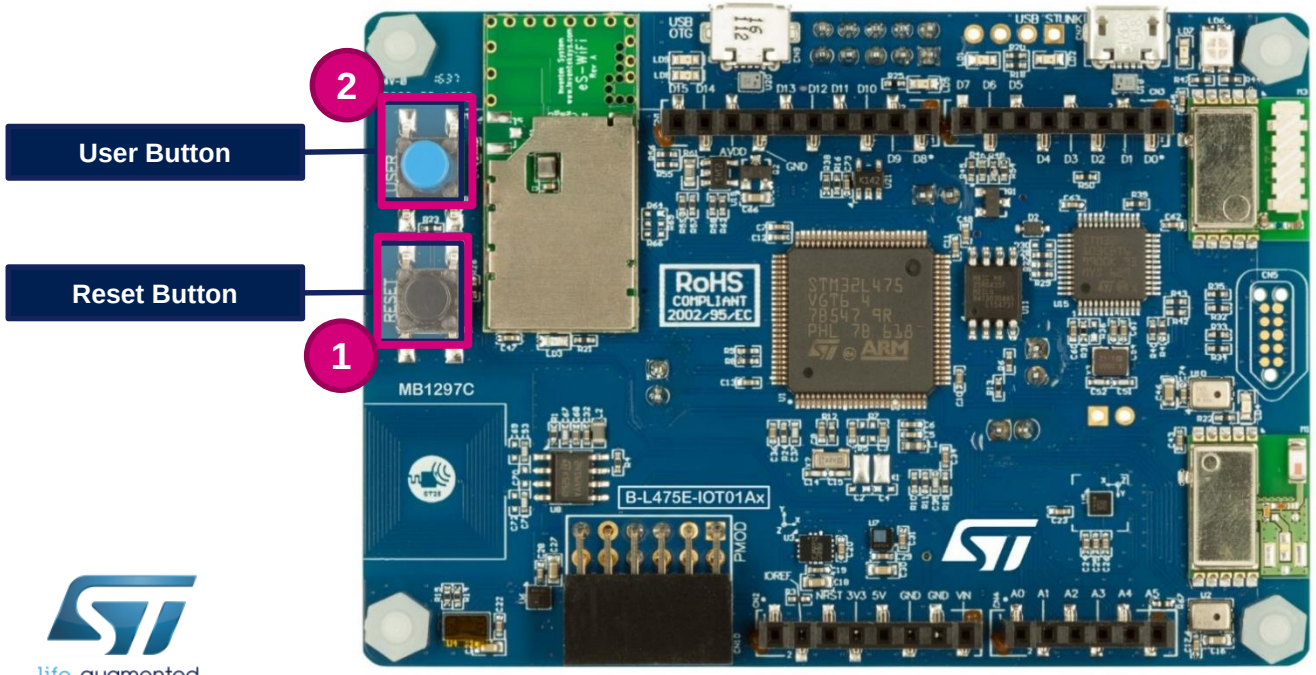
- Set baud rate speed in serial terminal to 115200 (**Setup → Serial port**)

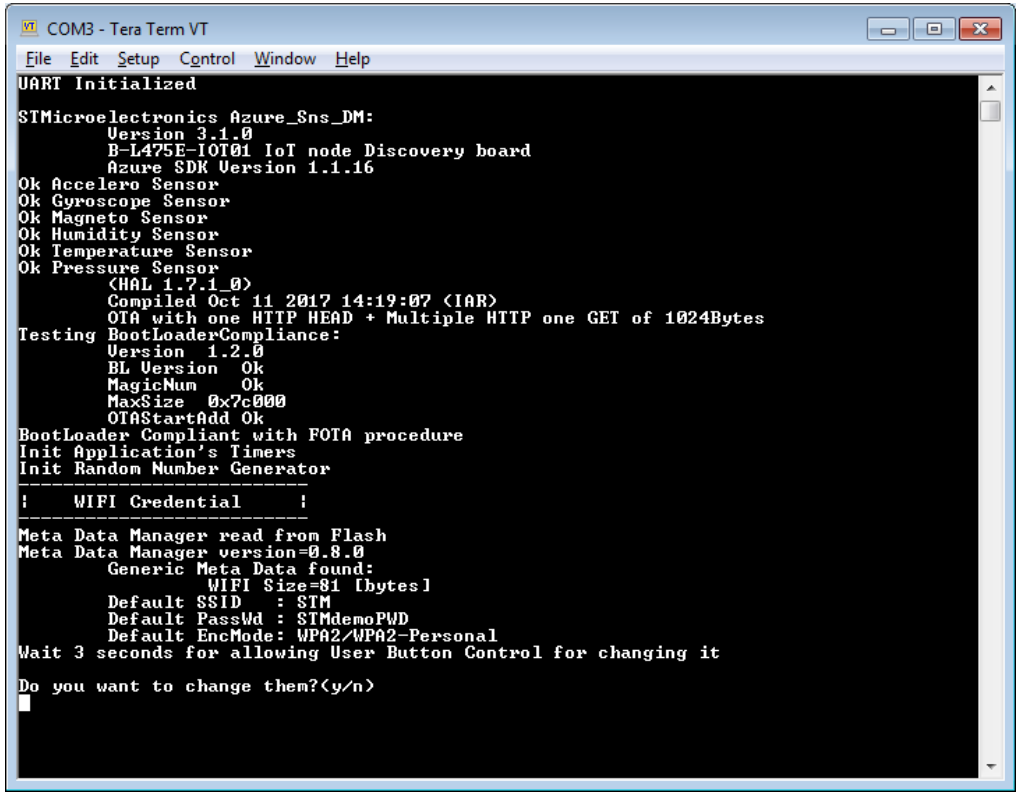


Note: If you see any Japanese characters, go to Setup > General and change language to “UTF-8”

Reset the Board

- Press the **black** Reset button
- Then, within 3 seconds, press the **blue** User button





life.augmented

Change the default WiFi Credentials

- Type your responses to questions as follows:
- Press Enter after each response

1. Do you want to change them?(y/n)?

   y

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

   n

3. Enter the SSID:

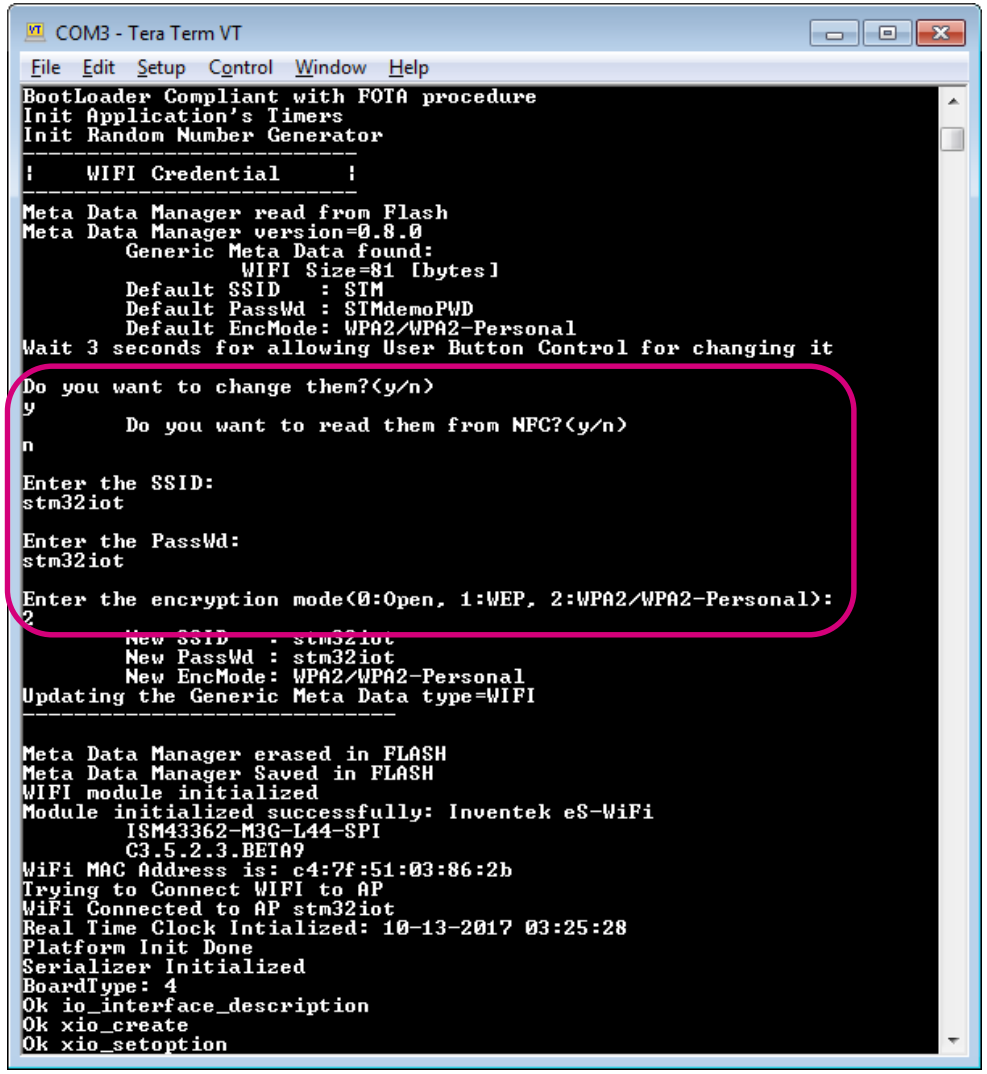
   stm32iot

4. Enter the PassWd:

   stm32iot

5. Enter the encryption mode (0:open, 1:WEP, 2:WPA/WPA2-Personal):

   2



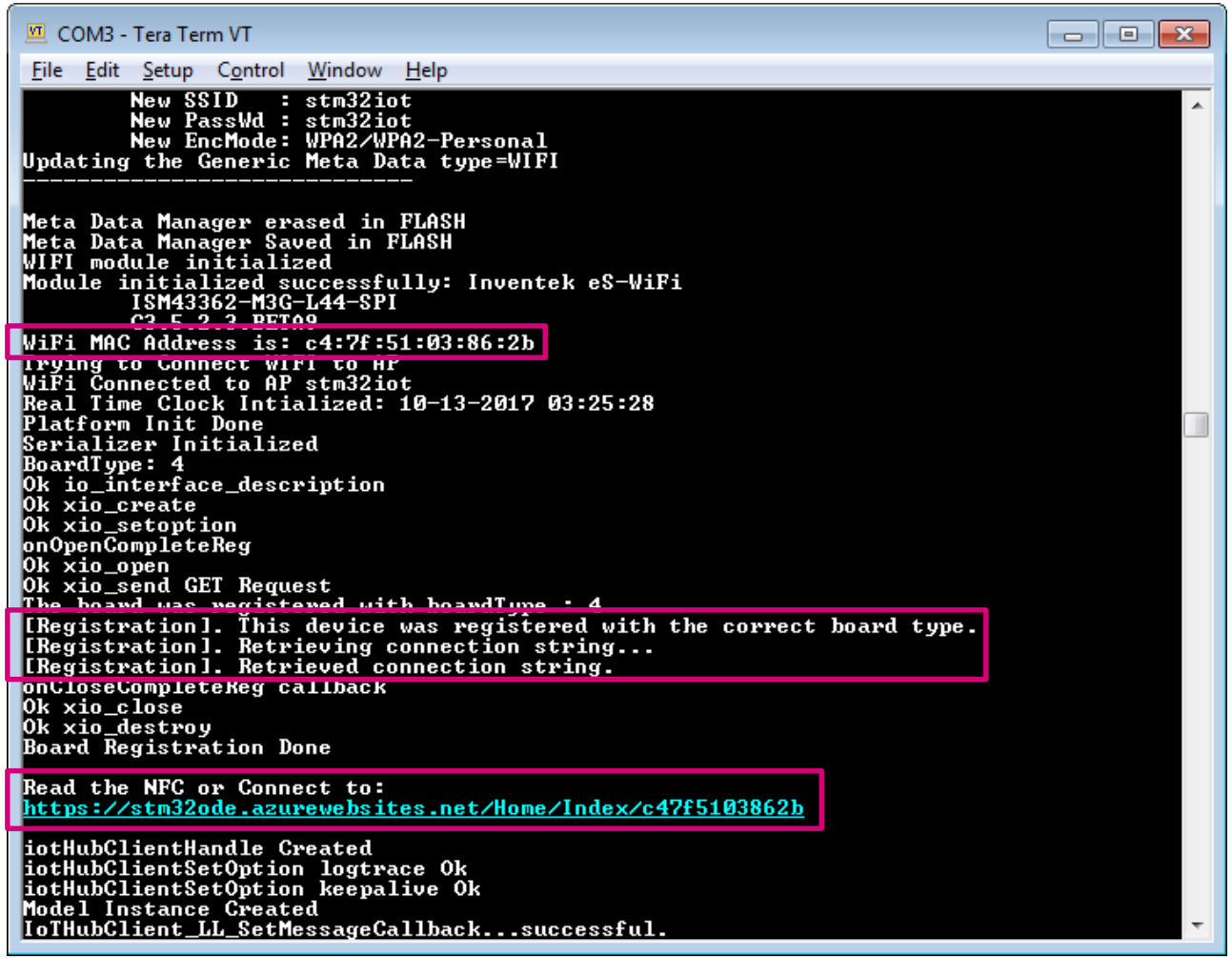
Note: You cannot use backspace or delete if you make a typo; in that case, press reset, then blue button and start over.

life.augmented

Auto Registration to Azure IoT Hub

- The WiFi credentials are stored in MCU Flash memory for future use

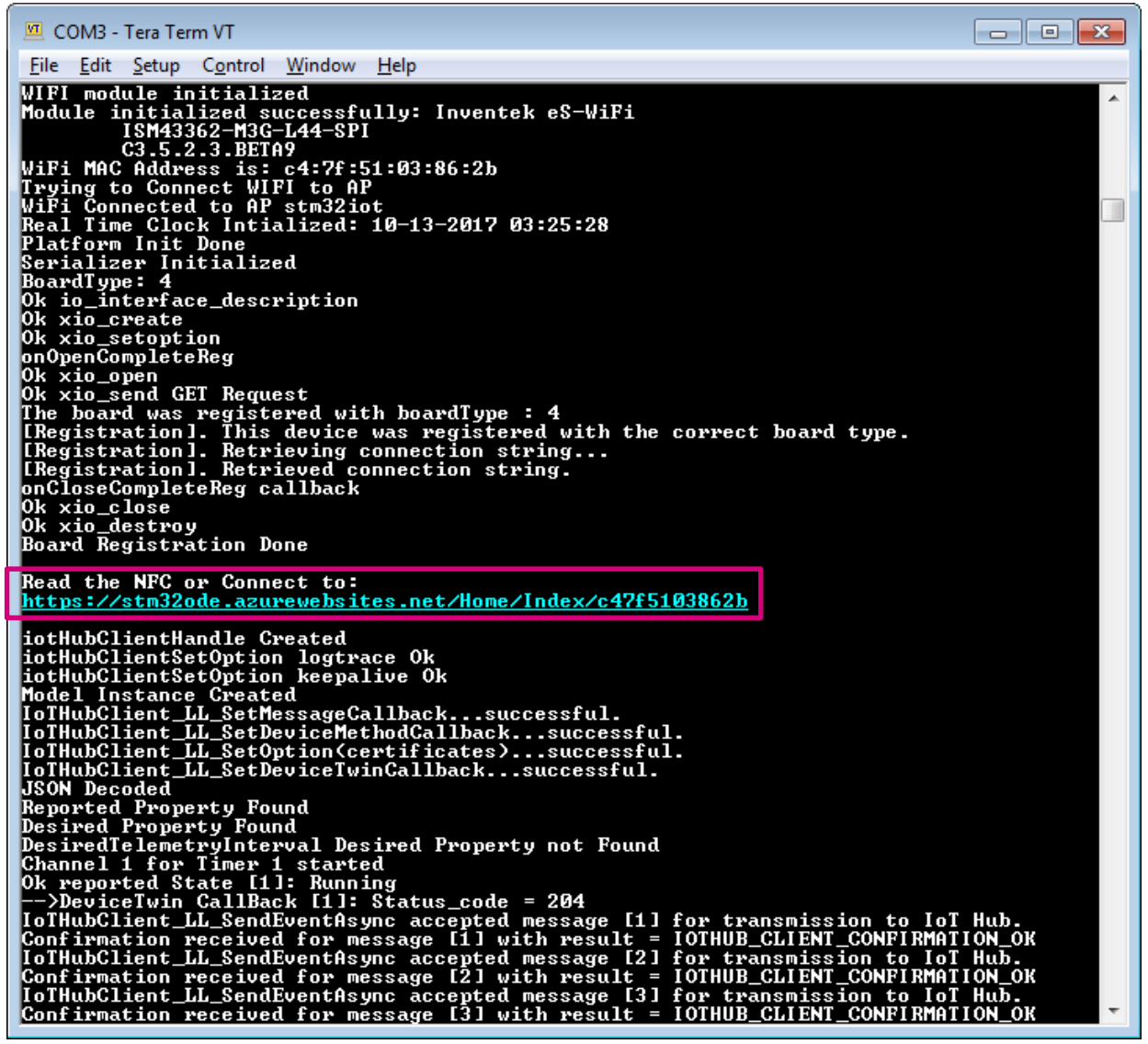
- Board is automatically registered to ST's demonstration Azure IoT Hub
  - The WiFi module's MAC address is used as the unique identifier
  - The unique URL displayed also uses the same unique identifier



life.augmented

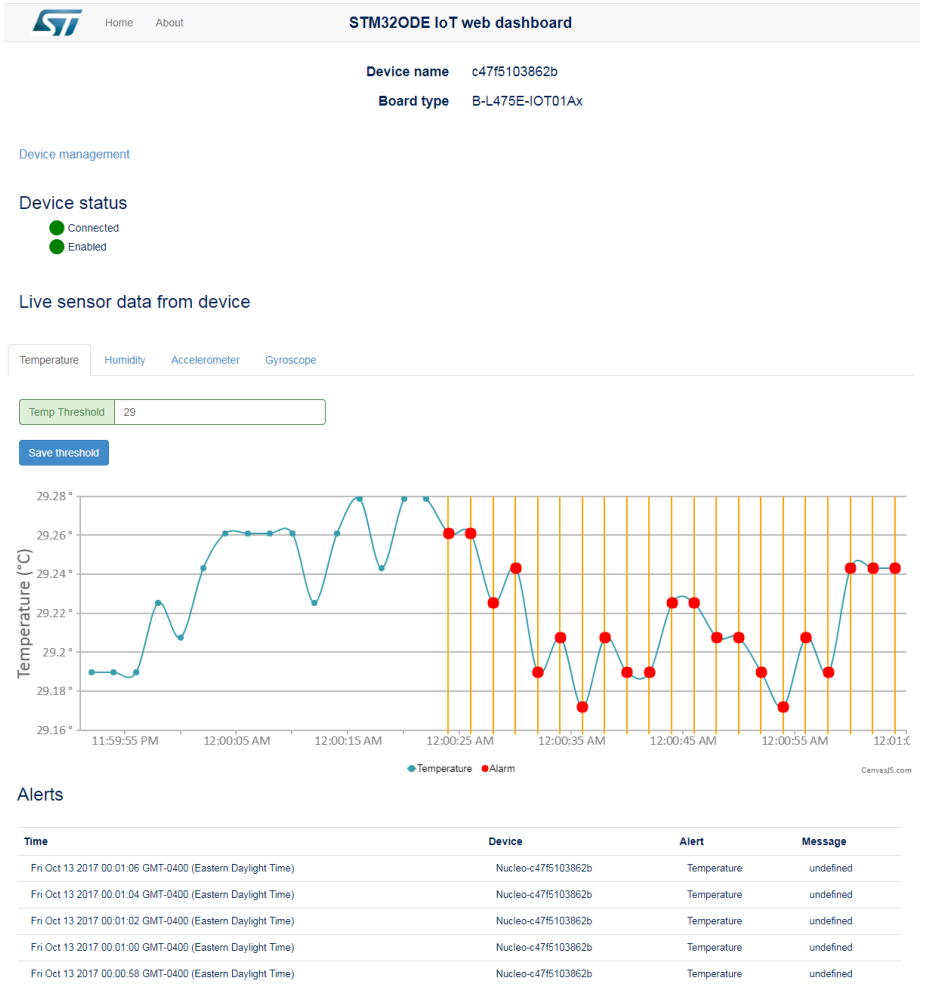
Open the Web Application

- Multiple ways to open the web app:
  - Double click on the URL in the TeraTerm terminal window to lauch your browser
  - Use your Android phone to touch the NFC on the board to launch the URL in your browser



STM32ODE IoT Web Dashboard

- The STM32ODE Web Dashboard is a web application hosted on Azure cloud
  - Connects to IoT hub and retrieves sensor data sent from device to cloud
  - Plots data on easy to view graphs
  - Separate tabs for each sensor
  - Allows setting alert thresholds and highlights values exceeding the threshold
  - Unique URL for each device that is registered to the IoT Hub
  - Displays device status
  - Allows device management



life.augmented

Visualize Sensor Data

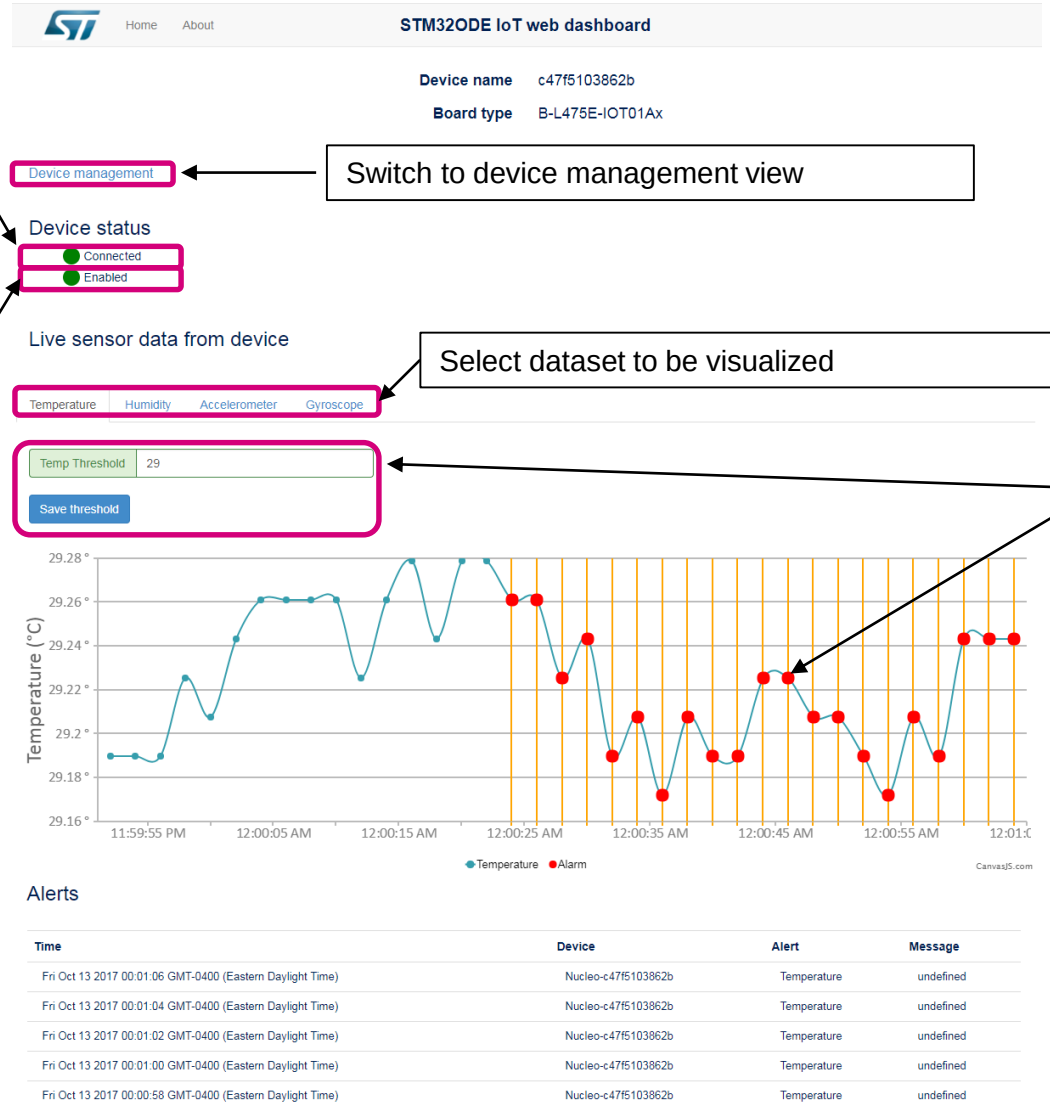
Connected: Device is connected to IoT Hub now

Disconnected:
Device is not connected to IoT Hub now

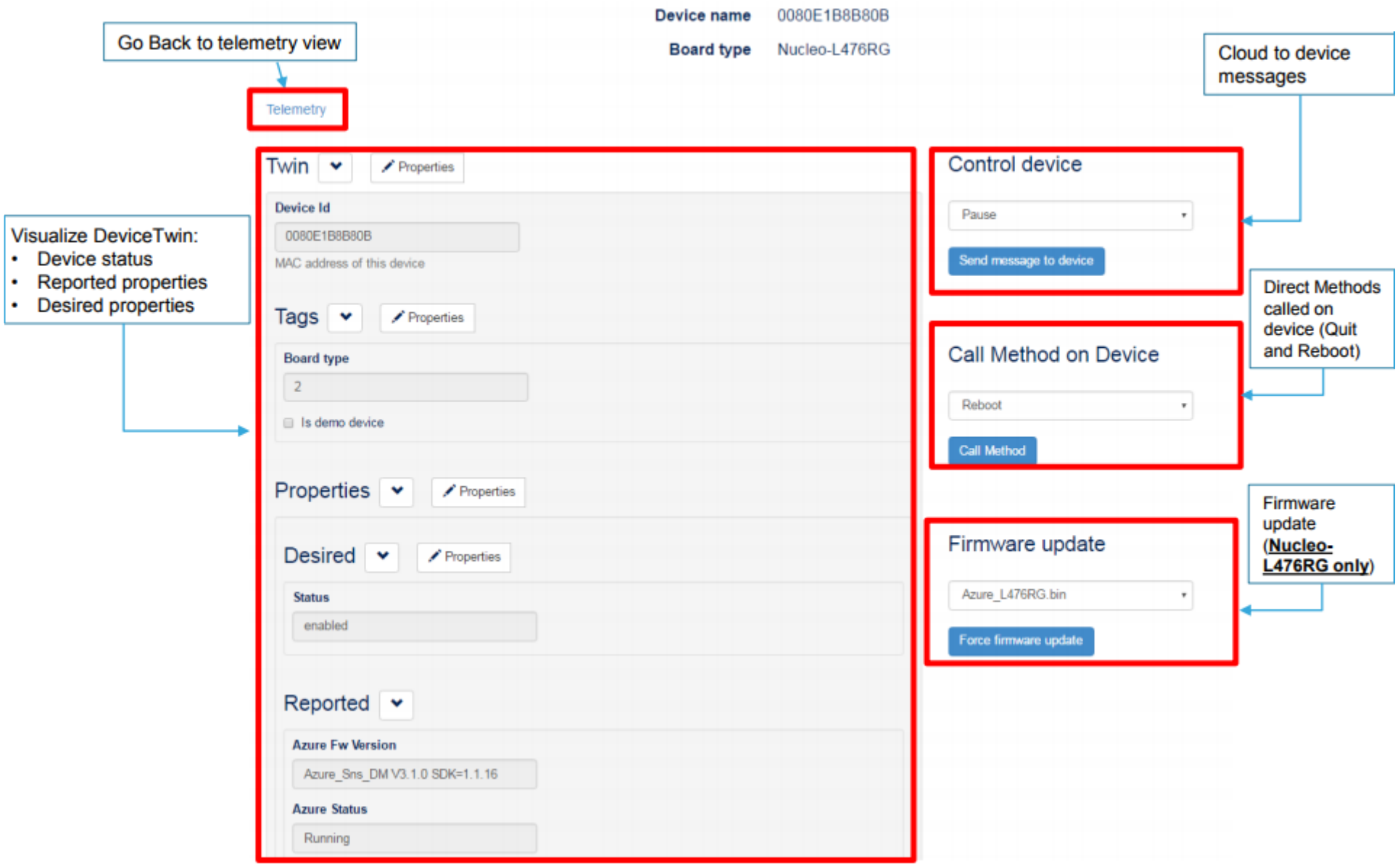
Enabled:
Device can send data if connected

Warning:
Device close to reaching maximum allowed messages per day

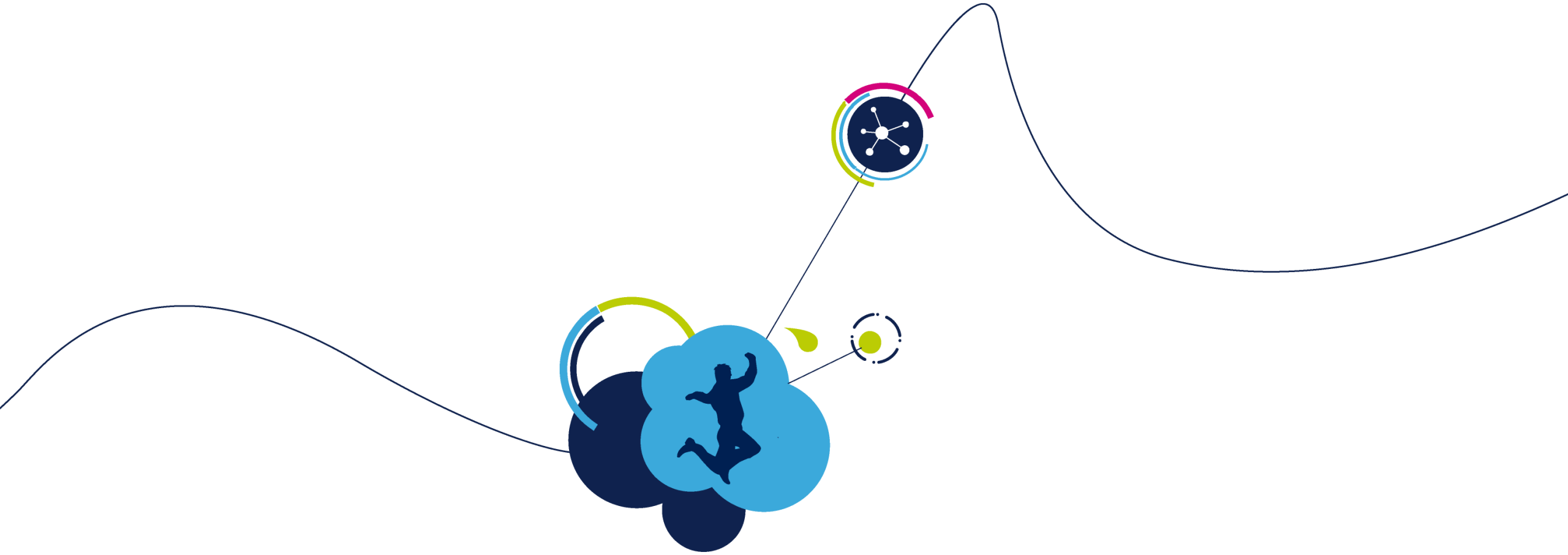
Disabled:
Device has reached maximum allocated data per day; it will be re-enabled next day



Explore Device Management Features



life.augmented



FP-CLD-AZURE1 Function Pack Overview

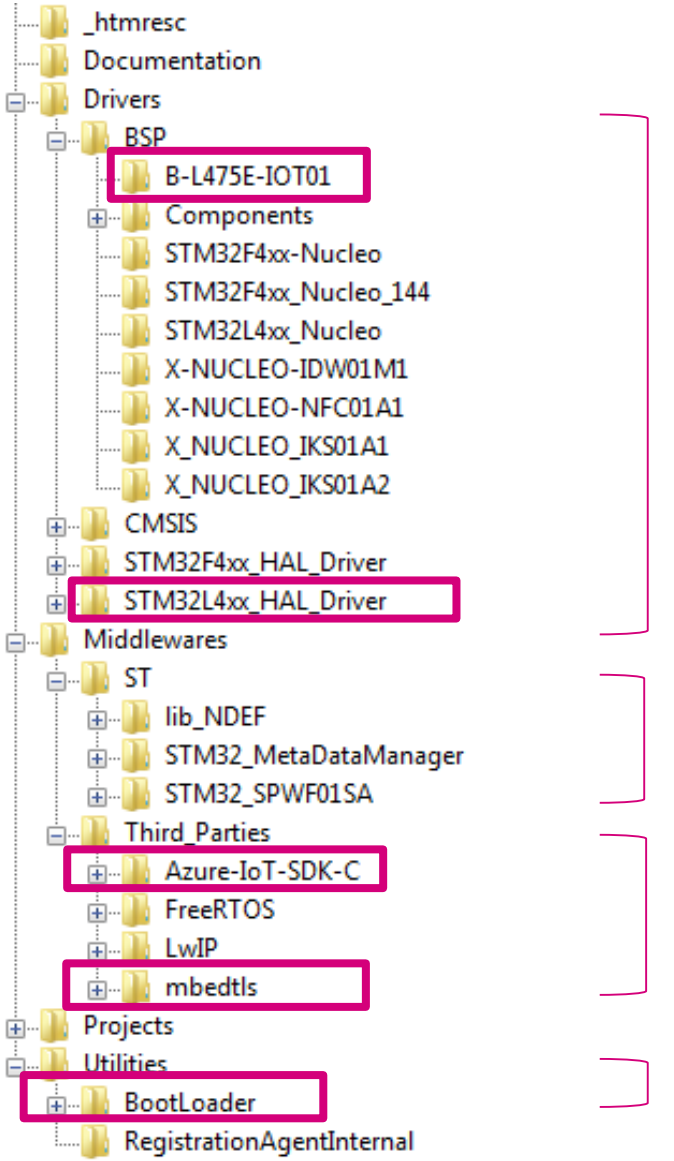
life.augmented

FP-CLD-AZURE1: Features

- Integrated Microsoft Azure IoT device Software Development Kit. Includes sample application for:
  - Connection to IoT Hub and transmission of Device-to-Cloud messages based on MQTT protocol
  - Implementation of a Device Management solution based on
    - Cloud-to-Device messages
    - Direct Methods
    - Device Properties
    - Firmware Update
- Secure network communication with the cloud application based on mbedTLS library
- Include middleware and drivers for Wi-Fi connectivity, NFC, inertial and environmental sensors
  - Provide software interface to access temperature and humidity sensor (HTS221), pressure sensor (LPS22HB), inertial sensors (LIS3MDL, LSM6DSL), ToF and gesture detection (VL53L0X) and to write and read the M24SR64-Y RFID/NFC tag

Latest info available at www.st.com
FP-CLD-AZURE1

FP-CLD-AZURE1: Folders Structure



Drivers and BSP for Discovery Kit IoT node

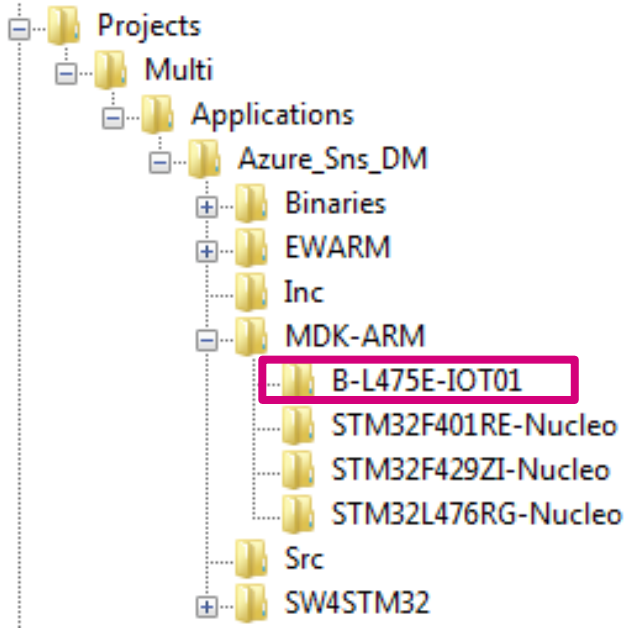
WiFi and NFC middleware

Azure IoT SDK and mbedTLS

Bootloader to handle FW update

FP-CLD-AZURE1: Folders Structure (Cont'd)

Sample application based on Azure IoT SDK for data telemetry and Device Management

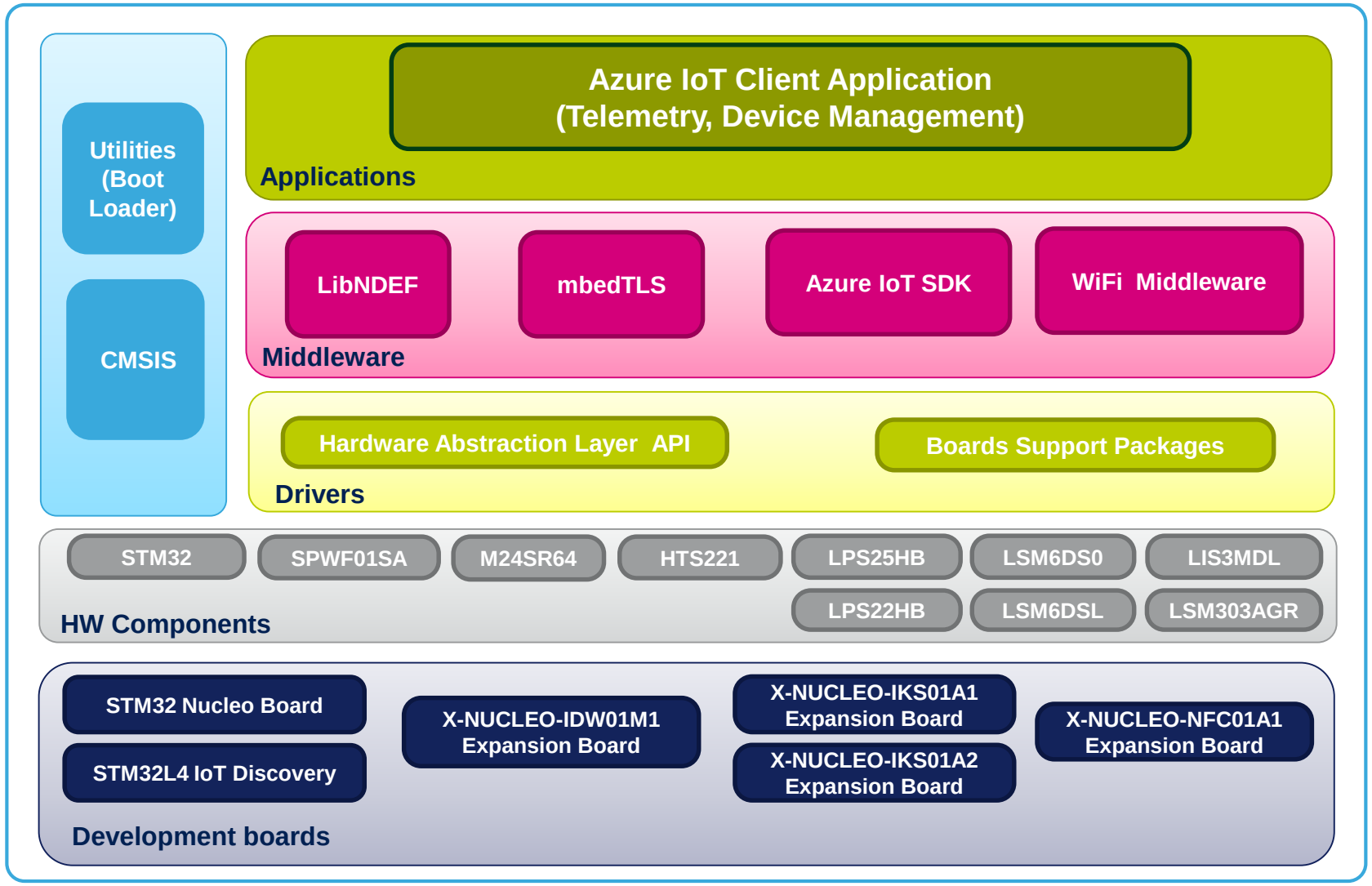


Solution files for 3 IDE:

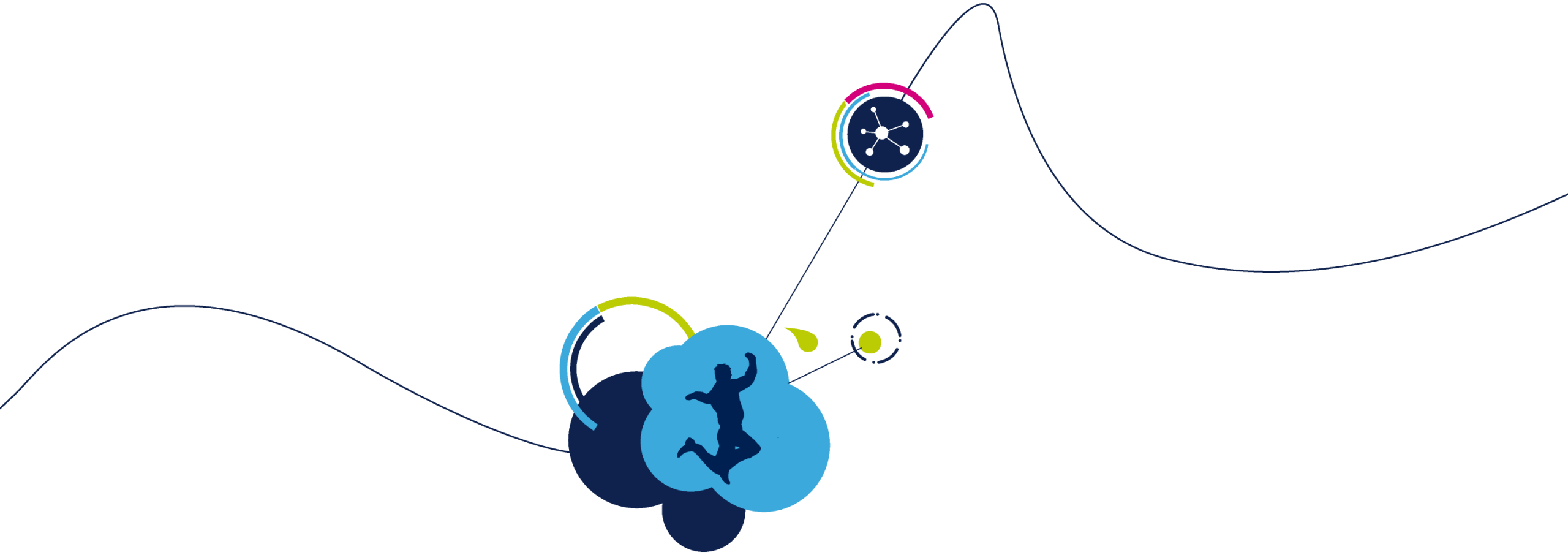
- IAR (EWARM)
- System Workbench (SW4STM32)
- ARM Keil (MDK-ARM)

C:\STM32L4AzureSeminar\Hands-on\STM32CubeFunctionPack_AZURE1_V3.1.0\Projects\Multi\Applications\Azure_Sns_DM\MDK-ARM\B-L475E-IOT01

FP-CLD-AZURE1: Software Architecture



life.augmented



Lab 3: Connect your STM32 IoT Node to Azure IoT Hub

life.augmented

Introduction

- In this lab, we will:
  - Create and login to a Free **Azure account**
  - Create an **IoT Hub**
  - Create a new **Device** node and retrieve it's **Connection String**
  - Setup your IoT node using the **sample application** provided with the Function Pack

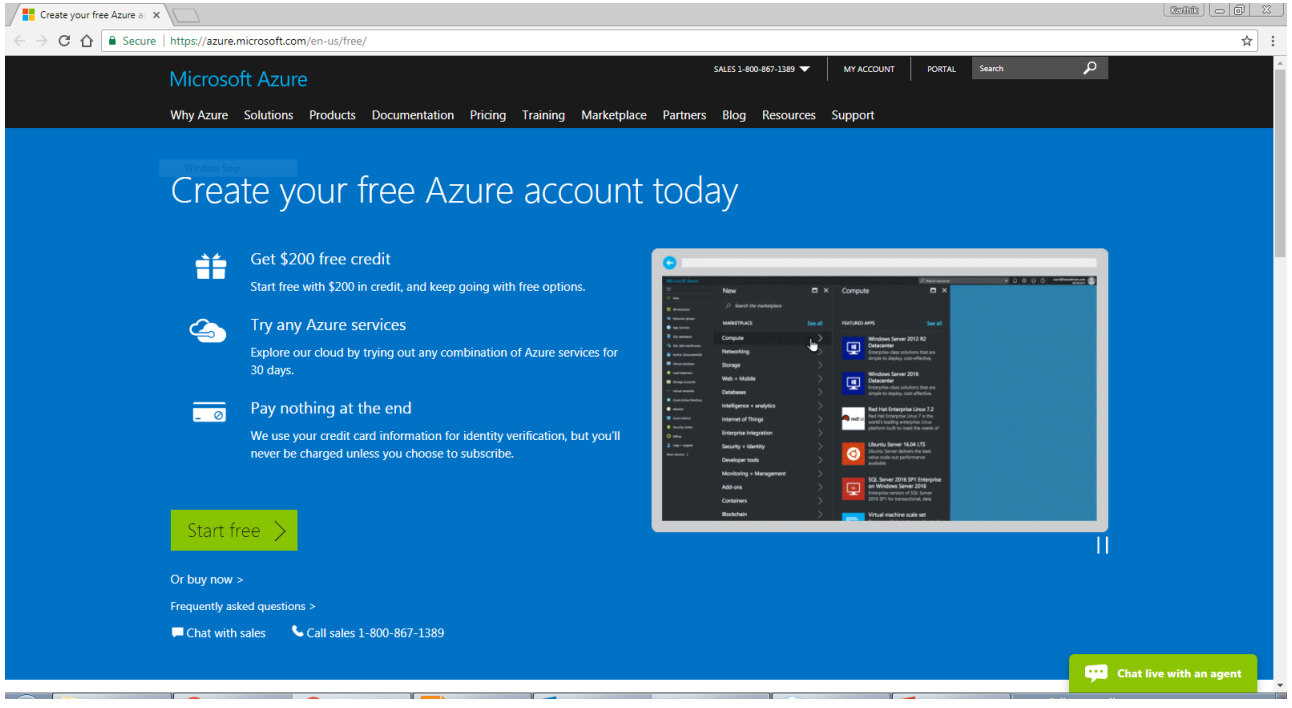
Azure Login: Seminar vs Individual Steps

- The following slides show the steps to create a new IoT Hub that you can follow later on your own

- For this seminar, we will use Azure login credentials and IoT Hub already created for this workshop by ST
  - For this workshop session, you should skip the steps to create a new Azure account and to create the new IoT Hub
  - The steps to be skipped during this workshop are marked with “We did these steps for you”.

Step 1: Create a Free Azure account

- https://azure.microsoft.com/en-us/free/
  - Need to create a Microsoft account if you don't already have one
  - Need mobile phone and credit card for identity verification
  - Free $200 credit; will not charge credit card; will not automatically renew at the end of trial period.

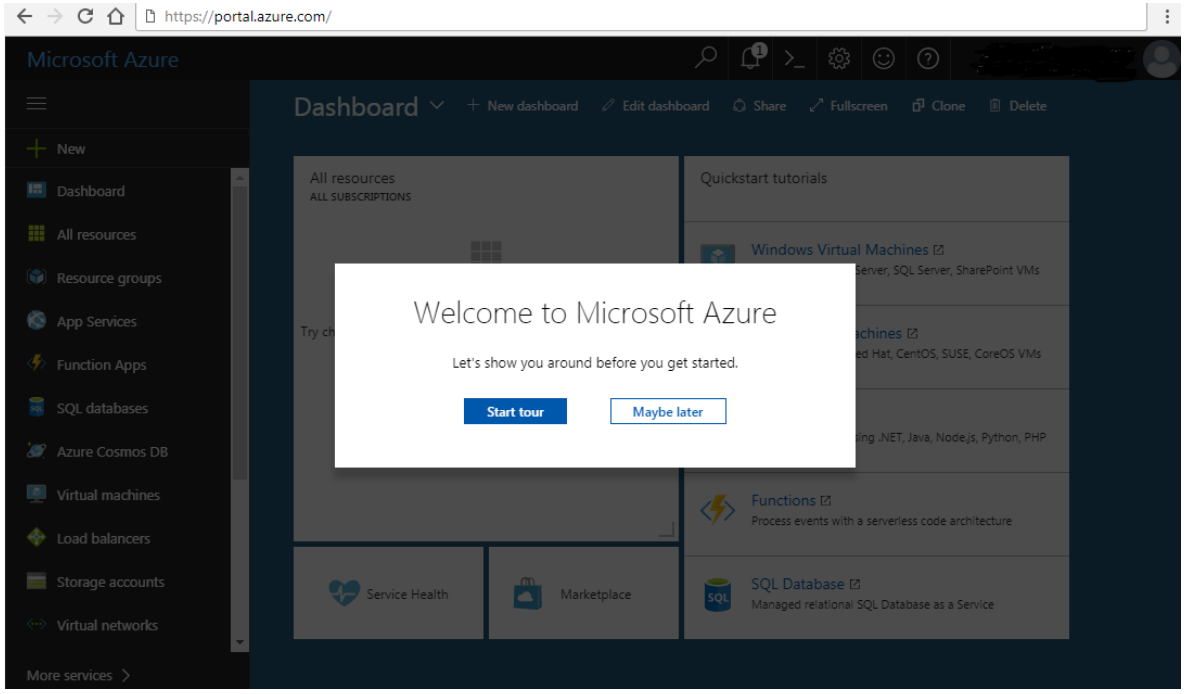


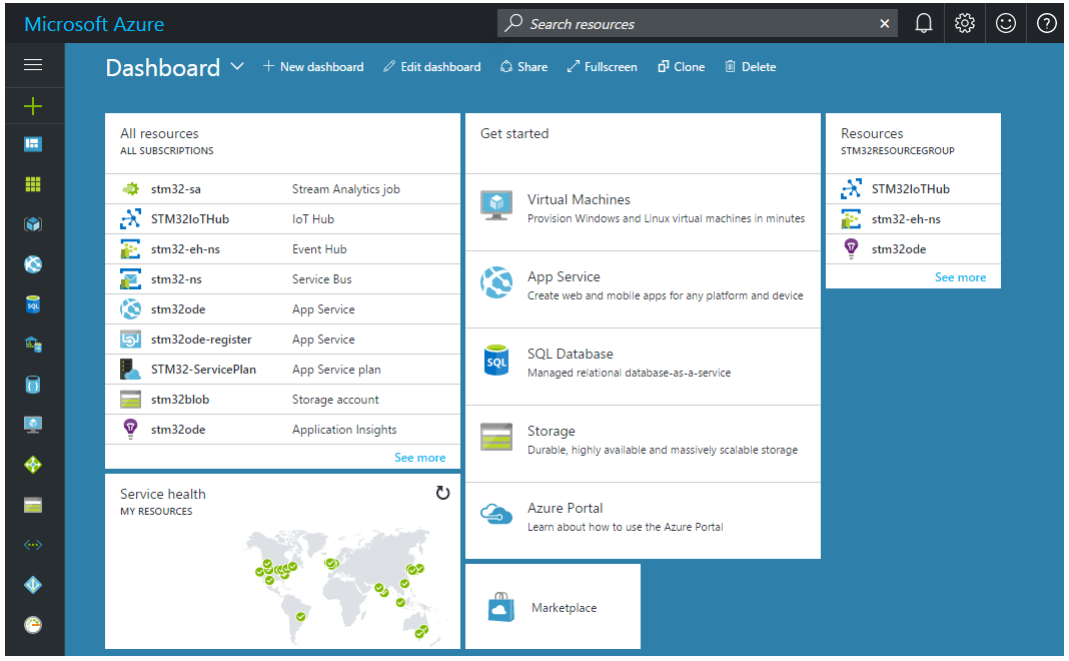
We did these steps for you

life.augmented

Step 2: Log into your Azure account

- Make sure your laptop is connected to the internet
- Go to https://portal.azure.com and sign in with the credentials below:
  - login: SeminarParticipant@iotcloudservicesst.onmicrosoft.com
  - password: STM32iot

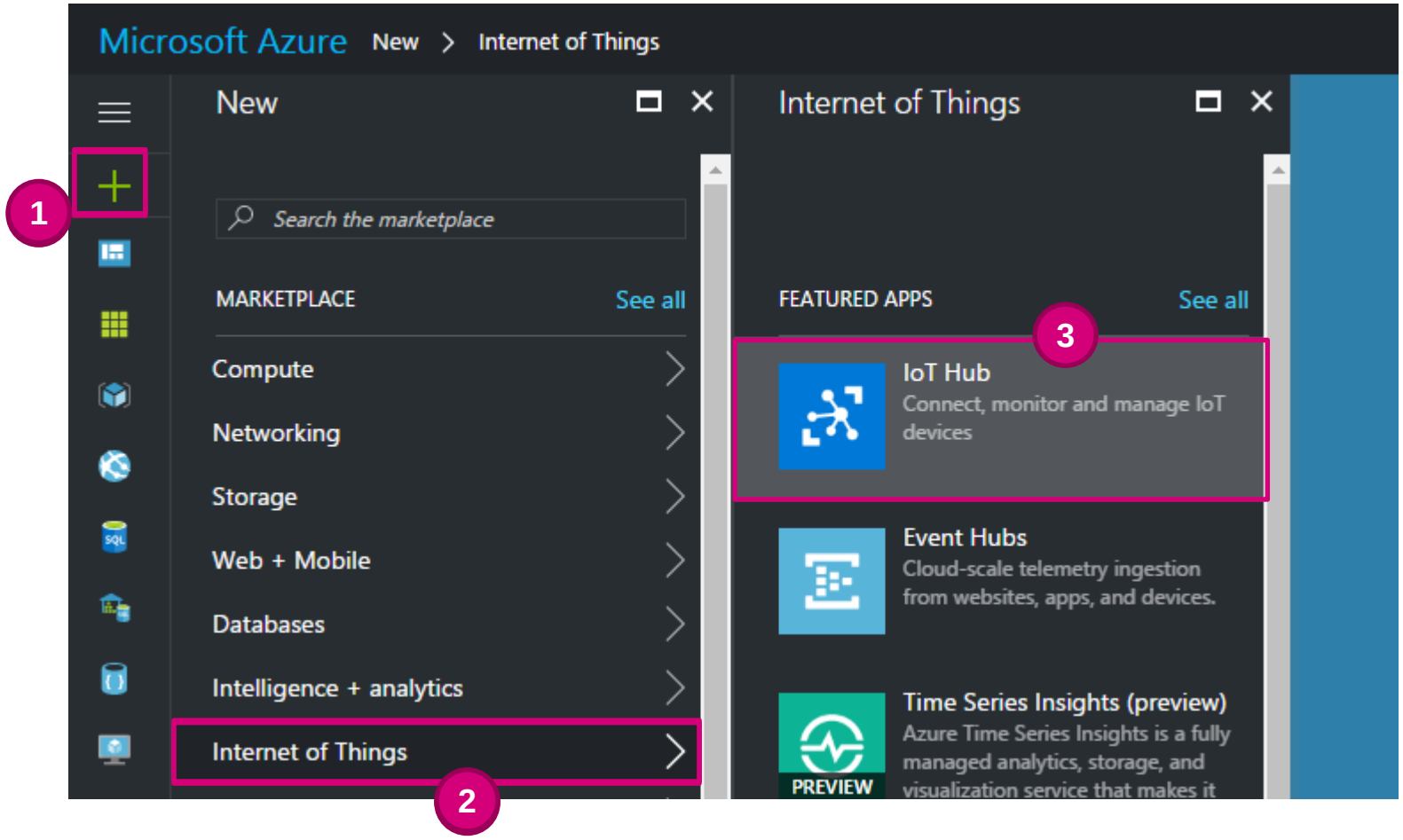




life.augmented

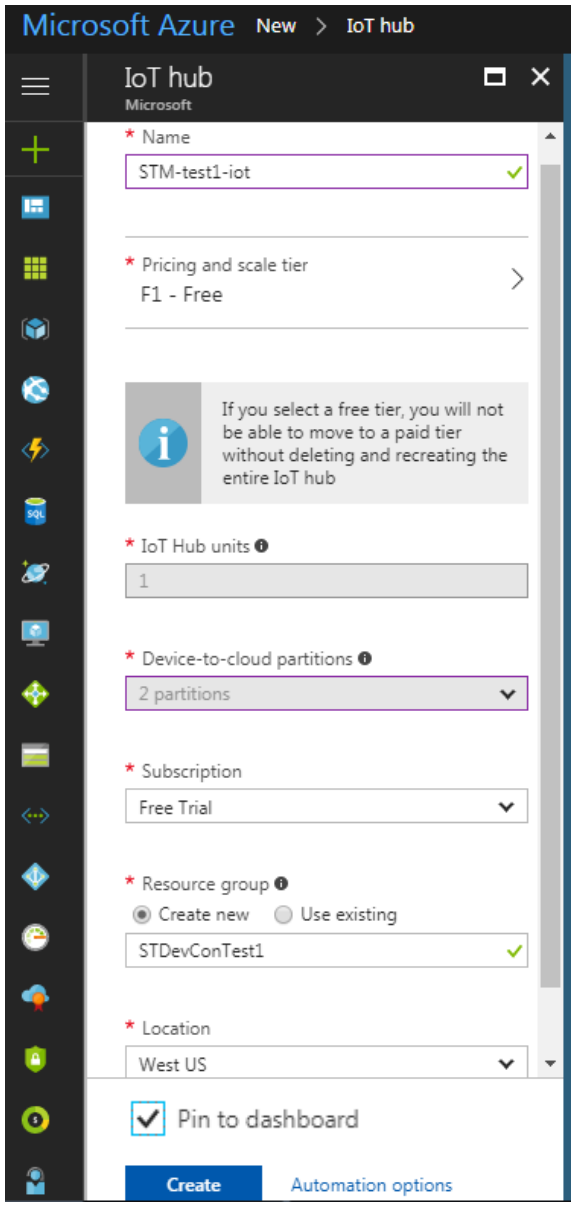
Step 3: Create an IoT Hub

- In the left jump-bar, click on + → **Internet of Things → IoT Hub**



We did these steps for you

Step 3: Create an IoT Hub (Cont'd)



Insert a unique id for your IoTHub

Sample application doesn't require a specific tier; the free tier can be used.

We did these steps for you

The Subscription would be "Free Trial"

You can create a container folder for your Azure services

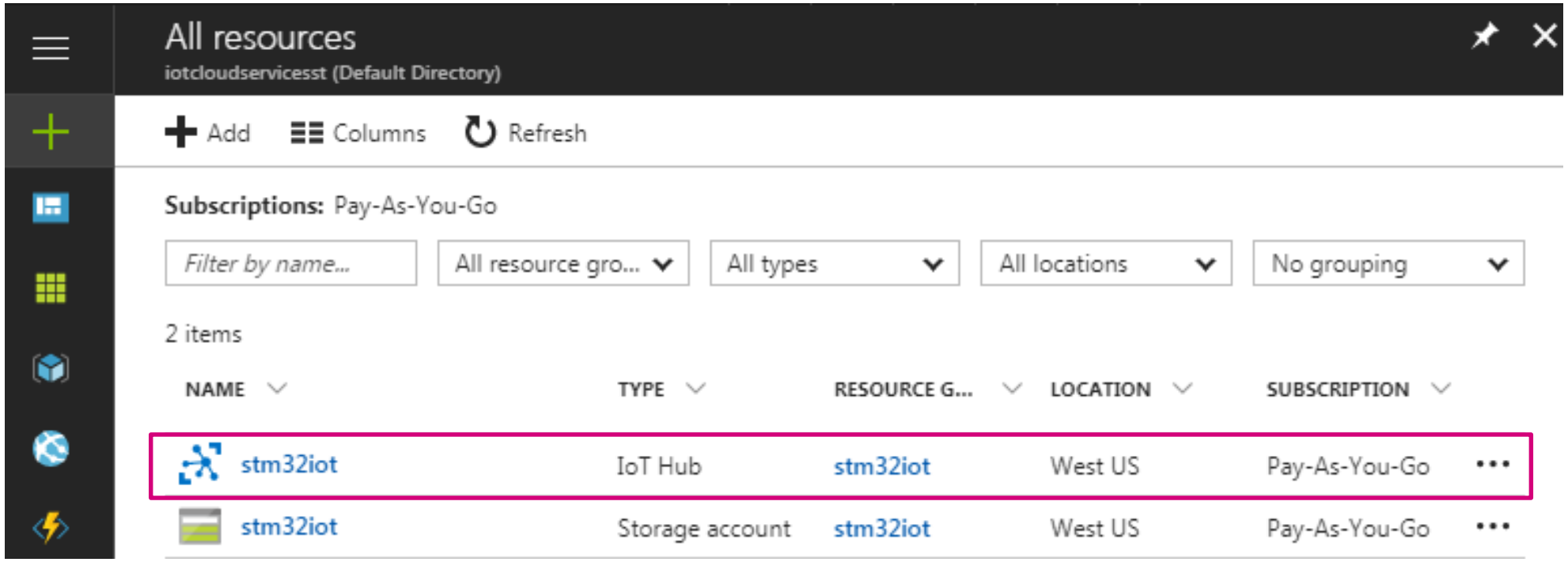
Geographical region where the IoT Hub will be hosted

Create the IoT Hub

life.augmented

IoT Hub Created

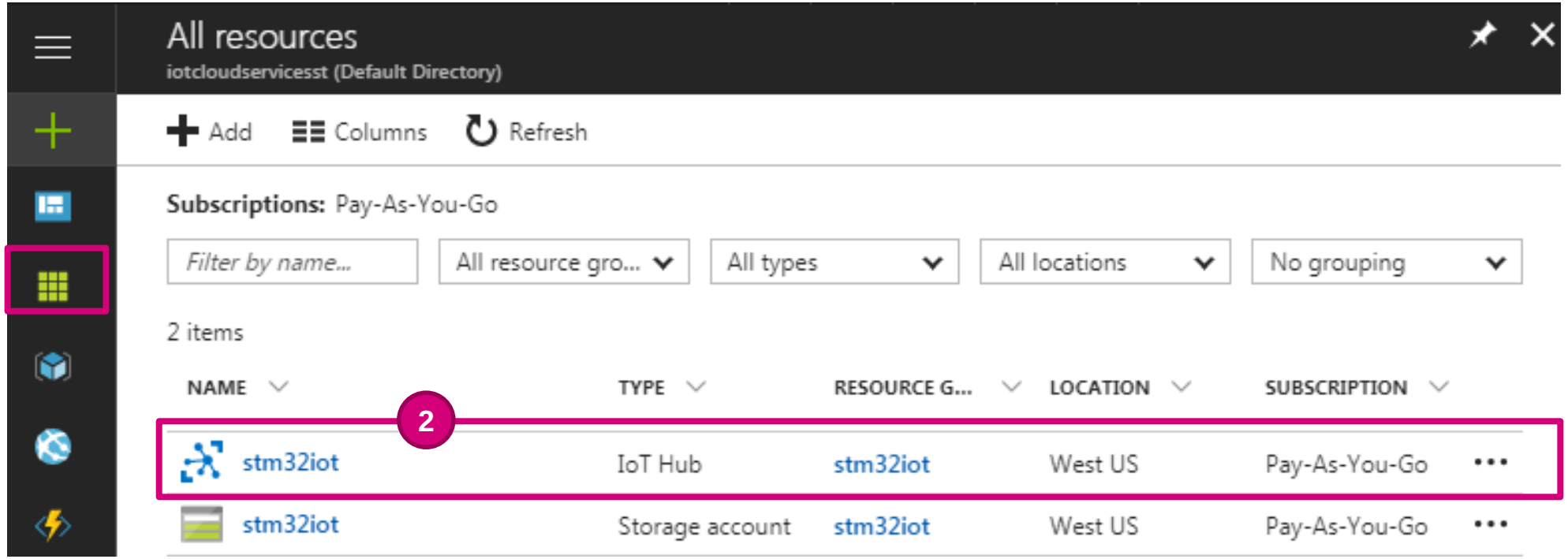
- The IoTHub is now included in the list of resources and resource groups



Step 4: IoT Hub Overview

Select IoT Hub

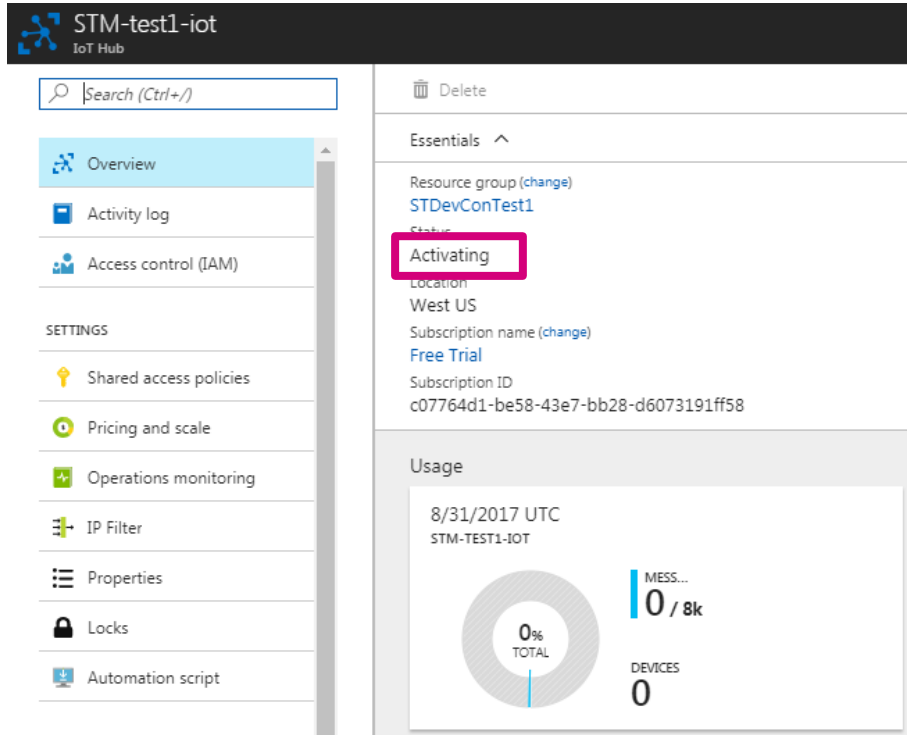
1. Click on the All resources icon to view all you resources
2. Click on the IoT Hub just created from the list of resources in the left jump bar



Step 4: IoT Hub Overview

wait for "***Active***" status (Cont'd)

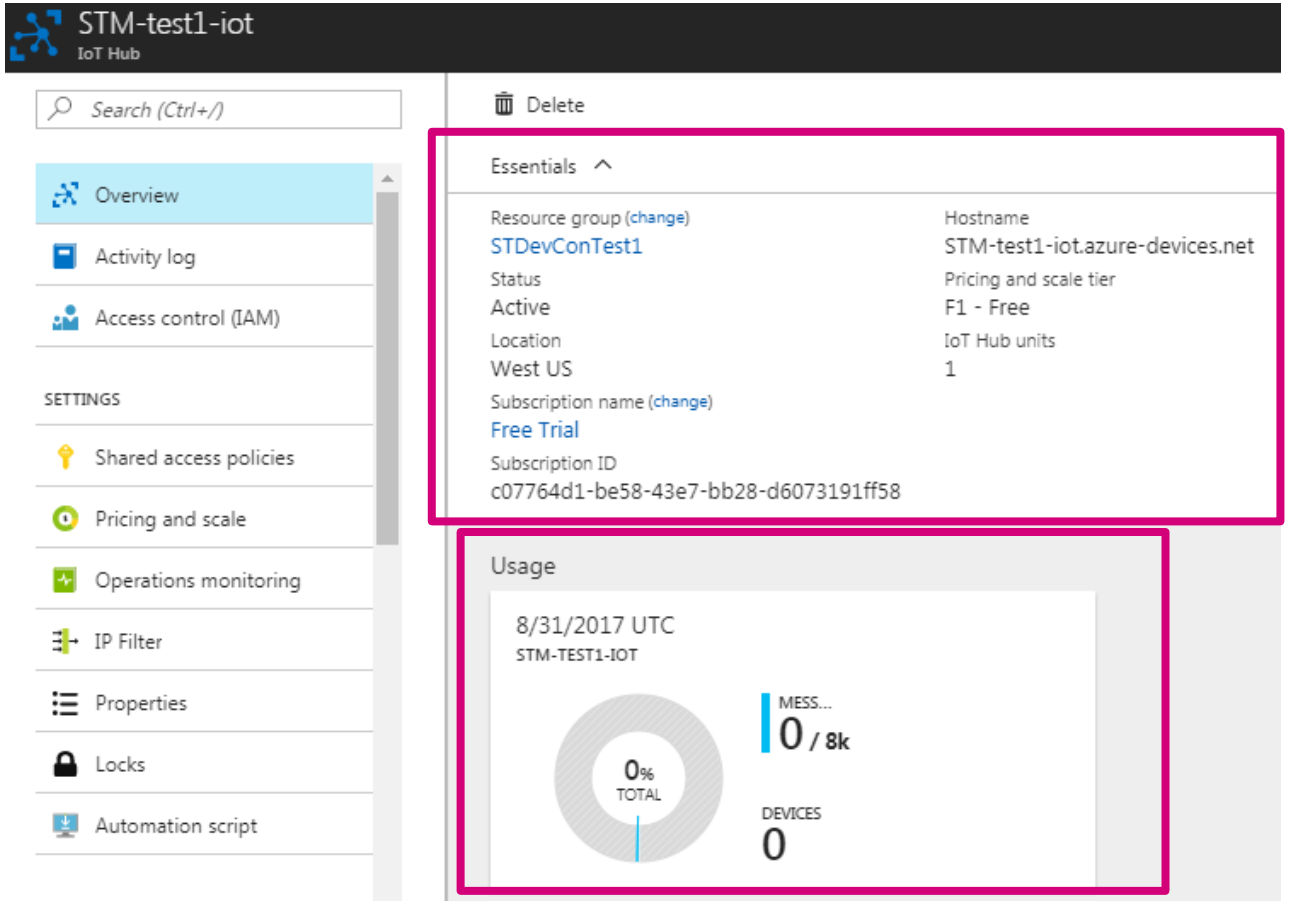
- If you see the status of your newly created IoT Hub as "***Activating***", you may need to wait for a few minutes for activation.



Note: You may also have to log out and log back in for status to change from "***Activating***" to "***Active***" for a newly created IoT Hub.

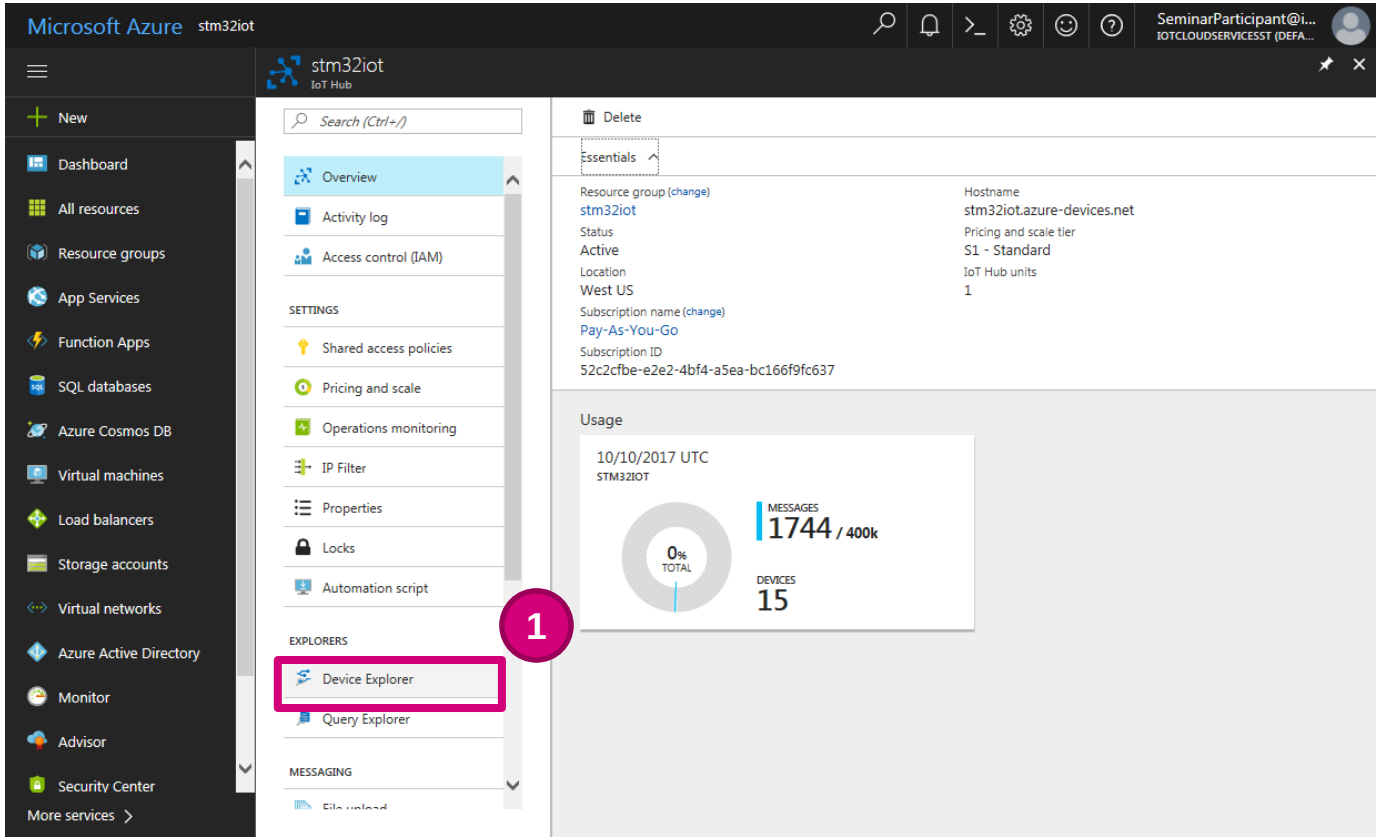
Step 4: IoT Hub Overview

- The overview page will be opened. It shows the **essential data of your IoT Hub** and the **current usage** in terms of registered devices and messages received.



Step 5: Create a new device
in the IoT Hub registry

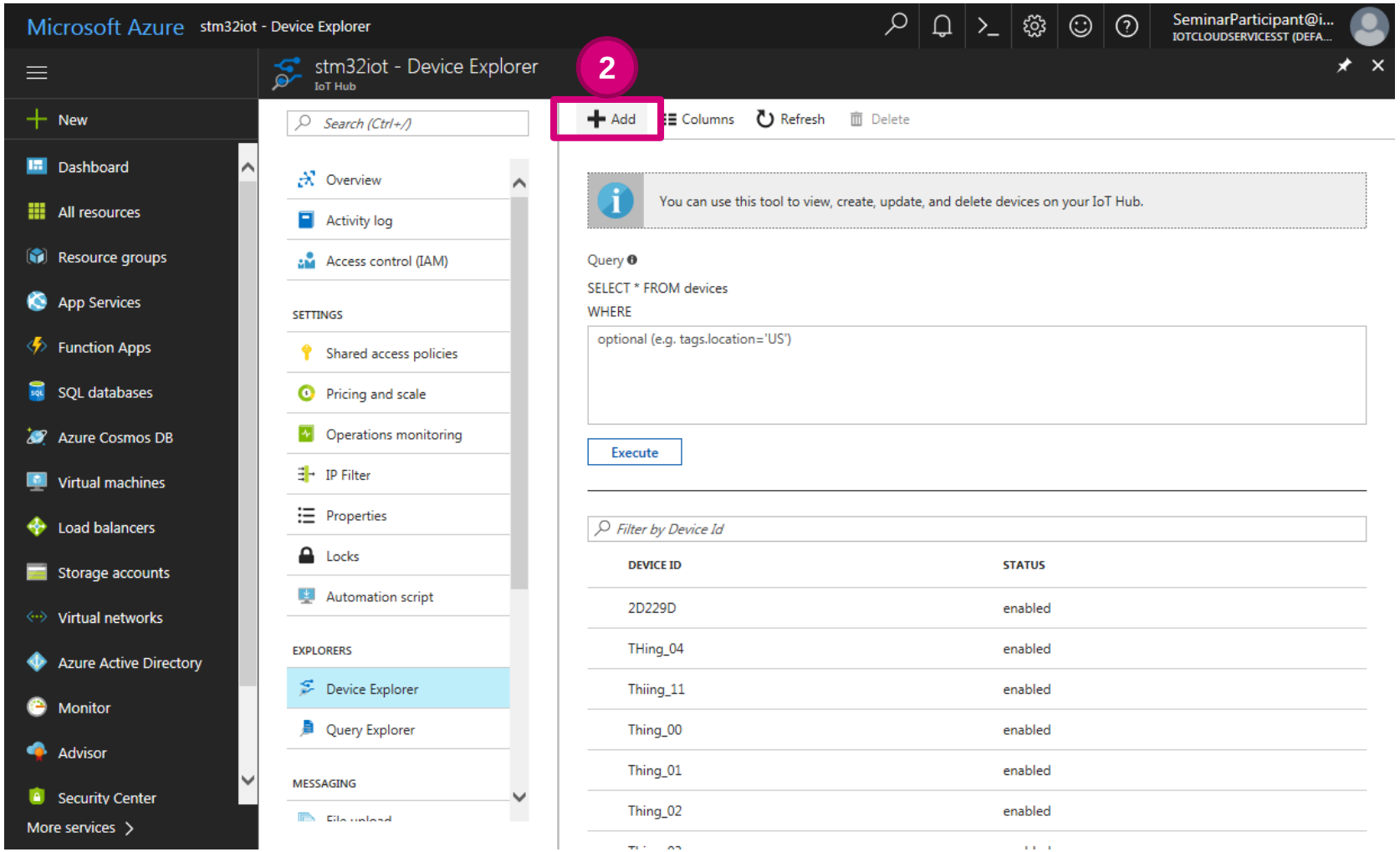
1 Click **Device Explorer**



life.augmented

Step 5: Create a new device
in the IoT Hub registry (Cont'd)

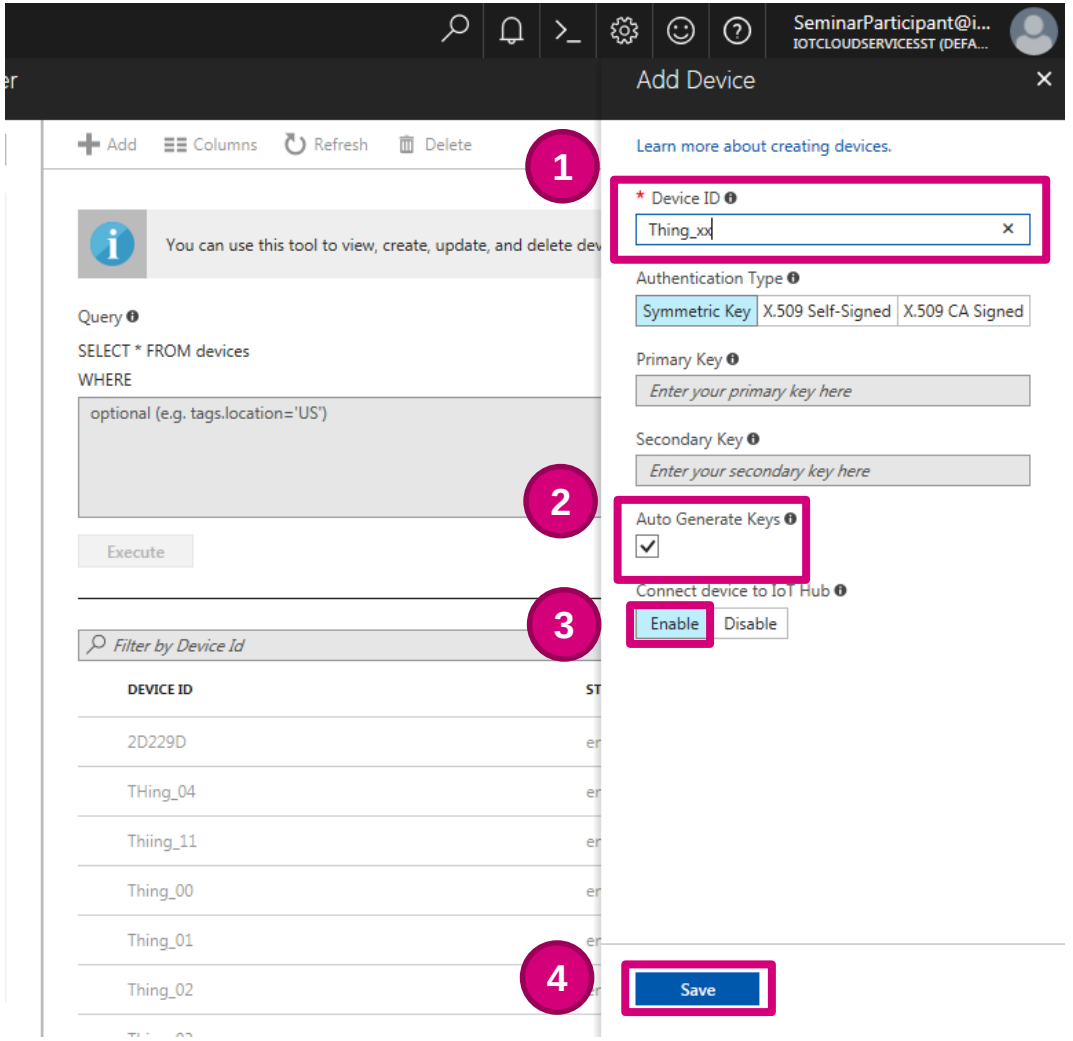
2 Click **Add**



life.augmented

Step 5: Create a new device in the IoT Hub registry

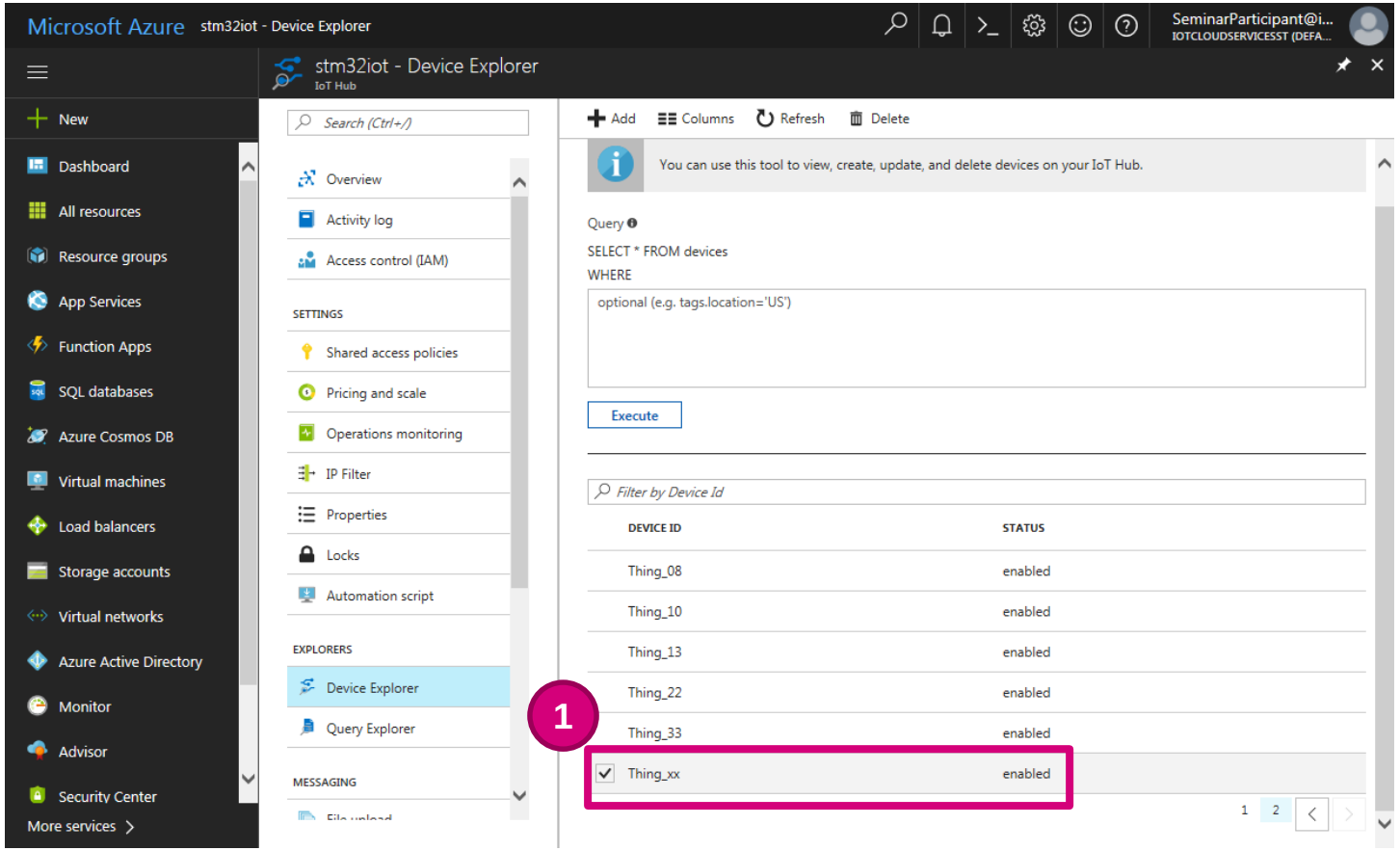
1. Insert your Device ID “Thing_**XX**”: “**XX**” is your participant number (2 characters) printed on the box
2. Check Auto Generate Keys
3. Enable Connect device to IoT Hub
4. Click Save



life.augmented

Step 6: Retrieve Connection string

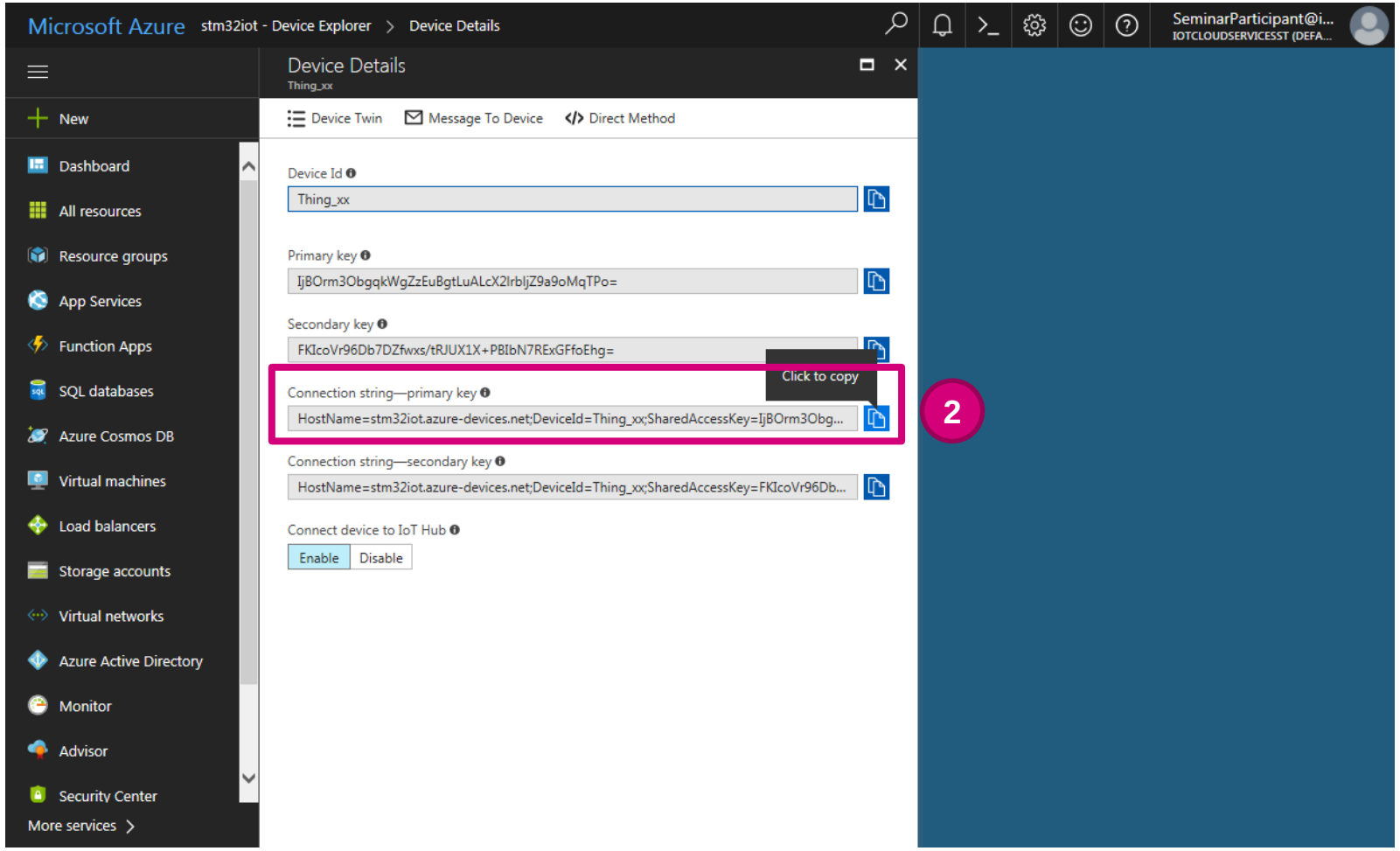
1. After the device is created, open the device in the **Device Explorer** pane.



life.augmented

Step 6: Retrieve Connection string (Cont'd)

2 Copy the Device **Connection string** into Notepad.



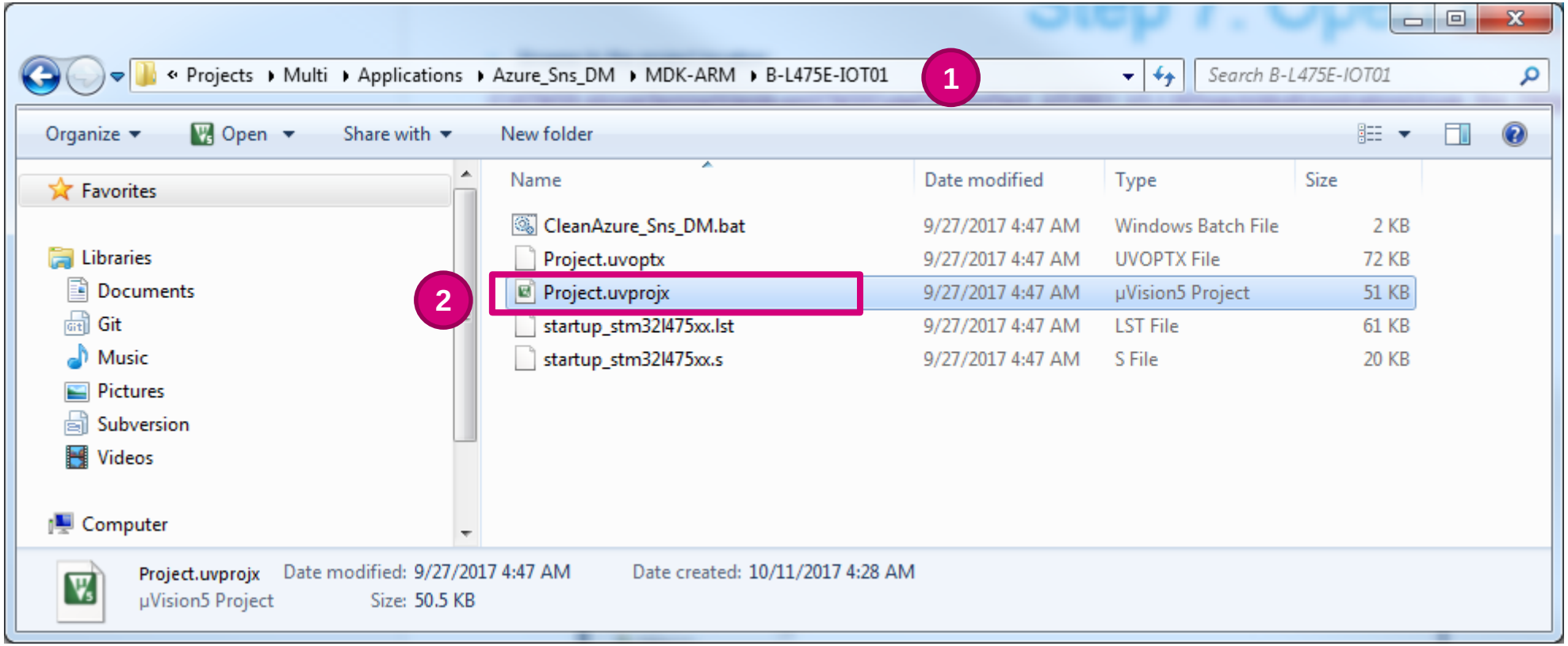
life.augmented

Step 7: Open Keil Project

1. Browse to the project location:

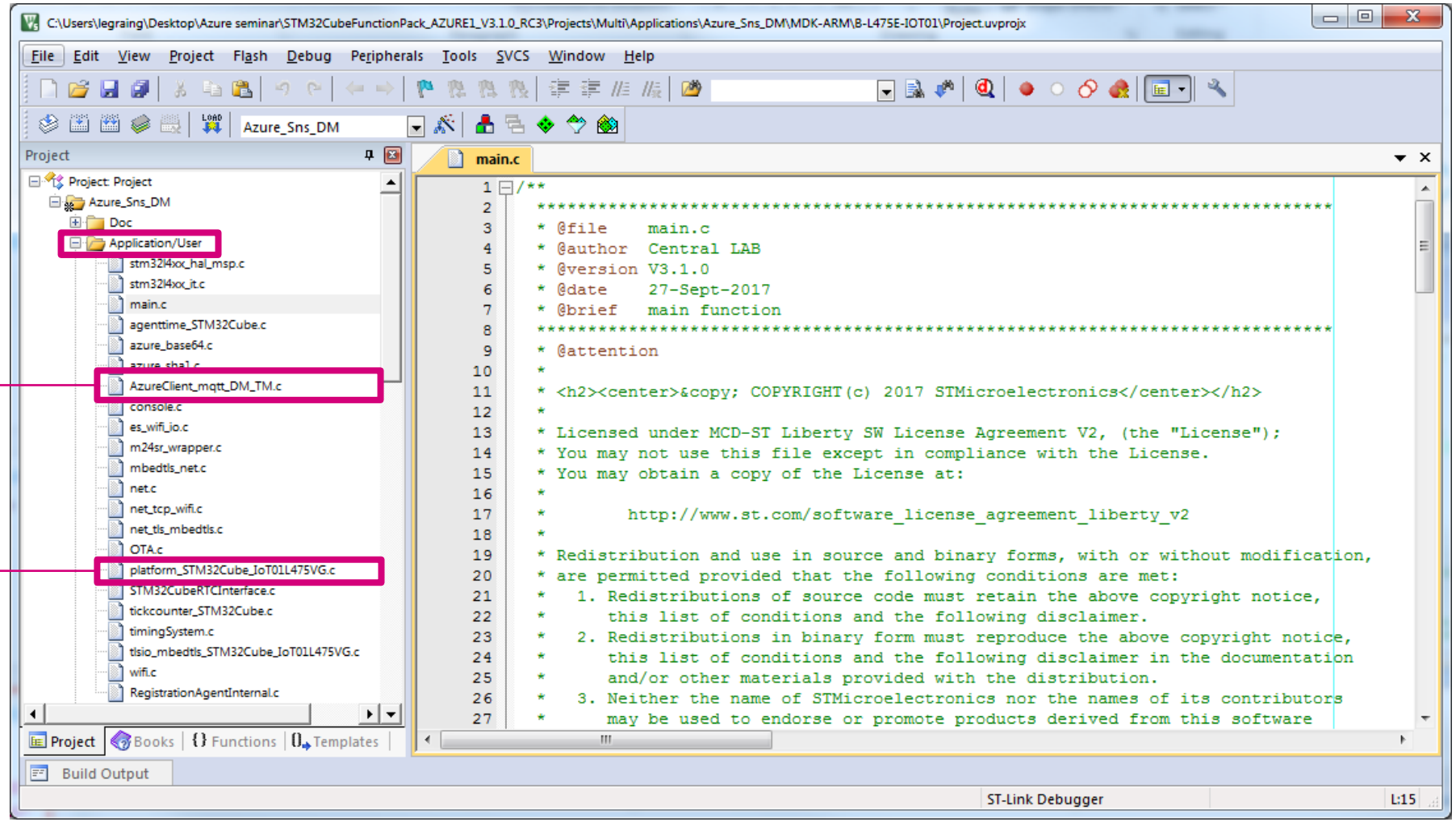
   C:\STM32L4AzureSeminar\Hands-on\STM32CubeFunctionPack_AZURE1_V3.1.0\Projects\Multi\Applications\Azure_Sns_DM\MDK-ARM\B-L475E-IOT01

2. Double click on **Project.uvprojx** project file to bring up Keil



Step 7: Open Keil Project (Cont'd)

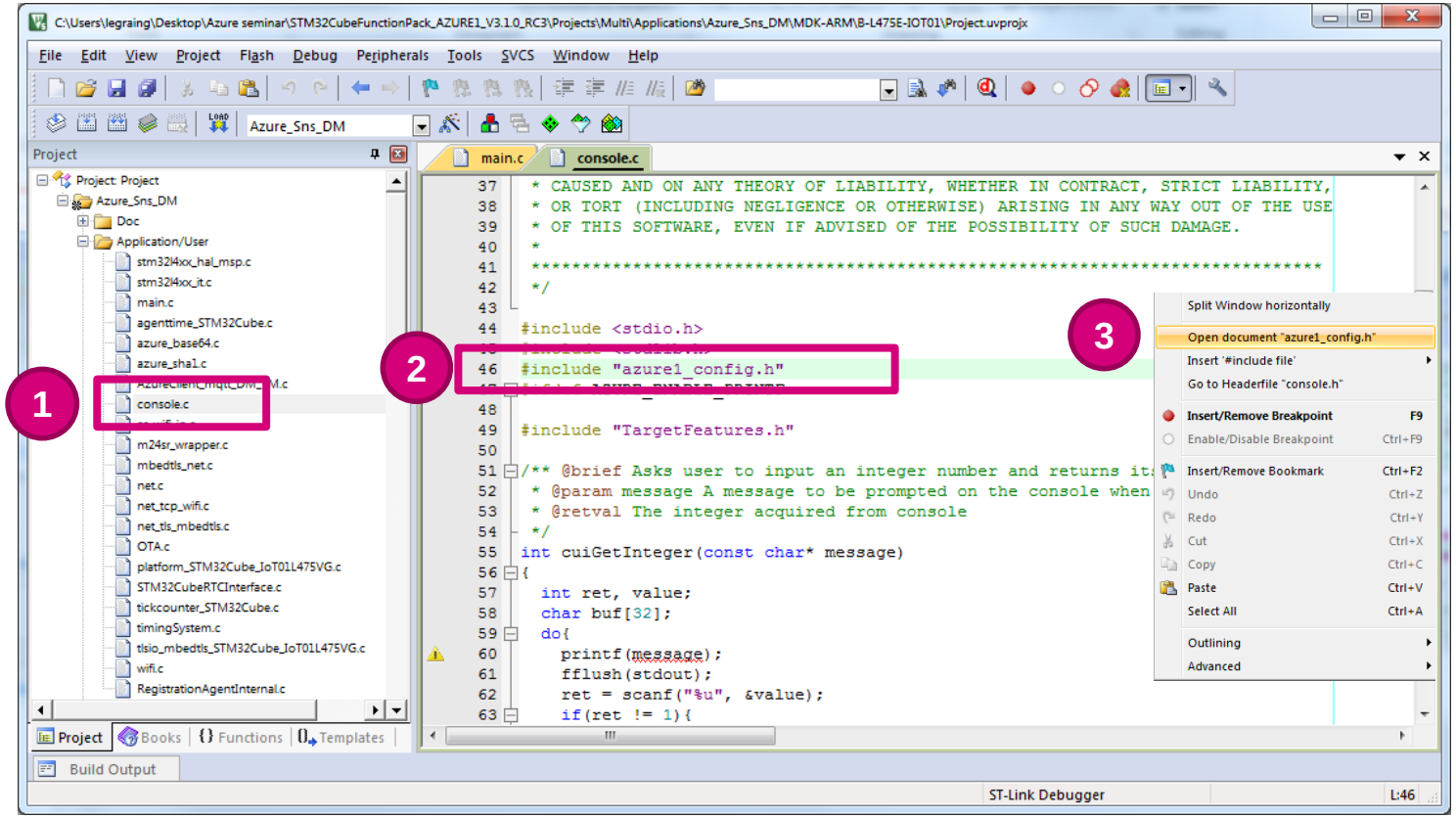
3 Expand Application/User Group in **Project Explorer** window



life.augmented

Step 8: Disable Registration and set Connection String

1. In Project Window double-click on console.c
2. Right-click on line 46
3. Click Open document “azure1_config.h”

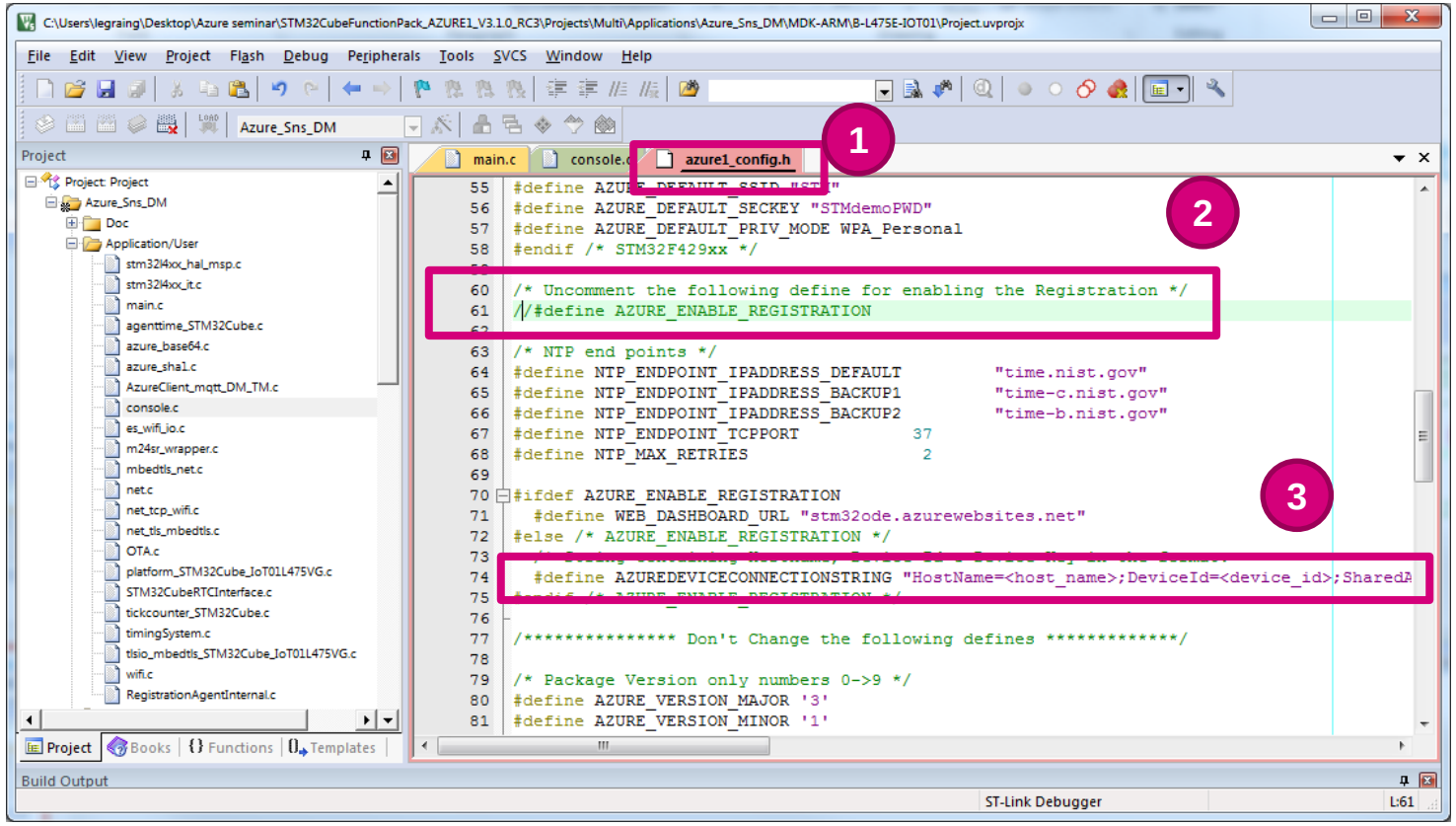


life.augmented

Step 8: Disable Registration

and set Connection String (Cont'd)

1. In file `azure1_config.h`
2. Comment Line 61
3. Uncomment line 74 and paste the Connection String retrieved in previous step (slide 134) replacing `"HostName=<host_name>;DeviceId=<device..."`



Make sure to have the Connection String in between double-quotes aka "…."

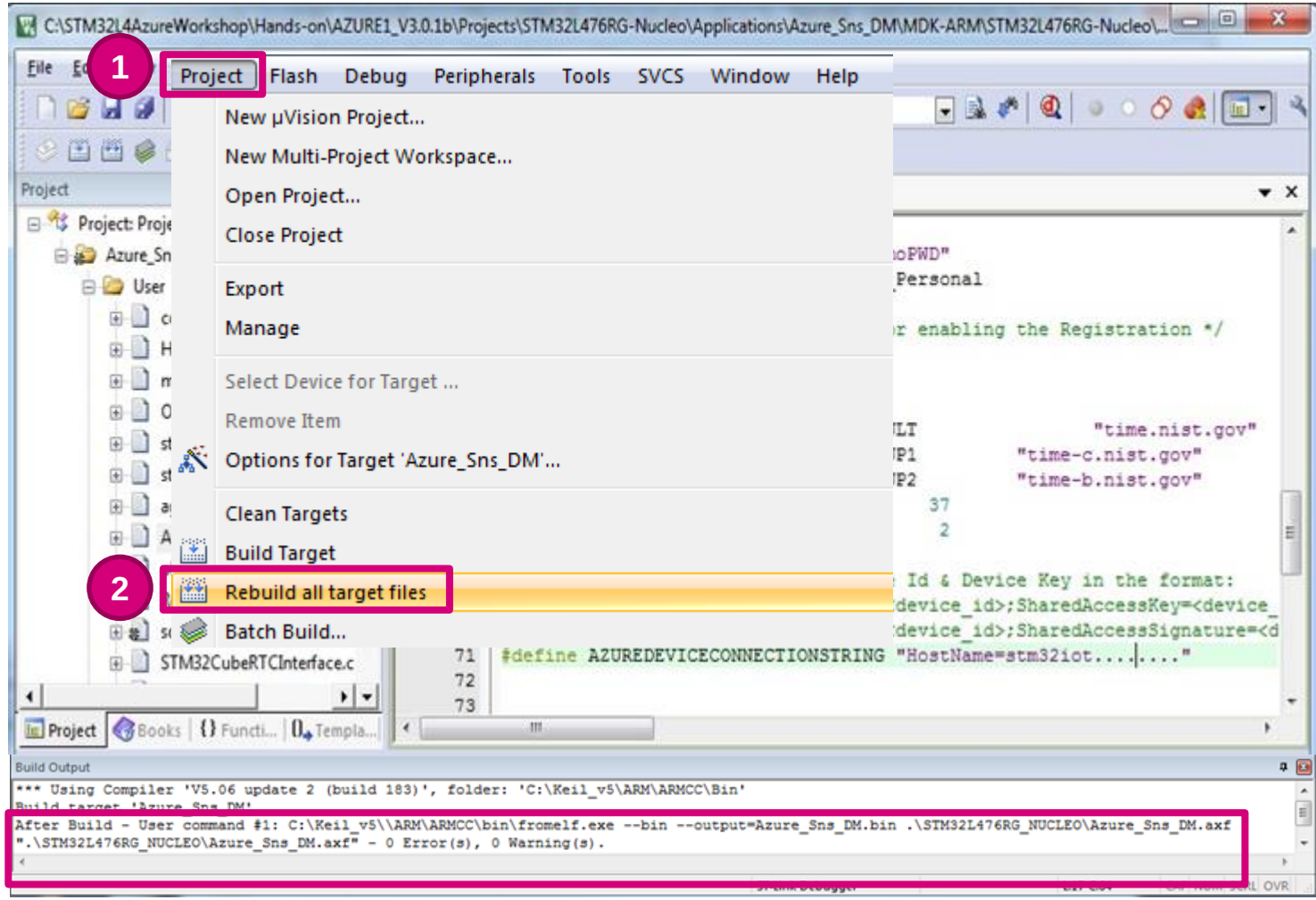
life.augmented

Step 9: Rebuild the Project

1. Click on Project menu
2. Click on Rebuild all target files

In the Build Output window, it should report '`0 Errors(s), 0 Warnings`'

You are now ready to Program your STM32



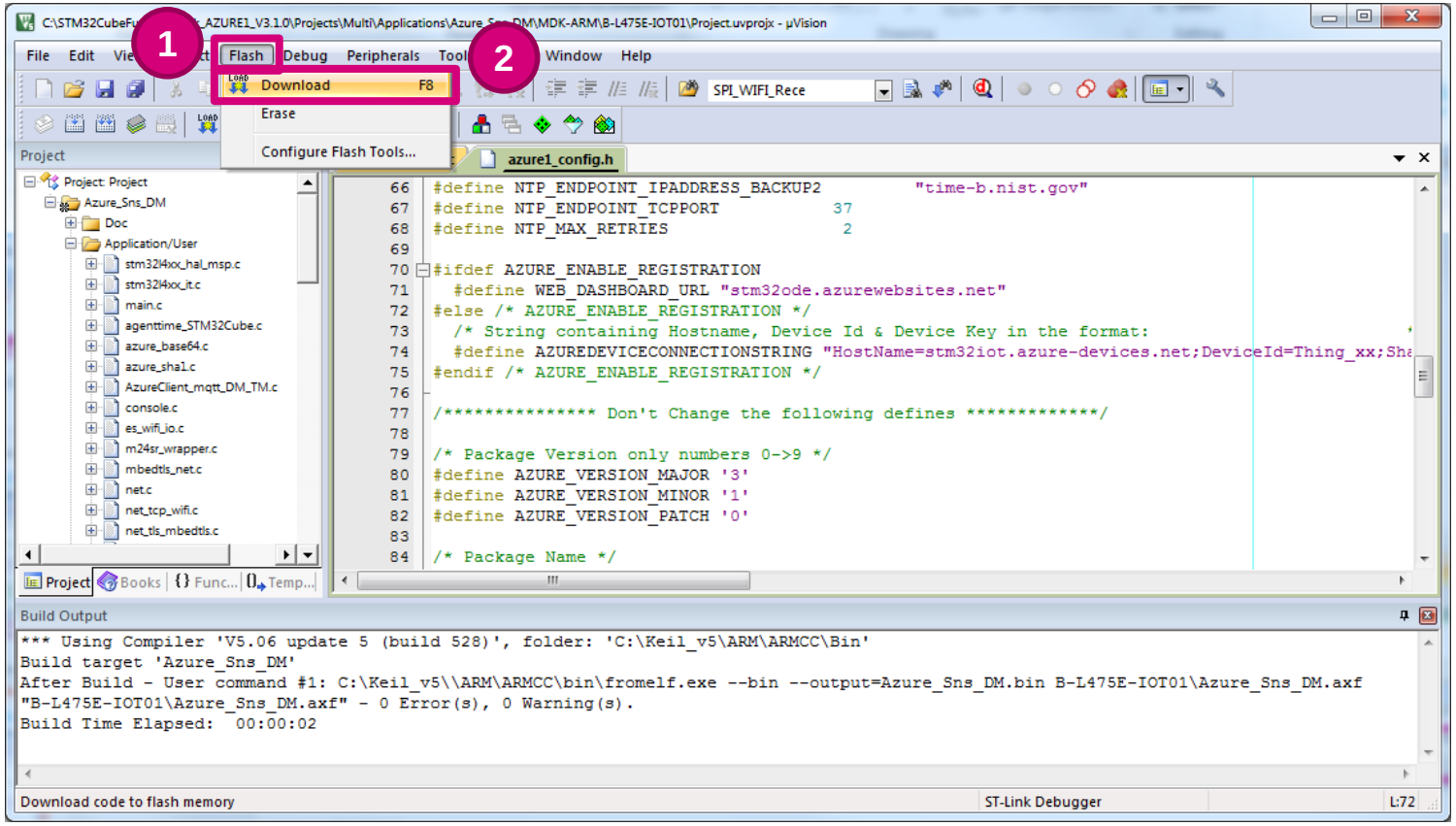
life.augmented

Step 10: Program IoT node

1. Click on Flash menu
2. Click on Download

Progress can be monitored in the bottom left **status bar**.

Your IoT node should now be programmed

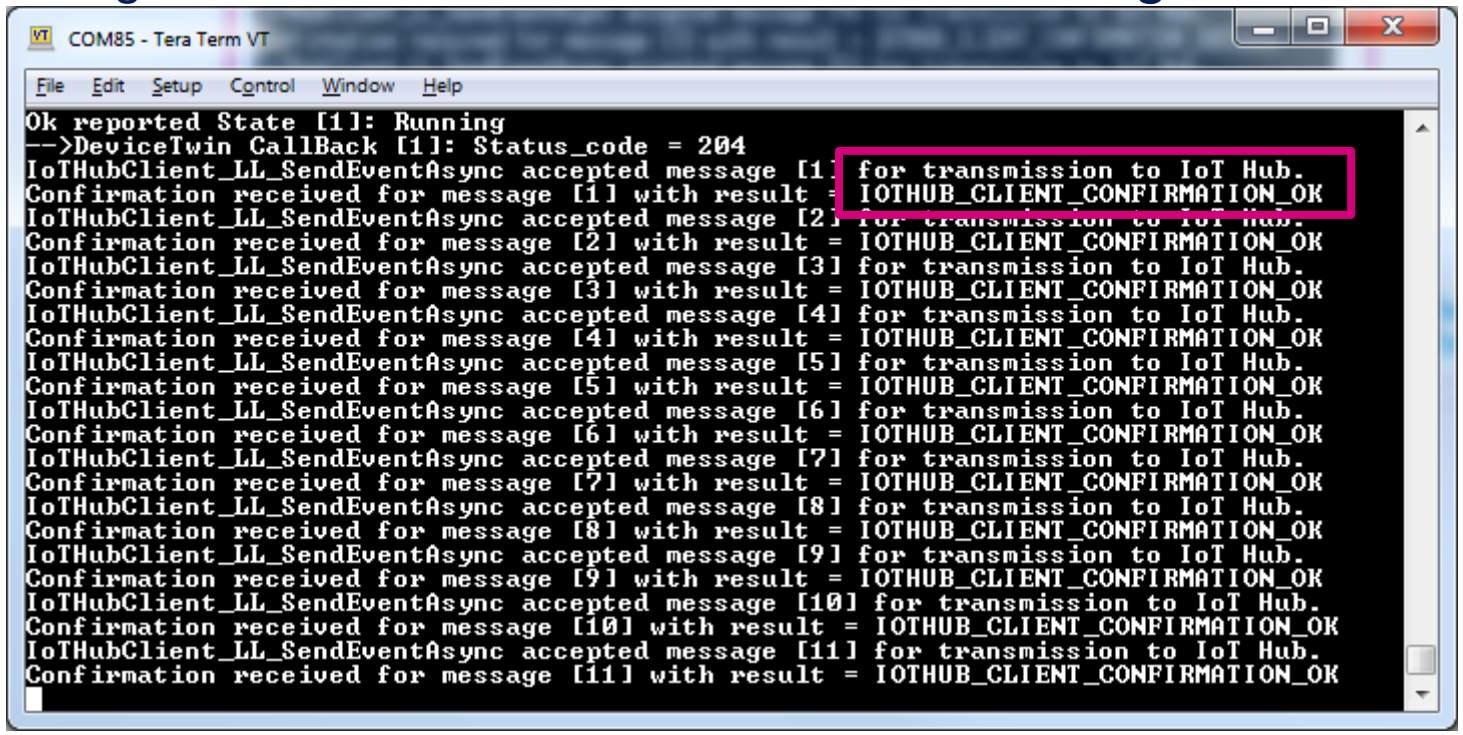


life.augmented

Step 11: Check Connection

- Restart the application by pressing the Reset Button on Nucleo Development Board (Black button). Wait until the board is connected to Wi-Fi access point and to the IoT Hub

- Each 2 seconds, a message is created and transmitted to the IoTHub using MQTT protocol. For each message transmitted, a **confirmation acknowledge** is received back by the device

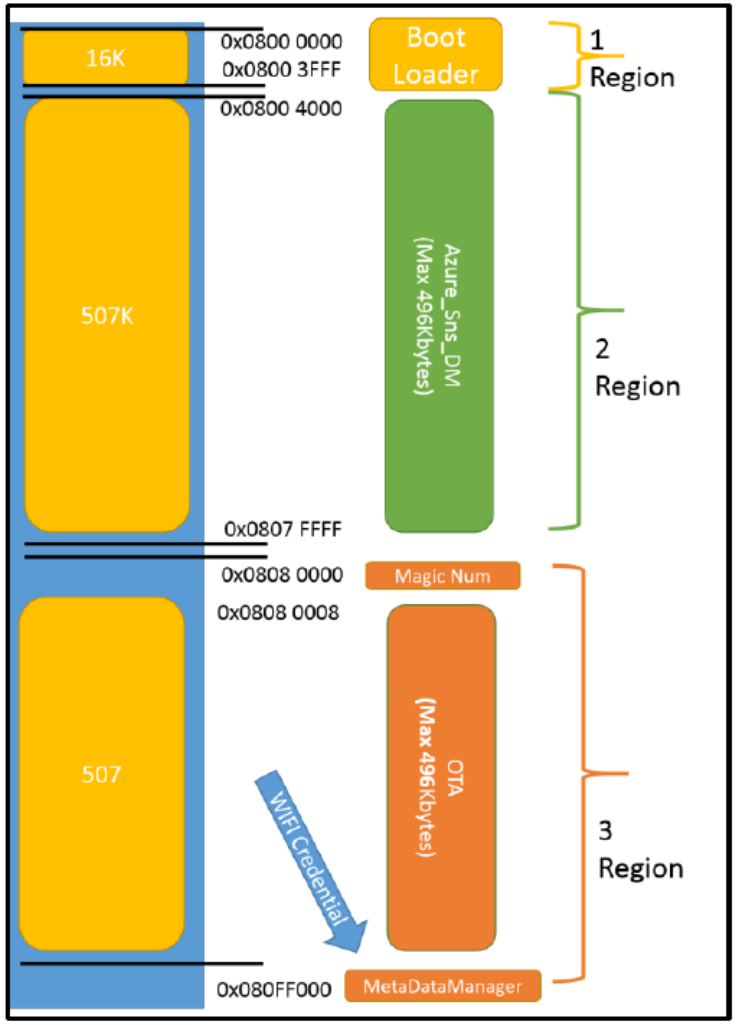


Flash the IoT node

Using Installed Script (Cont'd)

The STM32 flash memory is divided in different regions:

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 credentials inside the Meta Data Manager. The Meta Data Manager is placed at the end of the Flash



life.augmented

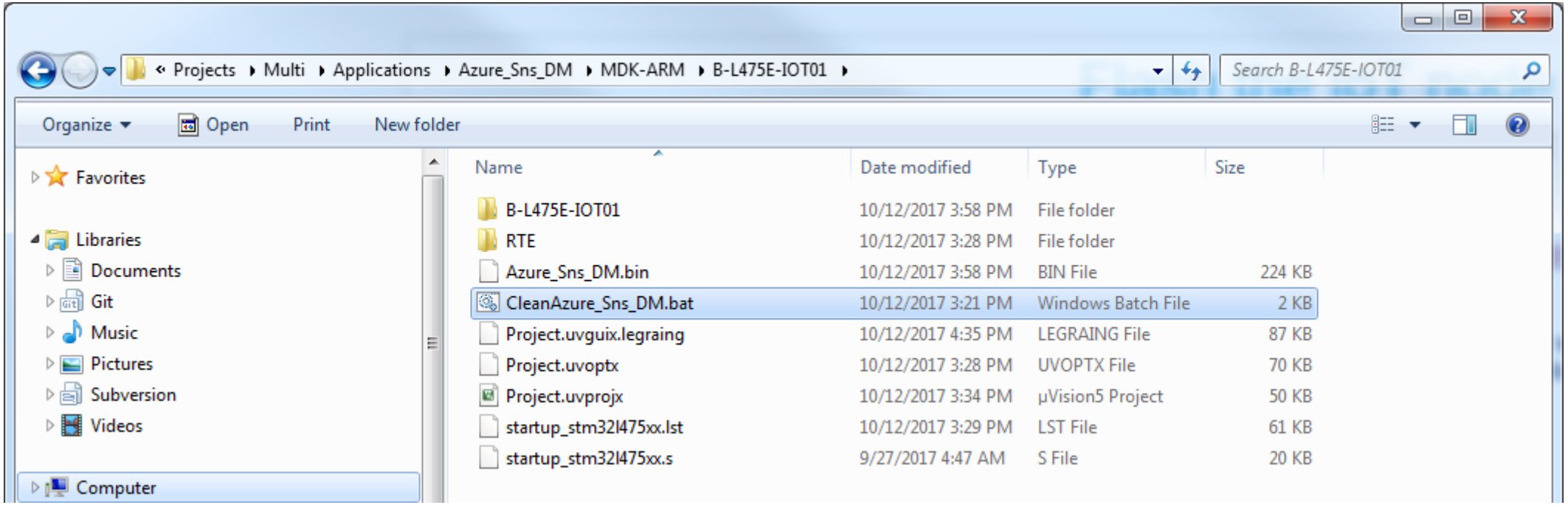
Flash the IoT node

Using Installed Script

- Once built the project, it is necessary to launch the ".bat" script located in folder

C:\STM32L4AzureSeminar\Hands-on\STM32CubeFunctionPack_AZURE1_V3.1.0\Projects\Multi\Applications\Azure_Sns_DM\MDK-ARM\B-L475E-IOT01

In order to flash the MCU, the usage of the script is required to install in the MCU a bootloader together with the project binary. Bootloader is used in Lab 6 for Firmware update.



Flash the IoT node

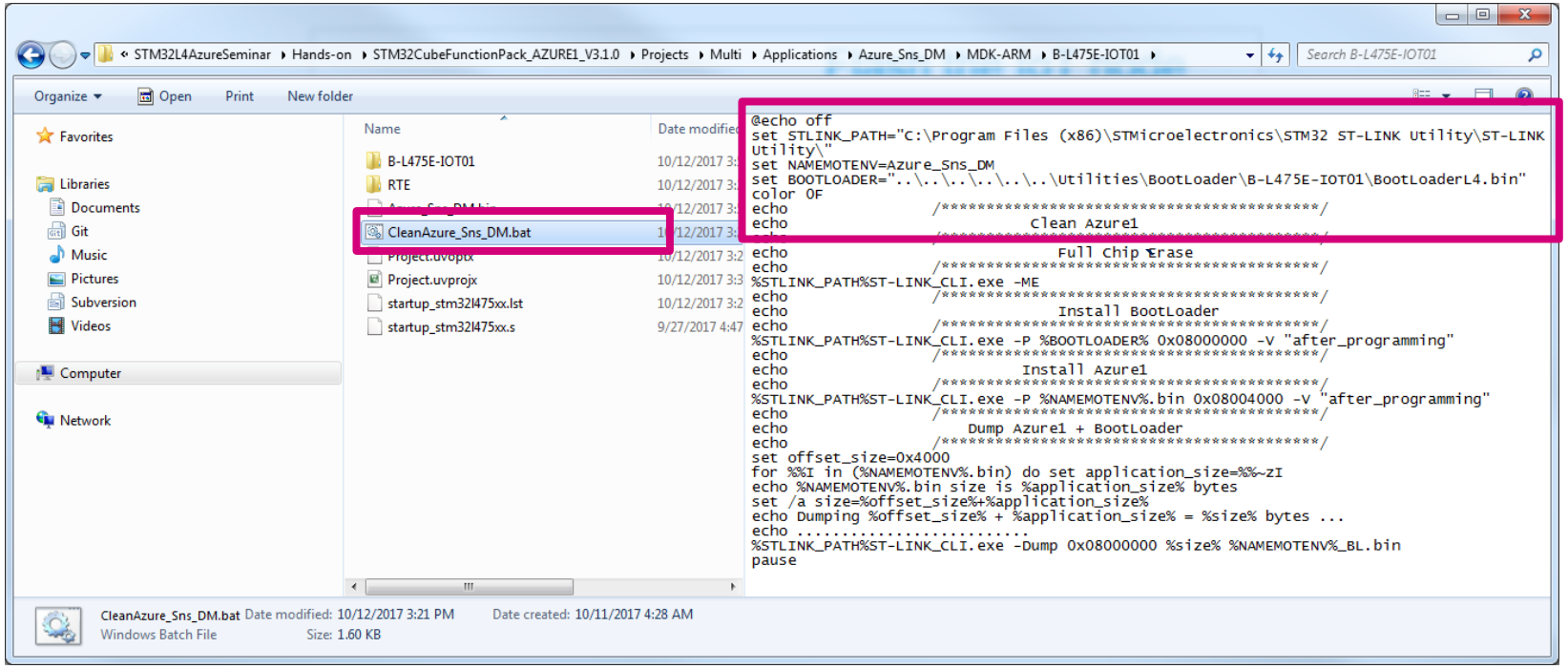
Using Installed Script (Cont'd)

- Windows script uses a tool available with ST-Link utility installed by the installer
- Installation path in Windows7 for ST-Link utility is

`C:\Program Files (x86)\STMicroelectronicsTraining\ST-LINK\`

which is the one used by default in the script

- You can verify that the path in the script correspond to your installation path for ST-Link utility



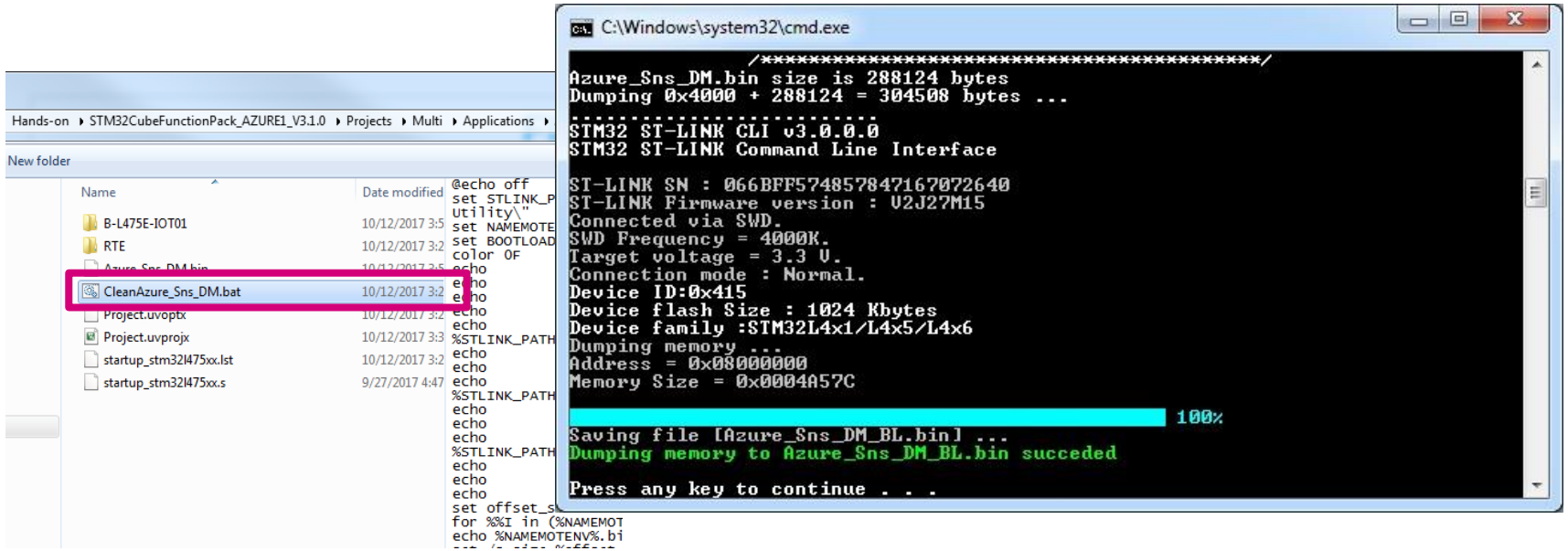
Note: If you see an error "The system cannot find the path specified", please run the ST-Link Utility installer from the path below to install the tool at the default path:

C:\STM32L4AzureSeminar\Software\STSW-LINK004_STM32_ST-LINK_Utility-4.0.0.exe

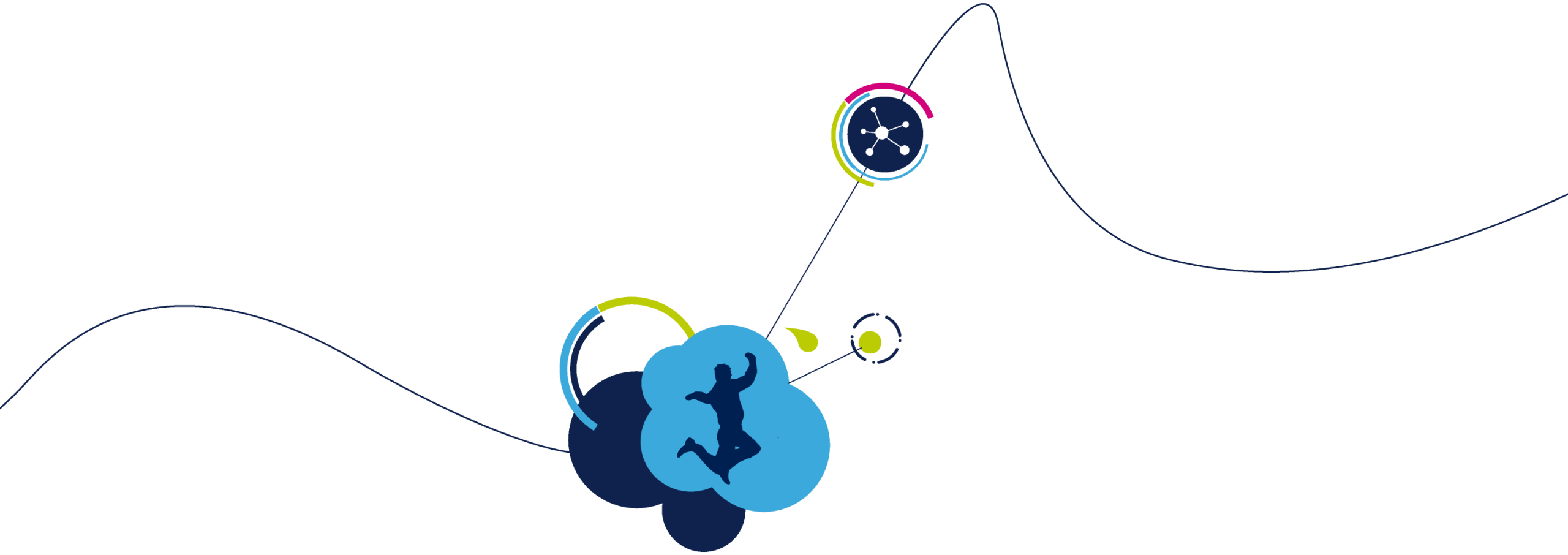
Flash the IoT node

Using Installed Script (Cont'd)

- Double click on CleanAzure_Sns_DM.bat to launch the script



- Once done, the console should show that the board was programmed successfully
- Press any key to close the console



Lab 4: Monitor IoT Node Sensor Data sent to Azure Cloud

life.augmented

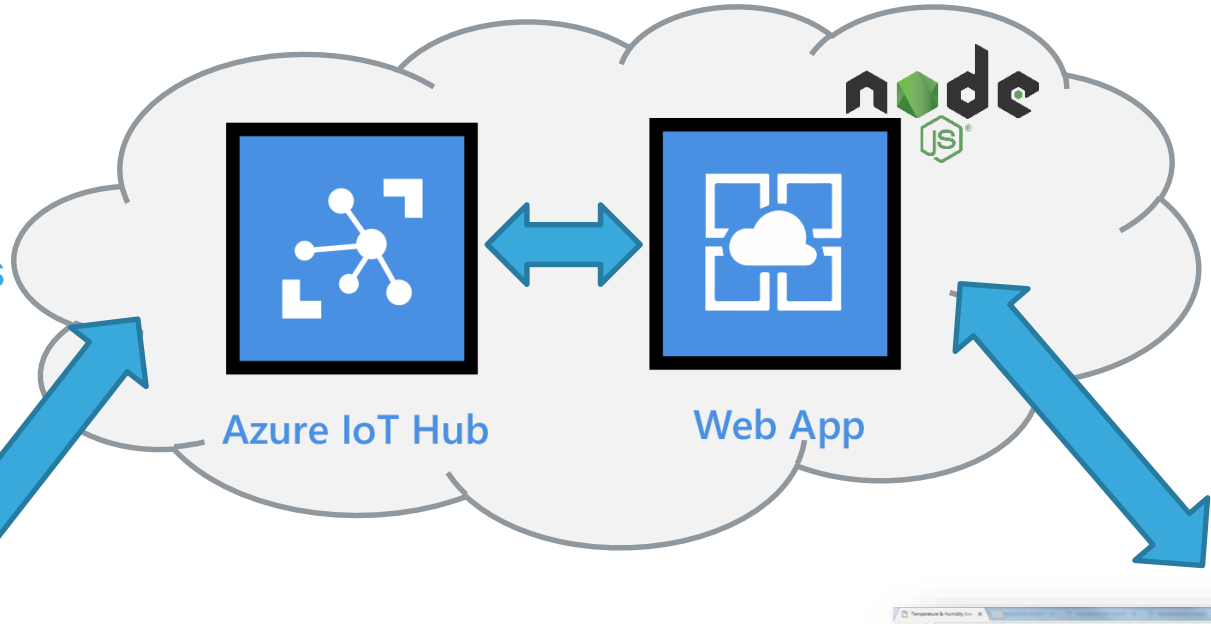
Introduction

- The sample application reads raw data from sensors board, compose a message and through Azure IoT SDK iteratively transmits messages to your IoT Hub
- In this lab you will learn how to visualize the log of messages received by your IoT Hub using a Web App
- You will also learn how to customize and extend data read from the sensor board

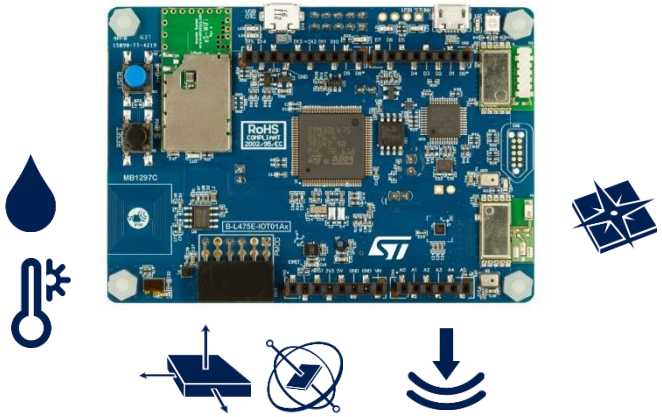
Introduction

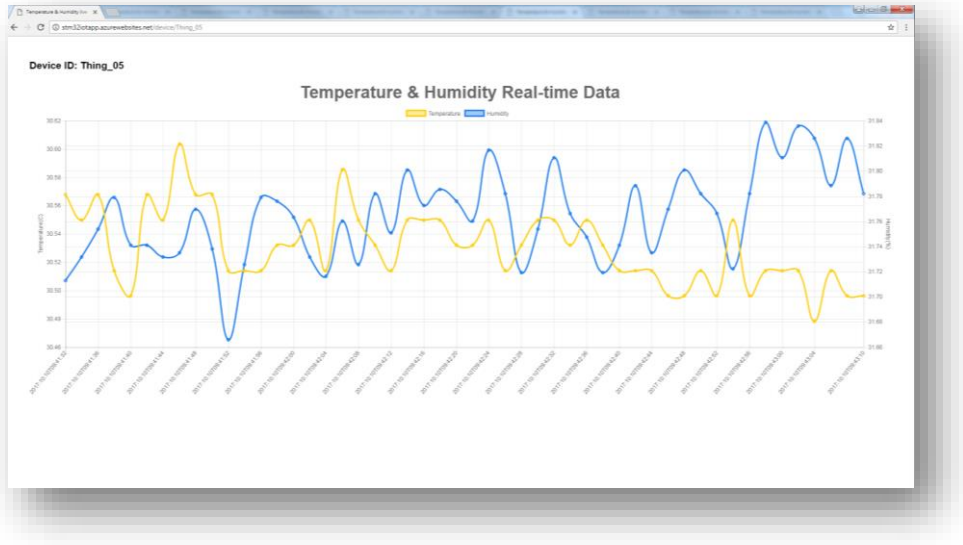
MQTT
device-to-cloud messages

```
{
    "deviceId":"c47f51037c8e",
    "messageId":3123,
    "Temperature":30.719666,
    "Humidity":30.874012,
    "ts":"2017-10-10T22:03:14Z"
}
```



HTTP & WebSocket



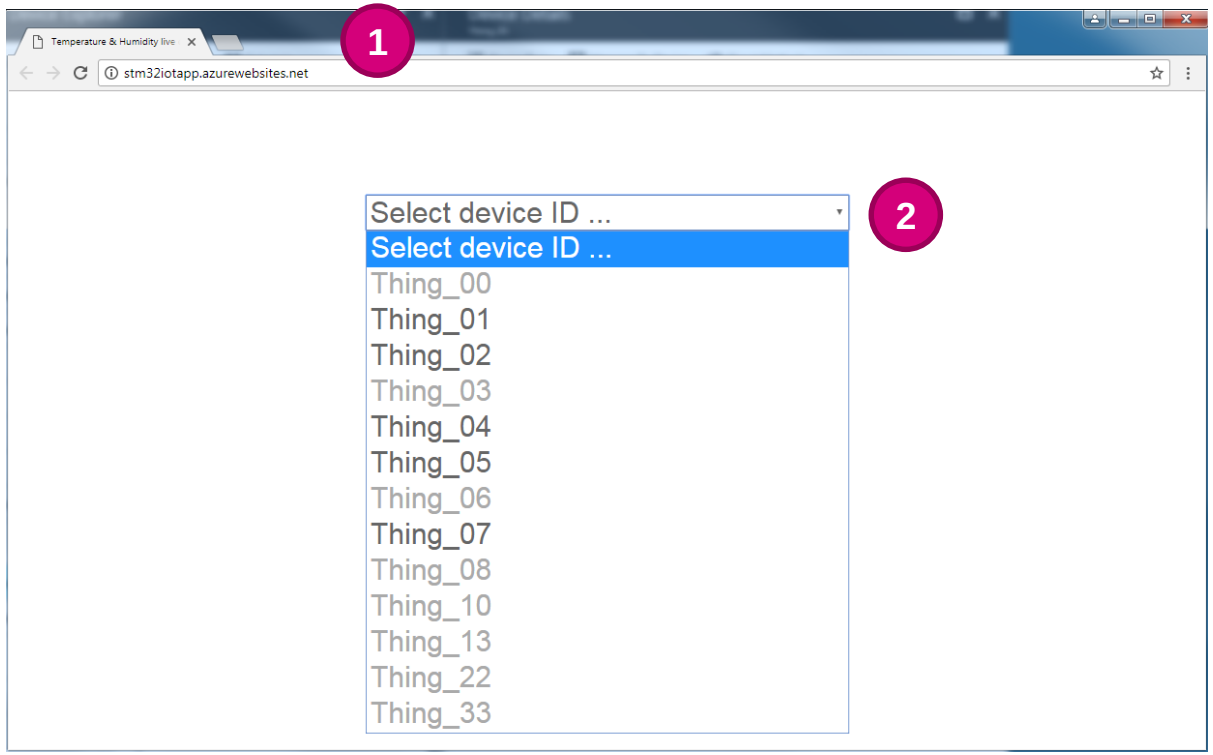


life.augmented

Step 1: Open Web App

1. Open
   - http://stm32iotapp.azurewebsites.net

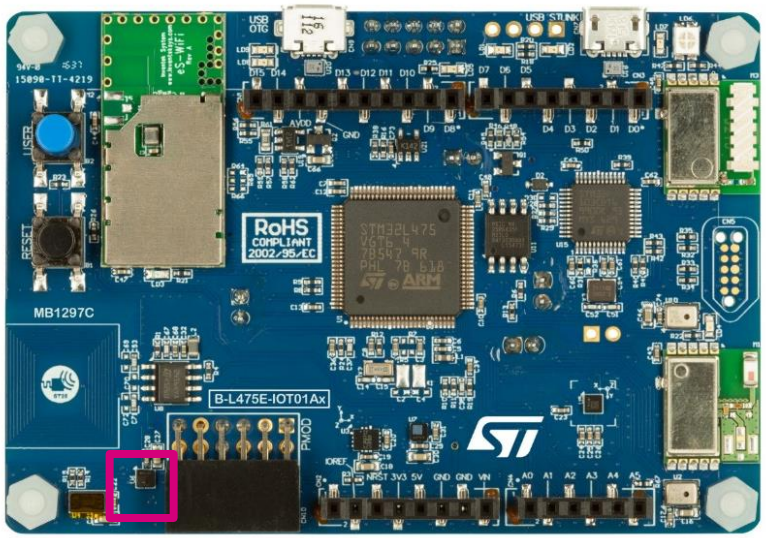
2. Select **Device ID**

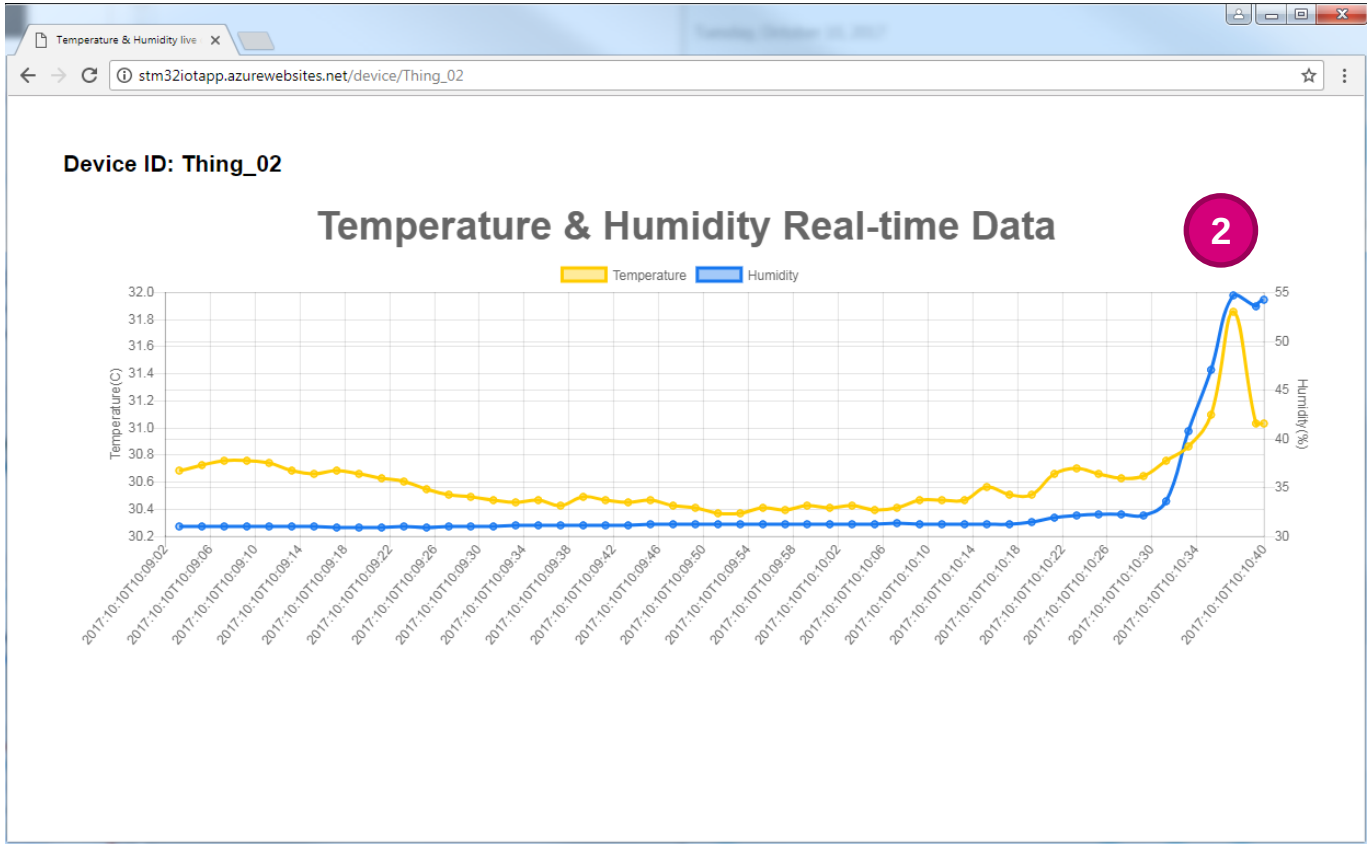


Step 2: Visualize real-time sensor data

1. Optionally, check data by blowing onto the temperature sensor

2. Monitor the spike in the real-time visualization





life.augmented

Create your own IoT Web App

- To learn how to create your own “Web App”, go to:

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-live-data-visualization-in-web-apps

- You will learn to
  - Create a web app in the Azure portal.
  - Get your IoT hub ready for data access by adding a consumer group.
  - Configure the web app to read sensor data from your IoT hub.
  - Upload a web application to be hosted by the web app.

Customize Usage of the Sensor Board

- The message transmitted to your IoT Hub is composed inside function **`SendSNSData()`** in file `AzureClient_mqtt_DM_TM.c` (Azure_Sns_DM/User)

```
/* Read the Data from the Sensors */
FillTheModelInstance();
```

Read raw data from sensors to Azure1 data structure, using driver APIs (**`BSP_TEMPERATURE_Get_Temp, BSP_TEMPERATURE_Get_Hum, etc.`**)

```
messages = (EVENT_INSTANCE *) calloc(1,sizeof(EVENT_INSTANCE));
if(messages==NULL) {
  AZURE_PRINTF("Err: Allocating Memory for messages to IoT Hub\r\n");
  while(1) {
    ;
  }
} else {
  messages->this = (void *)messages;
}

if (SERIALIZE(&destination, &destinationSize,
              Azure1->id    , Azure1->name, Azure1->ts,
              Azure1->mtype, Azure1->temp, Azure1->hum,
              Azure1->accX , Azure1->accY, Azure1->accZ,
              Azure1->gyrX , Azure1->gyrY, Azure1->gyrZ) != CODEFIRST_OK){
  AZURE_PRINTF("Err: Failed to serialize\r\n");
  while(1) {
    ;
  }
}
```

Convert sensors data structure into a JSON string

```
if ((messages->messageHandle = IoTHubMessage_CreateFromByteArray(destination, destinationSize)) == NULL) {
  AZURE_PRINTF("Err: iotHubMessageHandle is NULL!\r\n");
} else {
```

Create message handler

```
if (IoTHubClient_LL_SendEventAsync(iotHubClientHandle, messages->messageHandle, SendConfirmationCallback, messages) != IOTHUB_CLIENT_OK) {
  AZURE_PRINTF("Err: IoTHubClient_LL_SendEventAsync.....FAILED!\r\n");
} else {
  AZURE_PRINTF("IoTHubClient_LL_SendEventAsync accepted message [%d] for transmission to IoT Hub.\r\n", SentMessagesCount);
```

Send the message

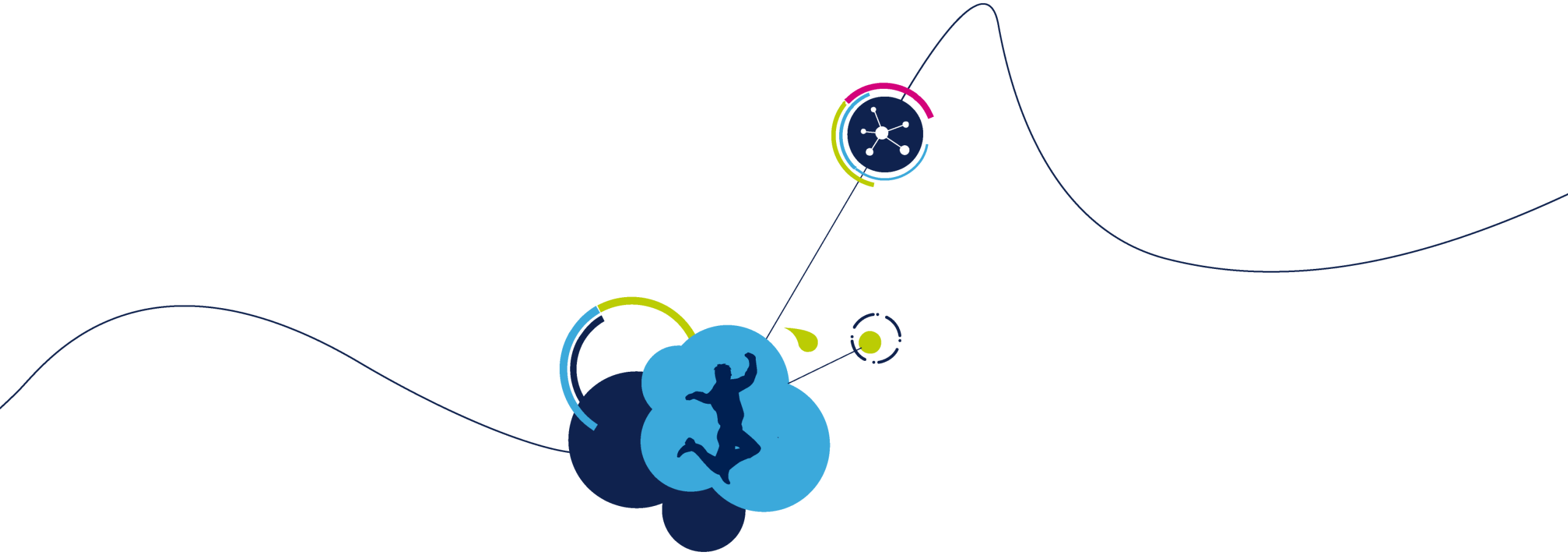
Customize Usage of the Sensor Board (Cont'd)

- Function **`SendSNSData()`** is called by a timer interrupt every two seconds. Transmission rate for messages can be changed using Desired Properties or directly in the code in `AzureClient_mqtt_DM_TM.c` by using **`ChangeTelemetryInterval`** API

```
void ChangeTelemetryInterval(void* argument)
{
  Azure1_t *Azure = argument;
  if((Azure1->DesiredTelemetryInterval>0) & (Azure1->DesiredTelemetryInterval<=30)){
    Azure1->TelemetryInterval = TargetBoardFeatures.TIM_CC1_Pulse = Azure1->DesiredTelemetryInterval*2000 /* 2Hz Timer Frequency */;
  } else {
    AZURE_PRINTF("Err: Received a Not Allowed desired Telemetry Interval=%d (1<=Allowed<=30)\r\n",Azure->DesiredTelemetryInterval);
  }
}
```

- By default the message transmitted to IoT Hub contains the following raw sensors data
  - Temperature, Humidity, Accelerometers (X,Y,Z), Gyroscope (X,Y,Z)

- The sensors board also generates raw sensor data related to:
  - Pressure (inside function `FillTheModeInstance()` use API: **`BSP_PRESSURE_Get_Press)`**
  - Magnetometer (inside function `FillTheModeInstance()` use API: **`BSP_MAGNETO_Get_Axes`**)



Lab 5: Manage IoT Node from Azure portal

life.augmented

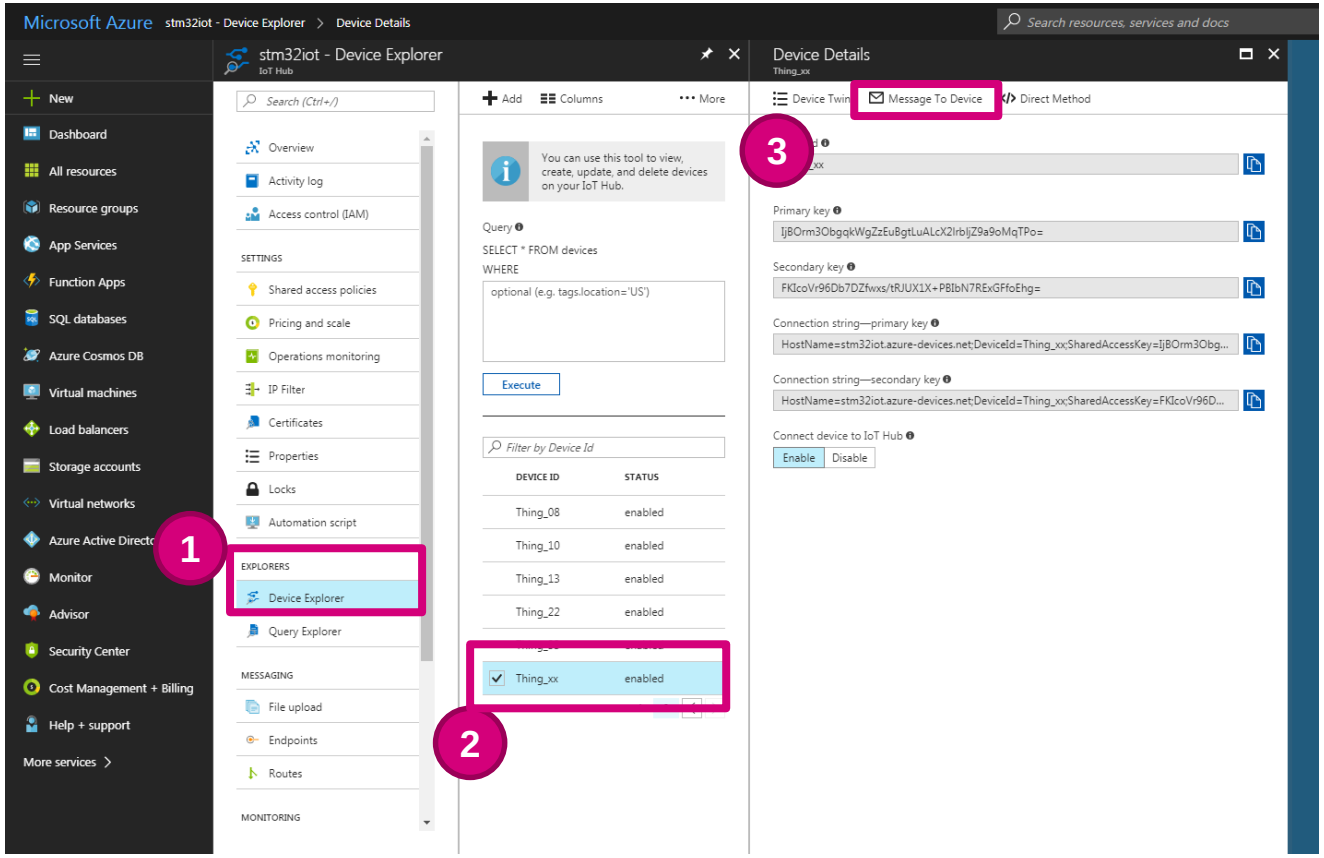
Introduction

- IoT Hub provides three options for device apps to expose functionality to a back-end app:
  - Direct methods for communications that require immediate confirmation of the result. Direct methods are often used for interactive control of devices such as turning on a fan.
  - Twin's desired properties for long-running commands intended to put the device into a certain desired state. For example, set the telemetry send interval to 30 minutes.
  - Cloud-to-device messages for one-way notifications to the device app.

- In this lab you will learn how to send **cloud-to-device messages** from your **IoT Hub** to your **device** kit using the **Azure Portal**, and how to implement custom functions activated by specific messages (for example, to turn on/off an LED on the IoT Dk board)

Step 1: Send cloud-to-device message

1. In the Azure portal, go to your IoT Hub **Device Explorer**
2. Select you **Device**
3. Click on **Message to Device**

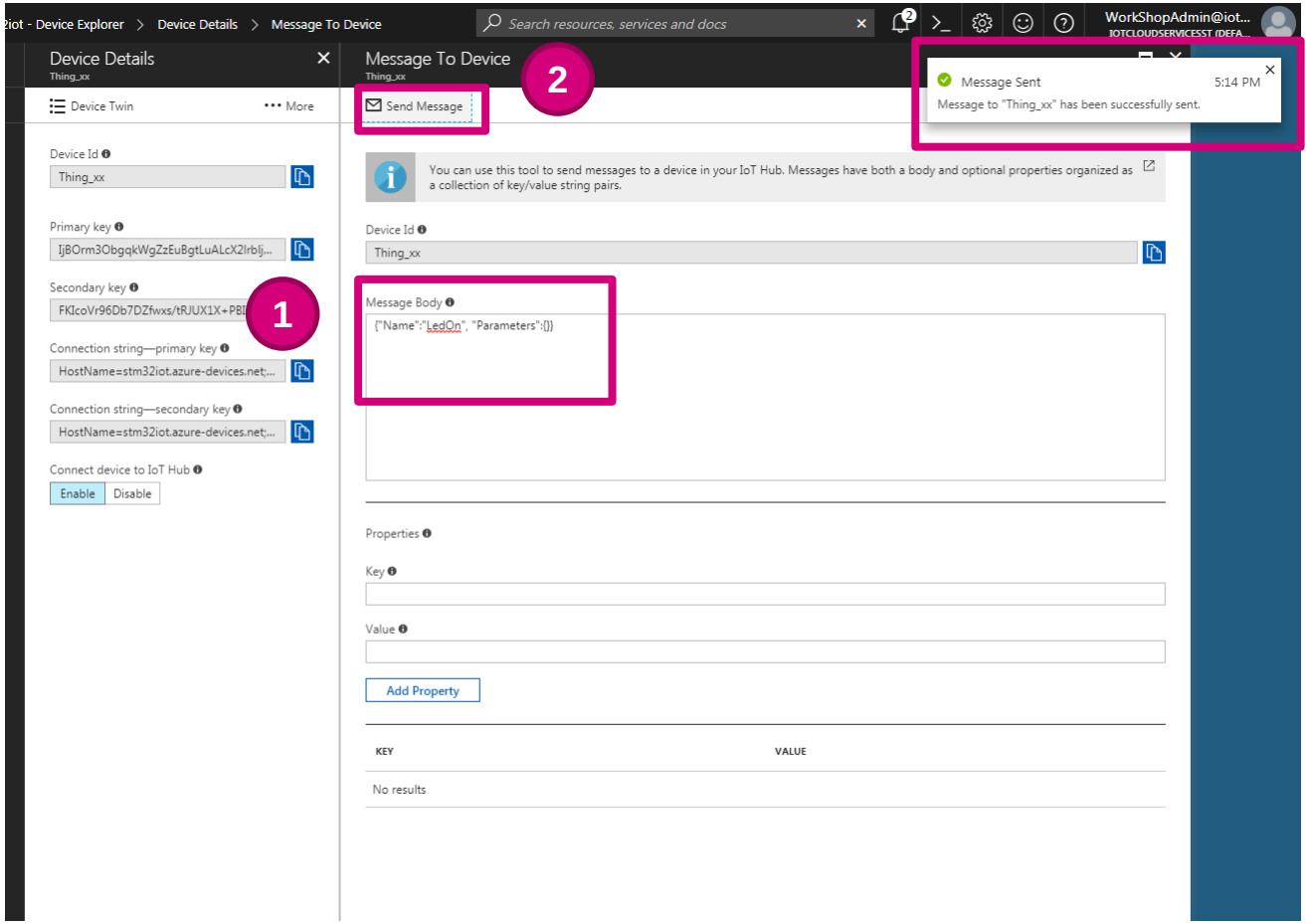


life.augmented

Step 1: Send cloud-to-device message

Turn LED On

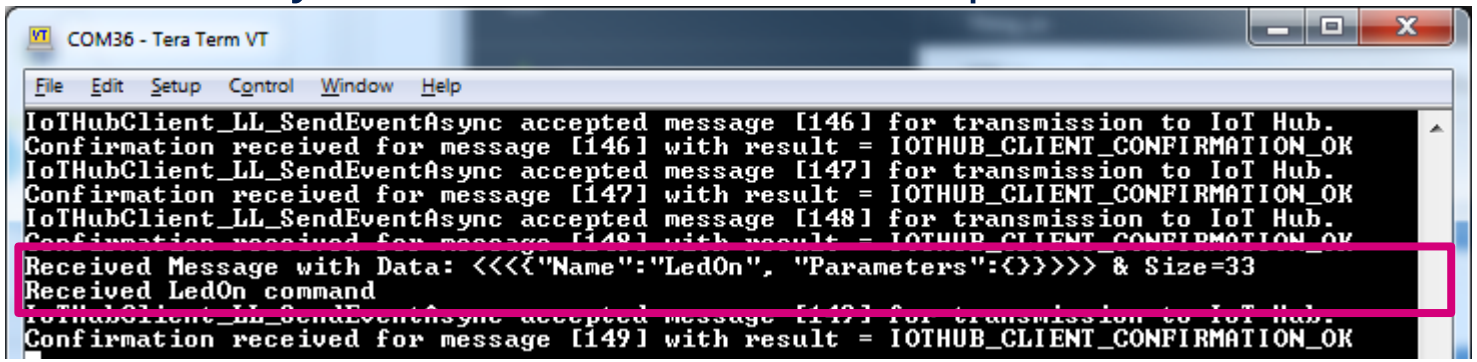
1. Copy-paste the message below to the Message Body field to turn the green LED on:
   - `{"Name":"LedOn", "Parameters":{}}`
2. Click on **Send**



life.augmented

Step 2: Visualize and Interpret Message received by device

- Each message received by the device from IoT Hub is printed over serial terminal



- Microsoft Azure IoT SDK provides means to define user specific callbacks (actions) associated to certain commands

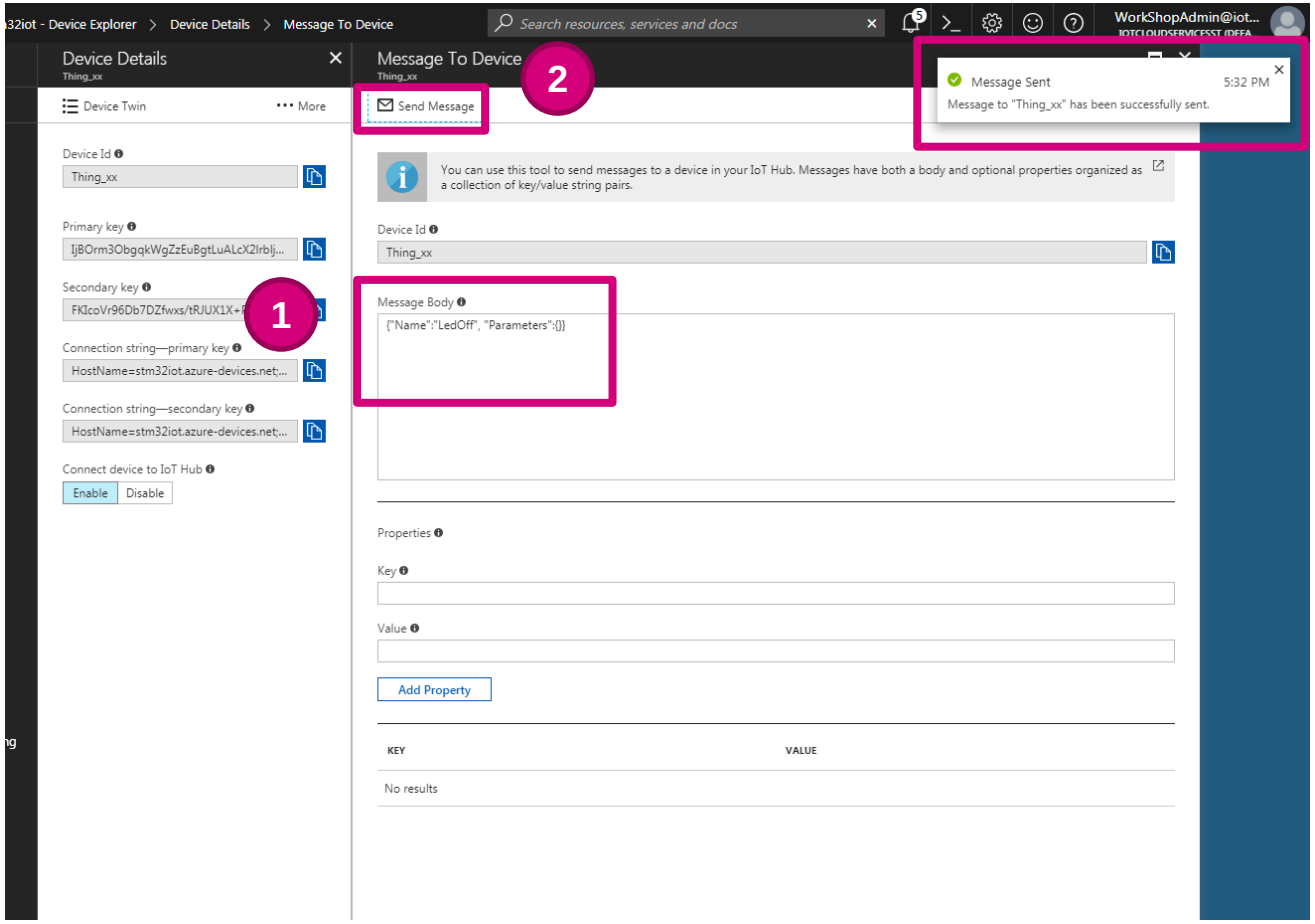
- By default the application can interpret the following messages:
  - **Pause** → pause the application
  - **Play** → resume the application
  - **LedOn/LedOff** → turn on/off the green LED on IoT Dk board
  - **LedBlink** → the green LED on IoT Dk board will blink for each message received

Step 3: Send cloud-to-device message

Turn LED Off

1. Copy-paste the message below to the Message Body field to turn the green LED off:
   - `{"Name":"LedOff", "Parameters":{}}`

2. Click on **Send**



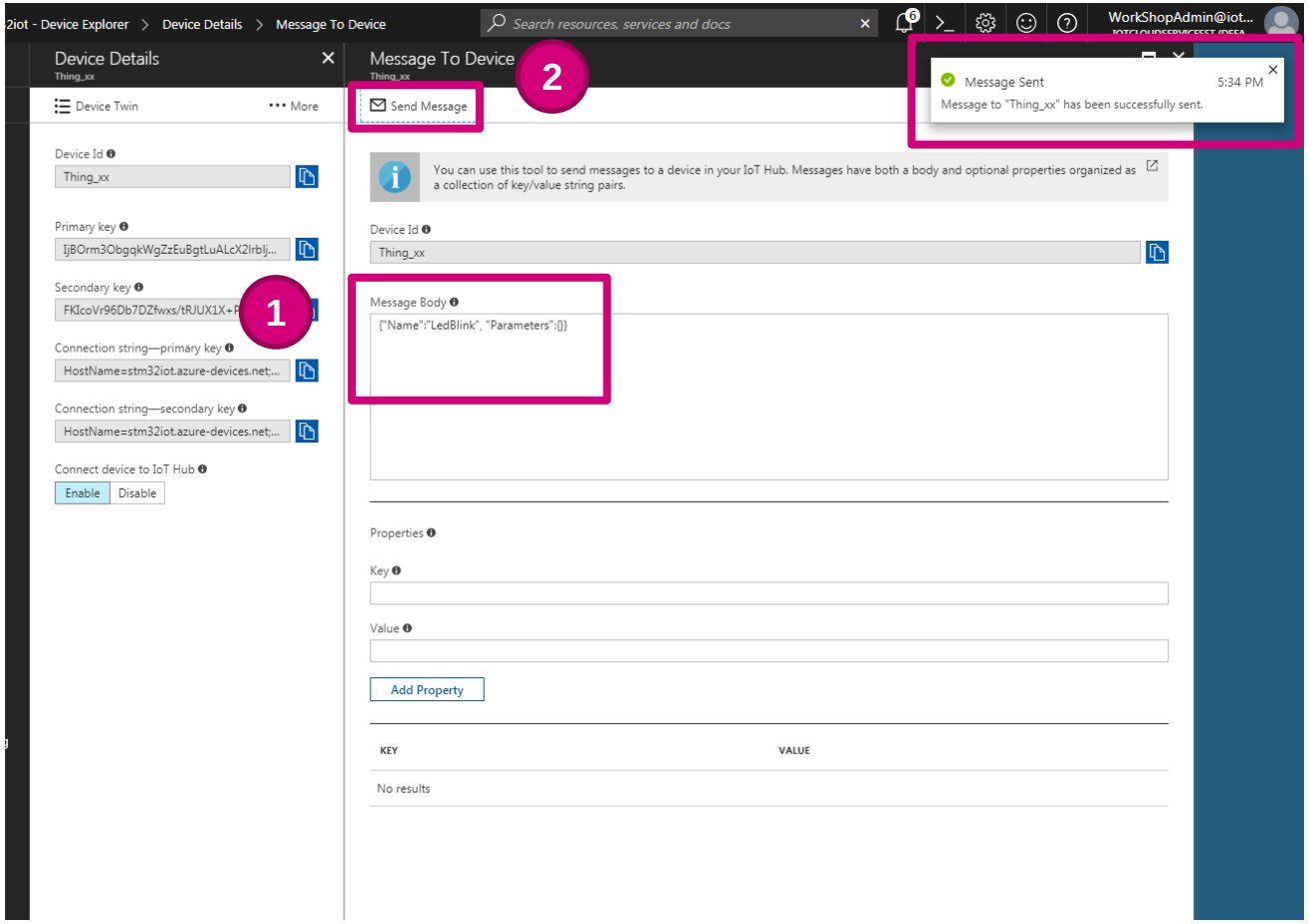
life.augmented

Step 4: Send cloud-to-device message

Blink LED

1. Copy-paste the message below to the Message Body field to blink the green LED:
   - `{"Name":"LedBlink", "Parameters":{}}`

2. Click on **Send**



life.augmented

Customize Code to Execute Commands

- You can define a new action associated to a message, see example in `AzureClient_mqtt_DM_TM.c`

```
WITH_ACTION(Pause),
```

Pause is the message and the name of the callback (action) that will be executed

- Define user specific implementation for the message. See example for Pause in `AzureClient_mqtt_DM_TM.c`

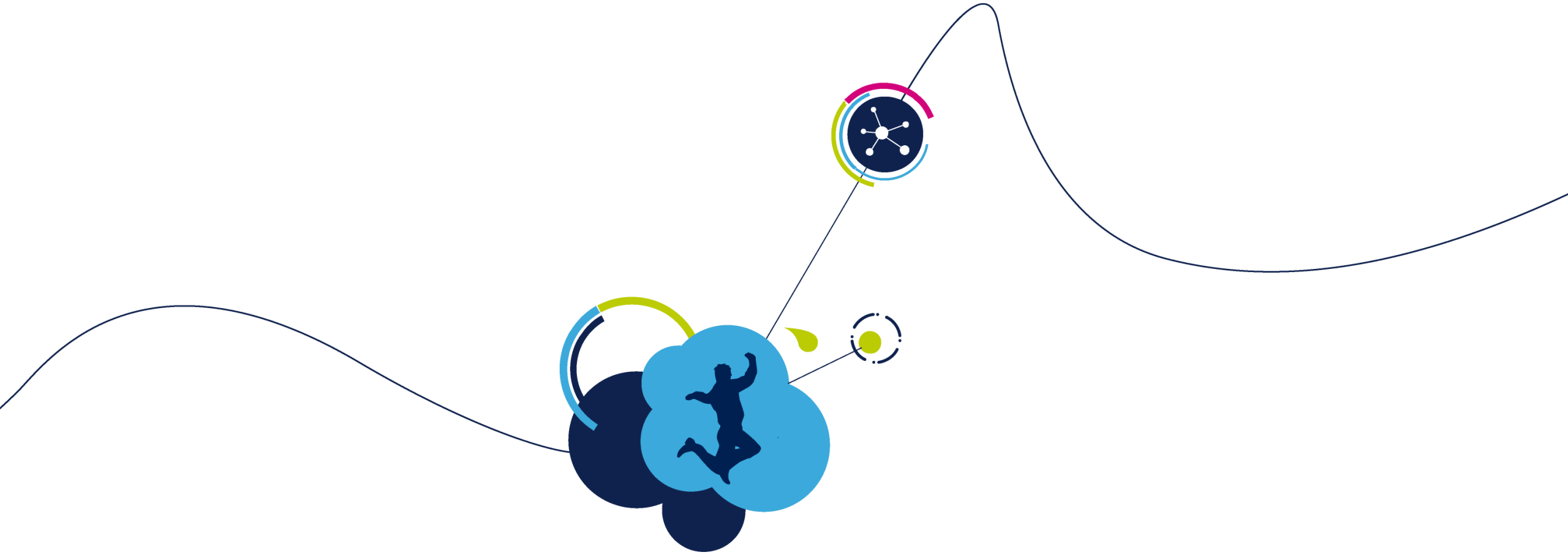
```
EXECUTE_COMMAND_RESULT Pause(Azure1_t *Azure1)
{
  (void)(Azure1);
  AZURE_PRINTF("Received Pause command\r\n");
  StopTimer1();
  return EXECUTE_COMMAND_SUCCESS;
}
```

- Function `ReceiveMessageCallback()` will be executed each time a message is received by the device. This function will take care of triggering the user specific action

```
AZURE_PRINTF("Received Message with Data: <<<%.*s>>> & Size=%d\r\n", (int)size, buffer, (int)size);

(void)memcpy(temp, buffer, size);
temp[size] = '\0';
executeCommandResult = EXECUTE_COMMAND(userContextCallback, temp);
```

Trigger callback associated to message received



Lab 6: Firmware-over-the-air (FOTA) Update

life.augmented

Introduction

- In this lab, we will:
  - Trigger a Firmware-Over-The-Air (FOTA) Update using a simple binary stored on the cloud.

Over-The-Air Firmware Update

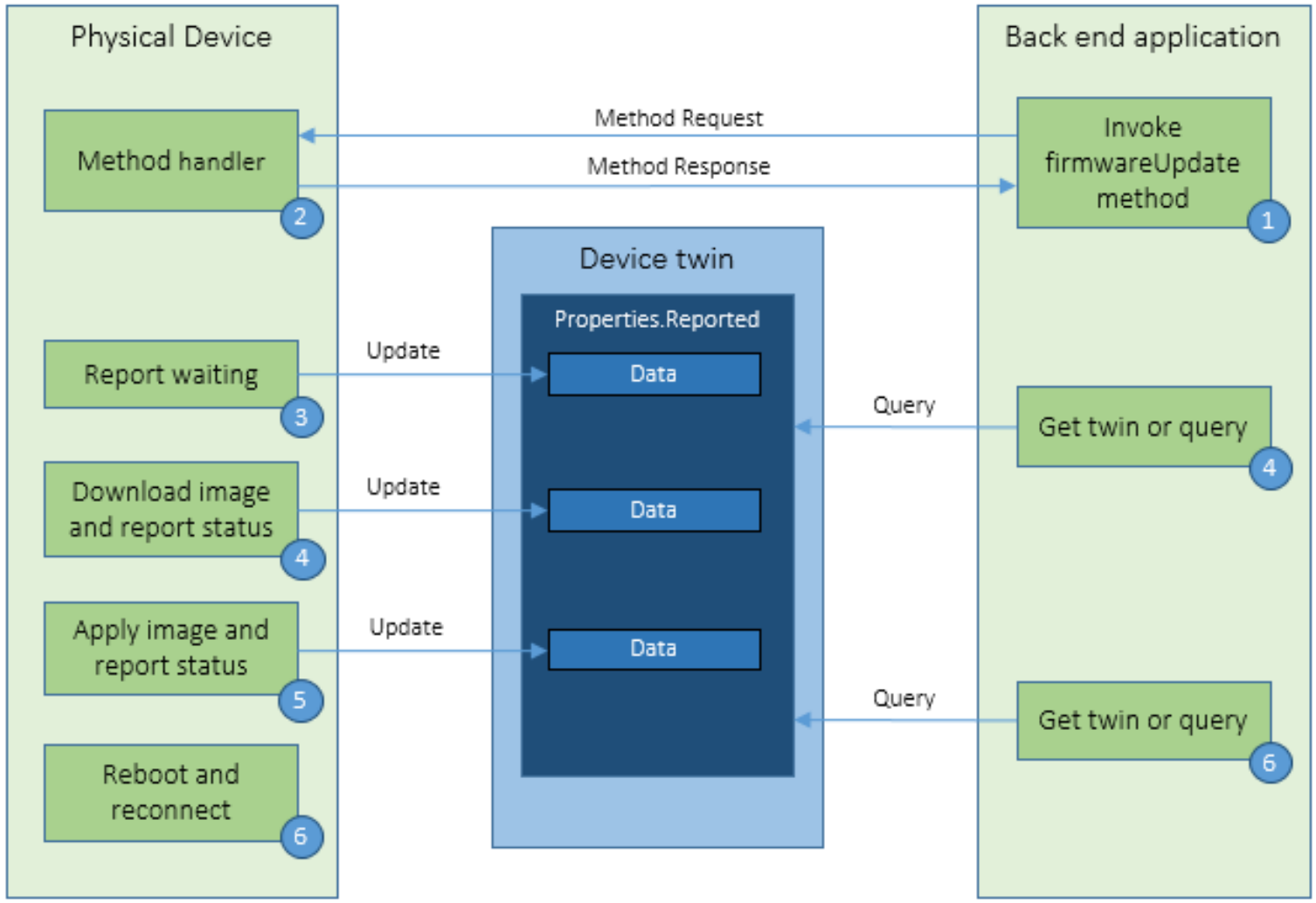
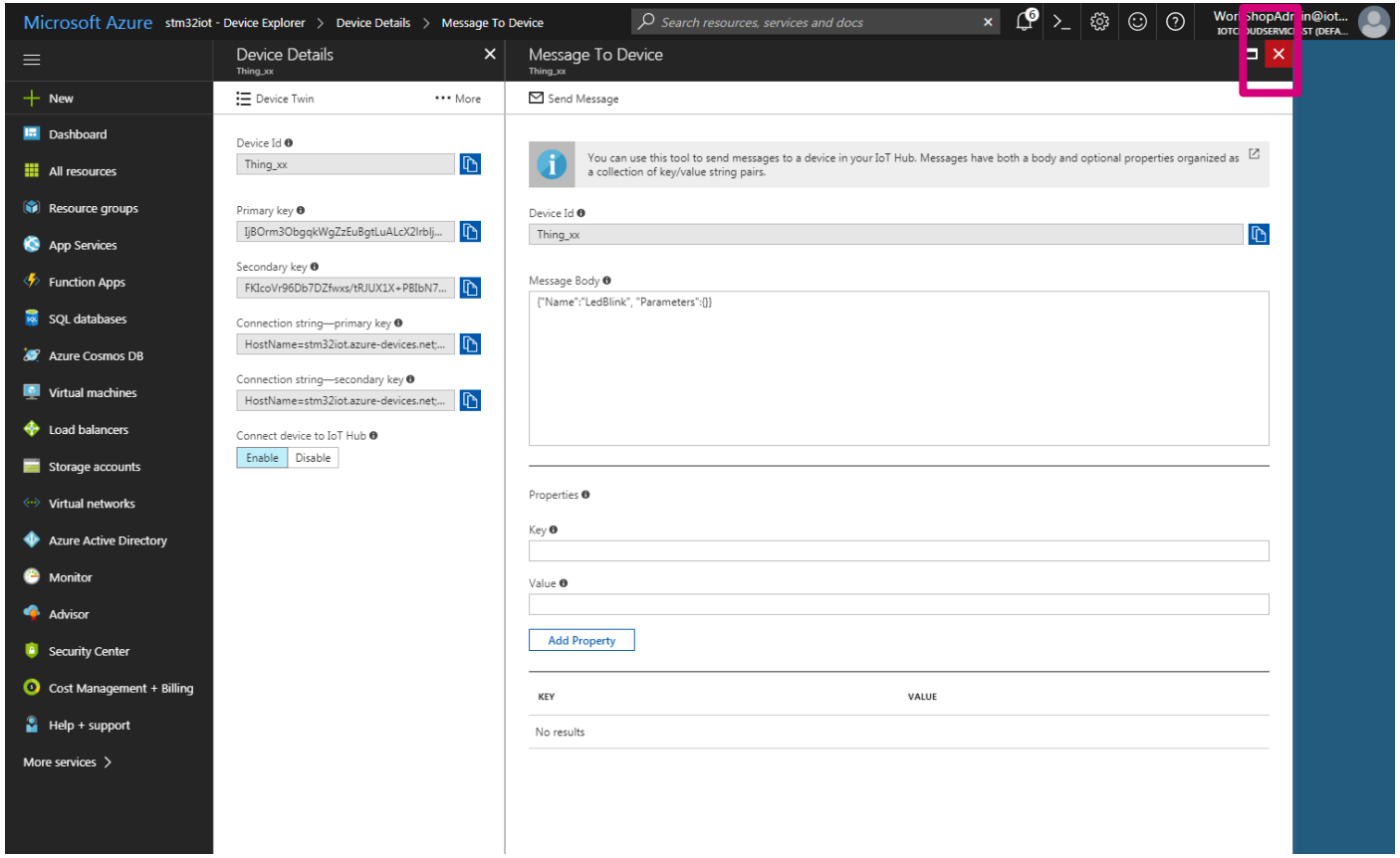


Image Source: Microsoft

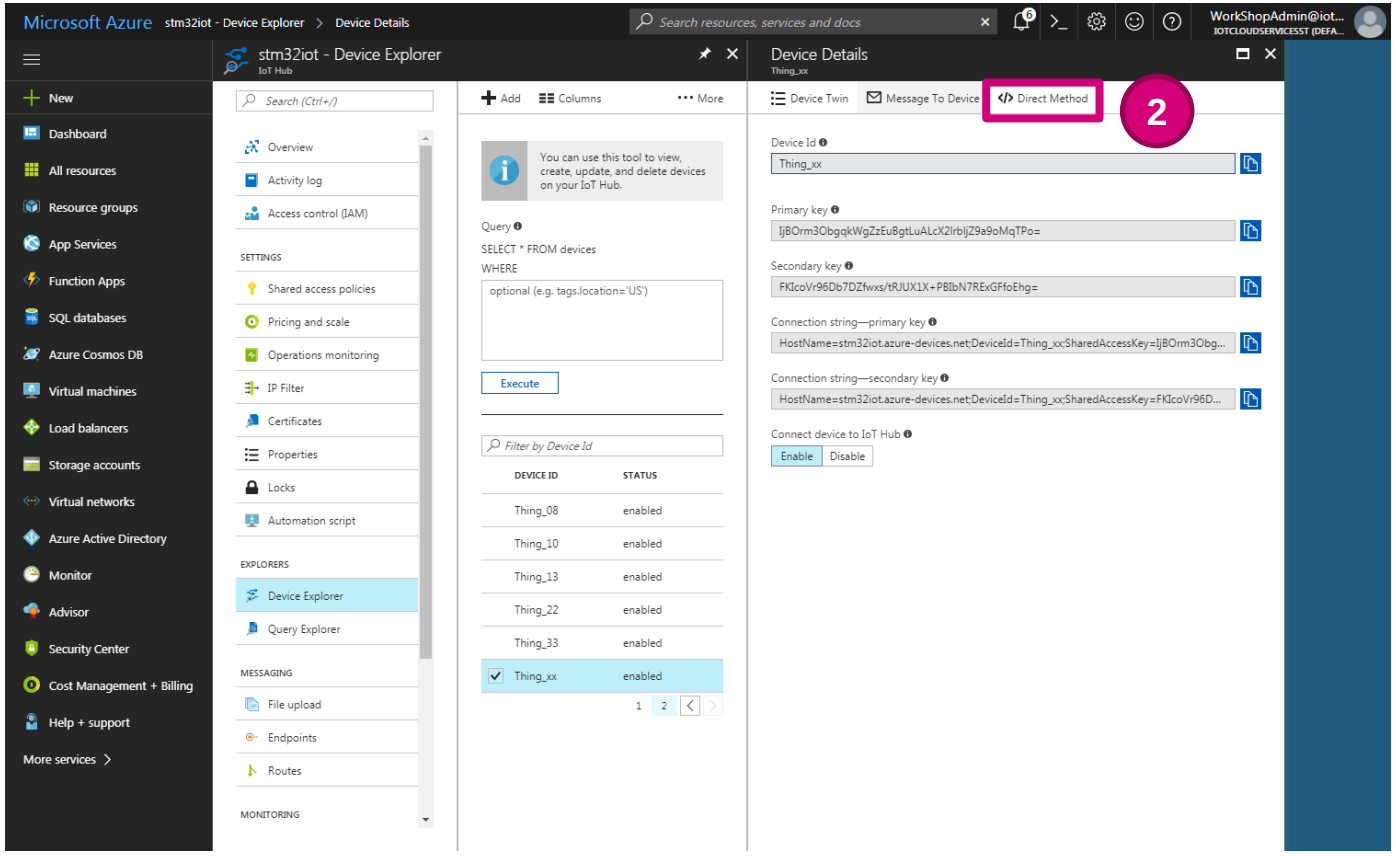
Step 2: Initiate Firmware Update method

1. Close the **Message to Device** window



Step 2: Initiate Firmware Update method

2 Click on **Direct Method**



life.augmented

Step 2: Initiate Firmware Update method

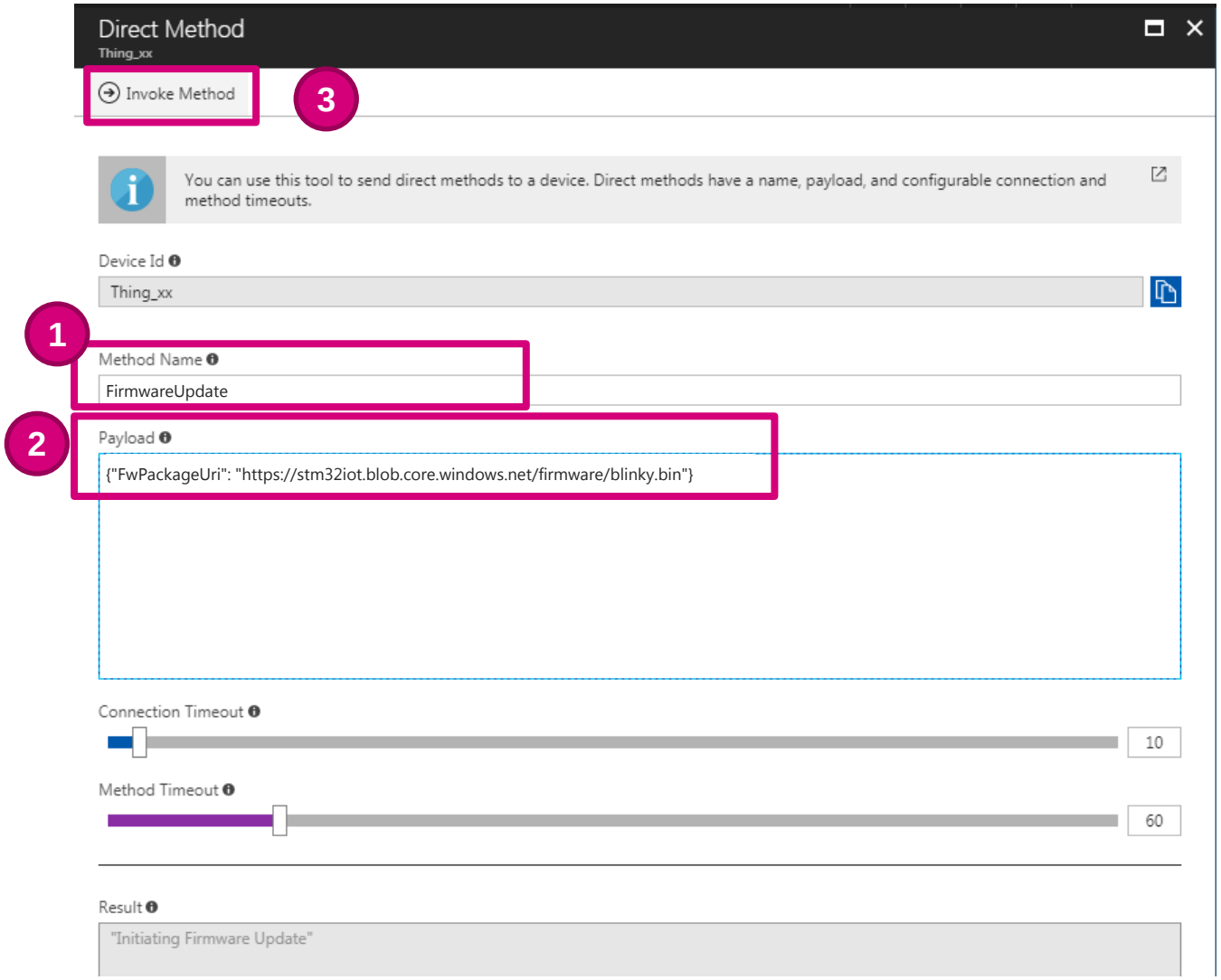
1. Copy-paste the message below to Method name

```
FirmwareUpdate
```

2. Copy-paste the message below to Method payload

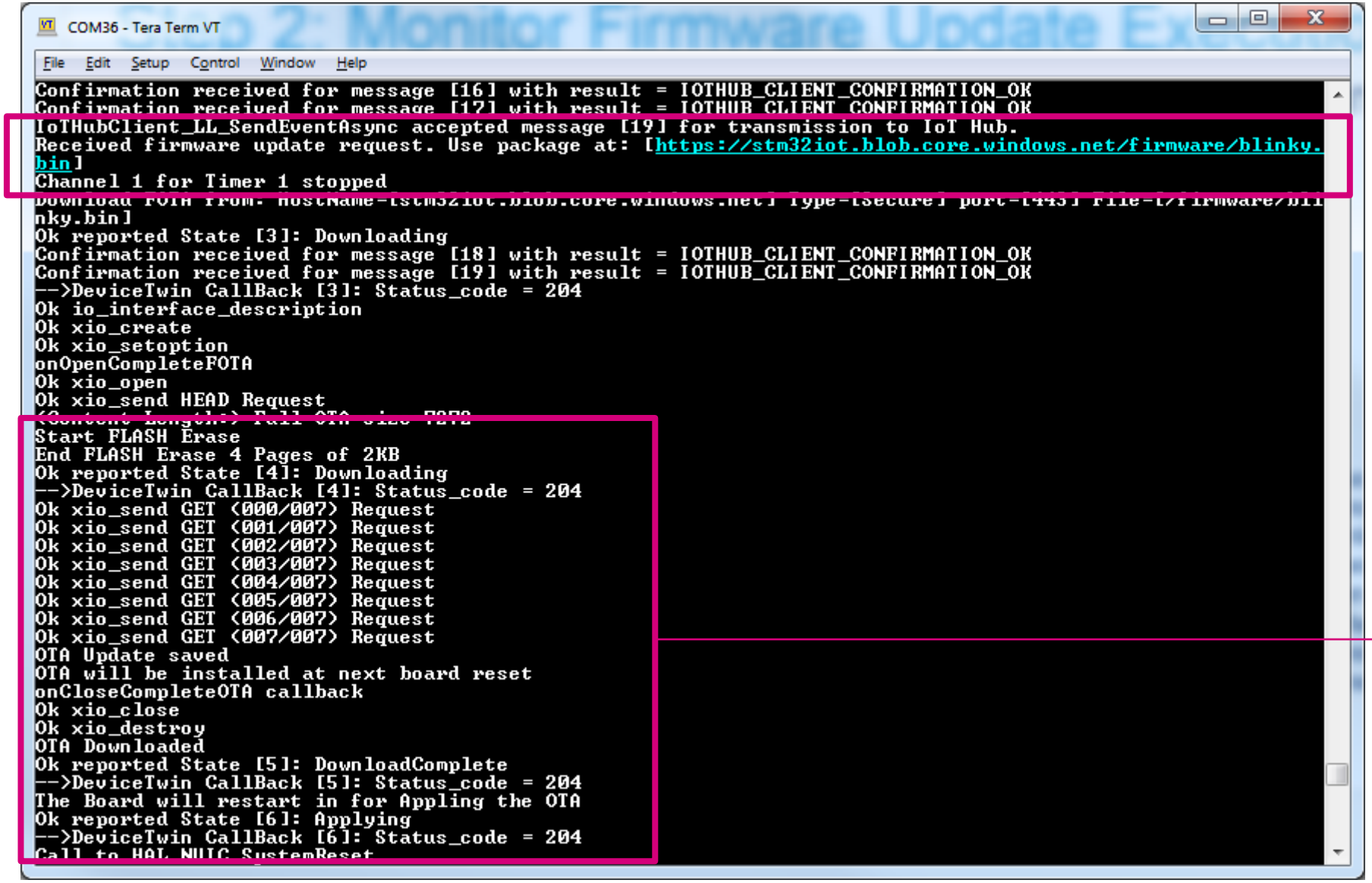
```
{"FwPackageUri":
"https://stm32iot.blob.core.windows.net/
firmware/blinky.bin"}
```

3. Click on Invoke Method



Step 2: Monitor Firmware Update Execution

on Device Via Serial Terminal

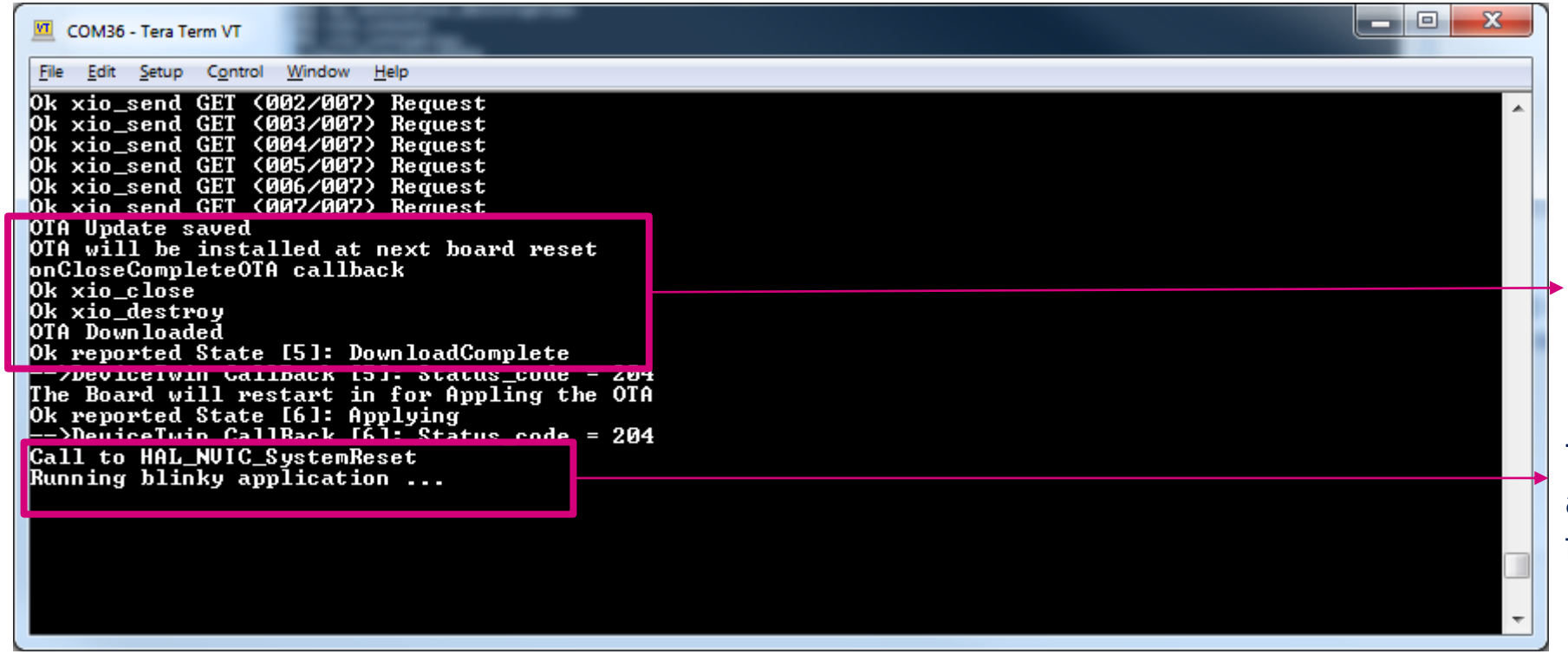


Erase FLASH memory and download new firmware
When download is finished, the board will restart and load the new firmware

Step 2: Monitor Firmware Update Execution

on Device Via Serial Terminal (Cont'd)

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

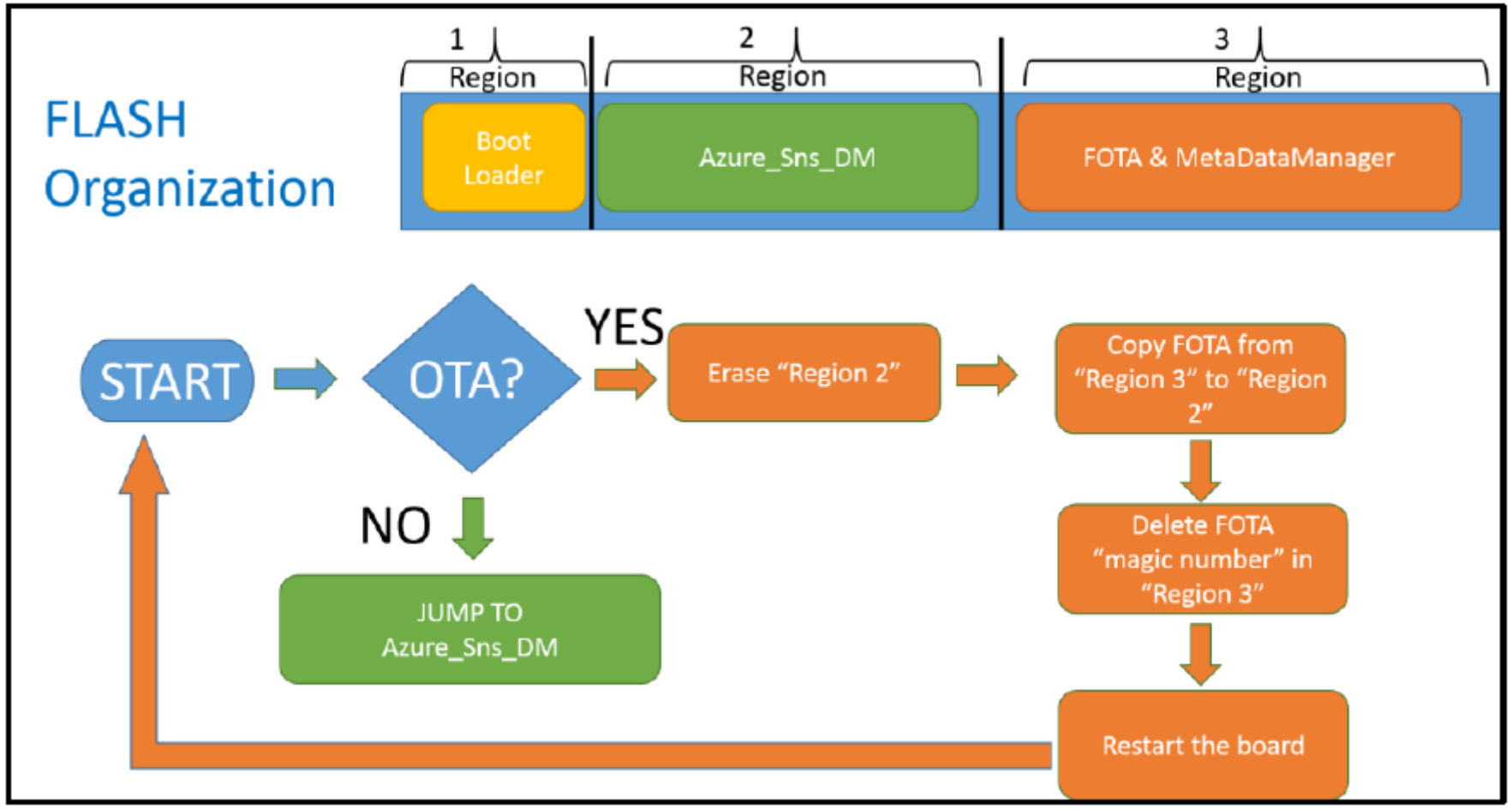


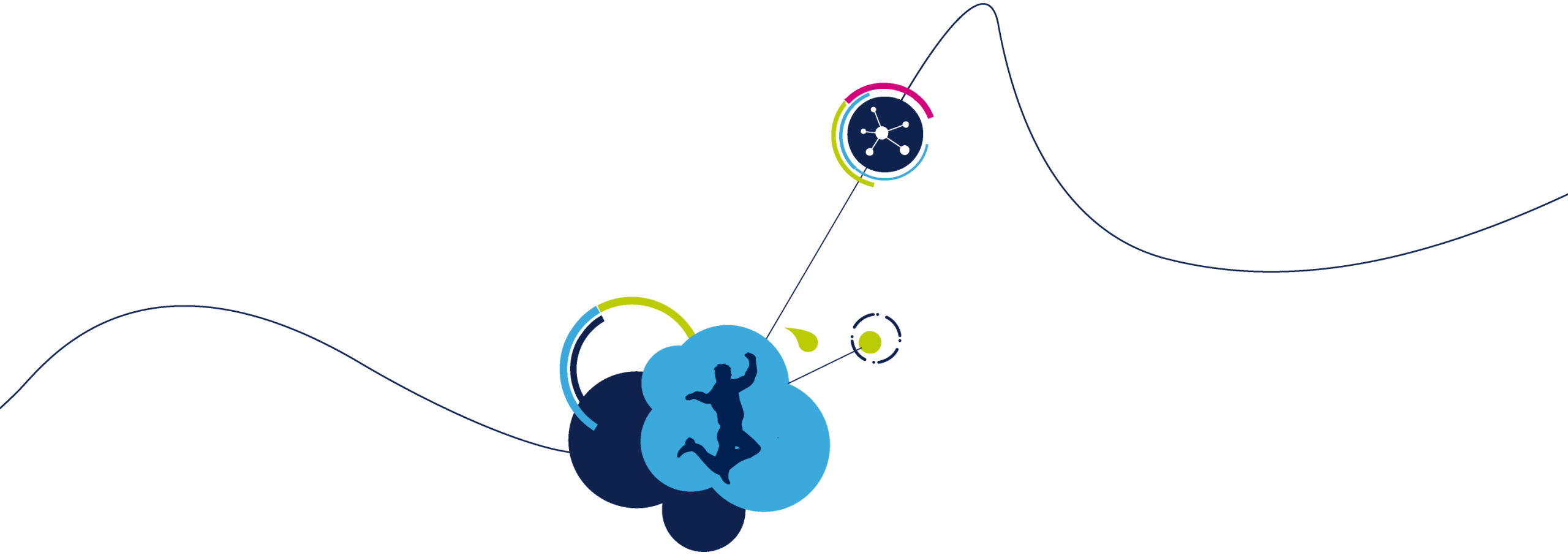
Download is finished

The board restarts and loads the new firmware

STM32 boot sequence

after a Firmware Update





Bluetooth® Low Energy Overview

life.augmented

What is Bluetooth® Low Energy?

- Bluetooth® Low Energy technology
  - Short range wireless ISM 2.4 GHz
  - Optimized for ultra low power
    - <15 mA peak current
    - <50 uA average current
  - Fast connection procedure
  - Client server architecture
  - Low data throughput application

  - Security including privacy/authentication/authorization
    - Based on encryption AES128
  - Master Role : Central Device (Scanning, Initiating Connection)
  - Slave Role : Peripheral Device (Advertising)

Bluetooth® Low Energy Branding

2011 Two flavors

Bluetooth® SMART

- Ultra low power consumption being a pure low energy implementation
- Months to years of lifetime on a standard coin cell battery

Bluetooth® SMART READY

- Classic Bluetooth® + Bluetooth® low energy on a single chip
- These are the hub devices of the Bluetooth® ecosystem

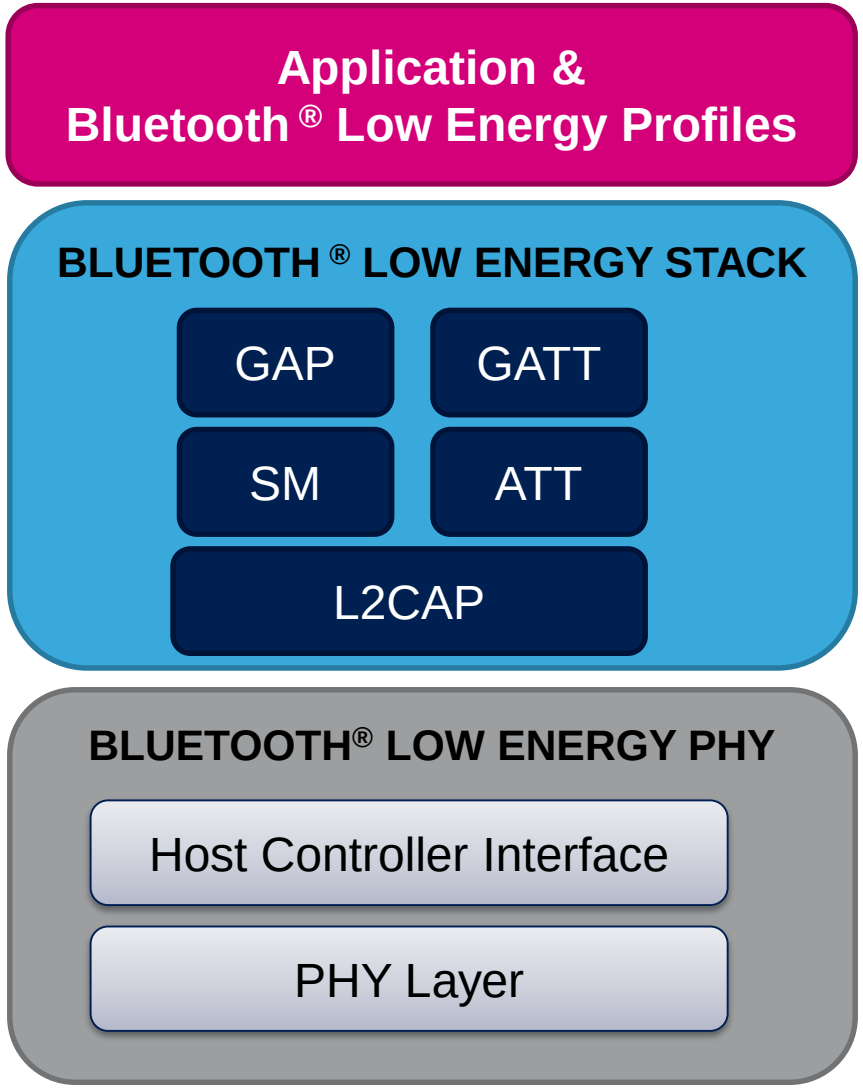
2017 Back to one flavor

Bluetooth®

- An implementation of the Bluetooth® core system has only one Primary Controller which may be one of the following configurations:
  - BR/EDR Controller (3.0 and earlier)
  - LE (low energy) Controller (4.0 and newer)
  - Combined BR/EDR Controller portion and LE controller portion into a single Controller (4.0 and newer)

Source: Bluetooth® SIG

Bluetooth® Low Energy Stack Partitioning



- The application collects & computes the data to be transmitted over Bluetooth® Low Energy.
- Data transmission and reception uses the Bluetooth® Low Energy stack services and characteristics capabilities thanks to standard or proprietary application profile.
- All low energy communication occurs within the **G**eneric **A**ttribute **P**rofile **(GAP)**. PHY layer insures transmission over the air

ST BlueNRG-MS Solution

Integration

Flexibility

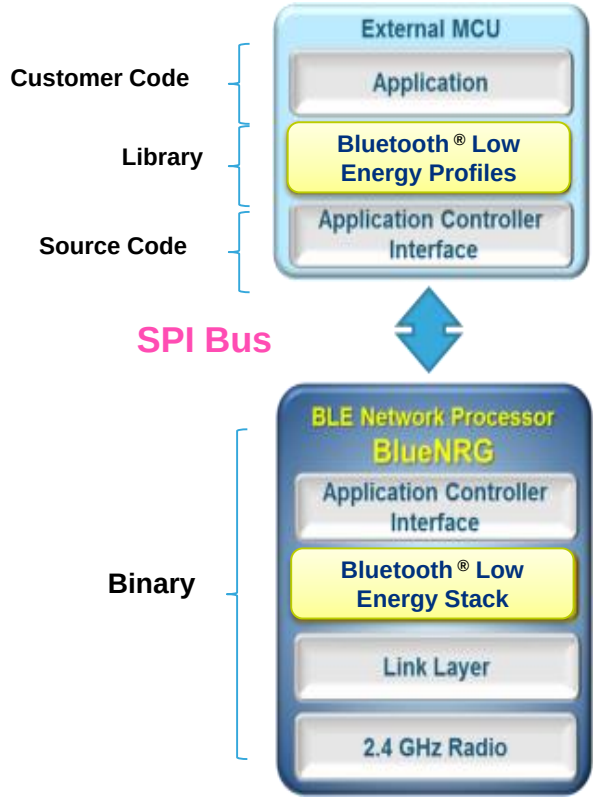
Low power

Best In Class

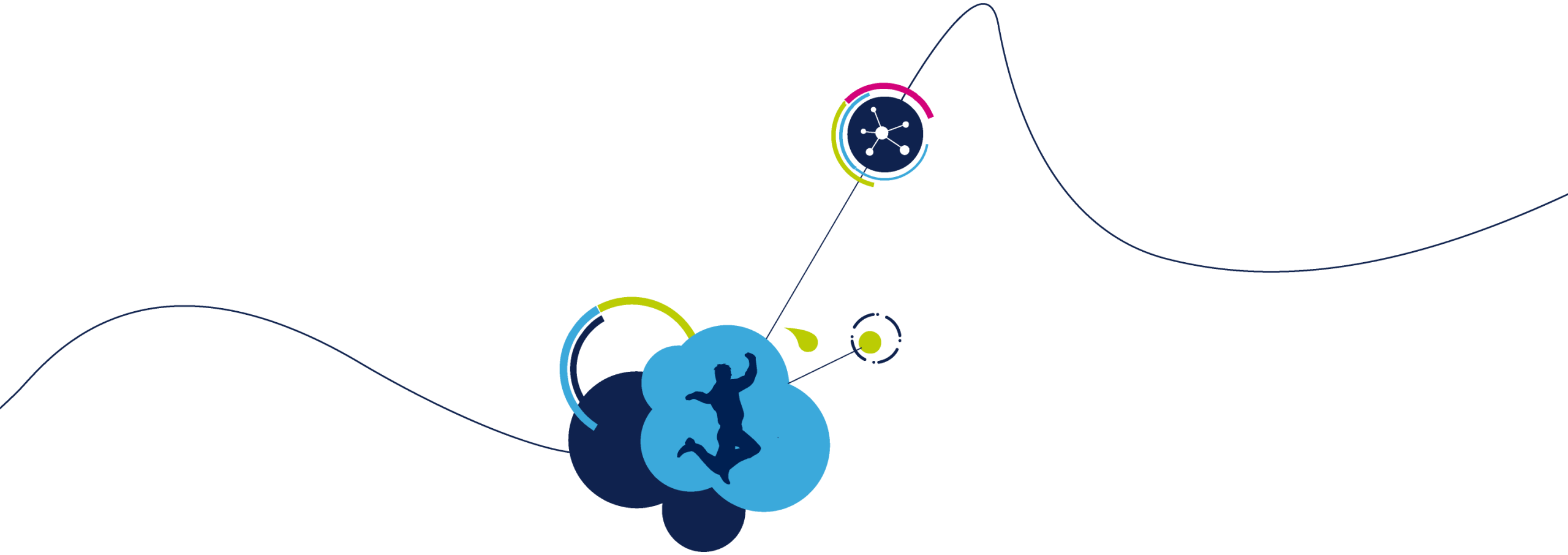
Small size

- **Single mode Bluetooth® Low Energy wireless network processor**
- 2.4GHz RF transceiver
- Cortex-M0 microcontroller (running the BT MS stack)
- AES 128-bit co-processor
- Master and Slave Mode Bluetooth® Low Energy (4.1) Network Processor.
- On chip non-volatile Flash memory allows OTA stack upgrade.
- I_{CC}RX 7.3mA
- I_{CC}TX 8.2mA @ 0 dBm
- I_{CC}Sleep 1.7µA
- I_{CC}Shutdown 2.5nA

- **+ STM32 Consumption & Size**



life.augmented



Demo: Bluetooth® Low Energy Pairing

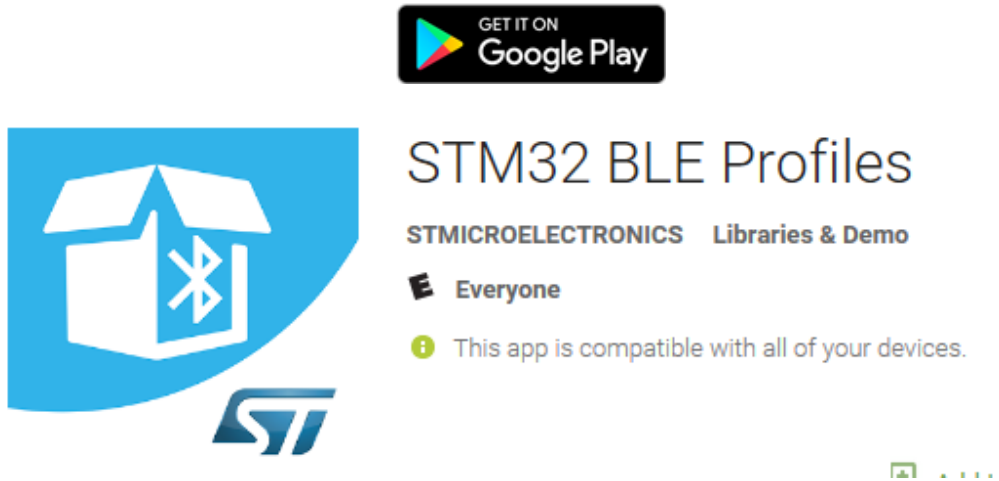
life.augmented

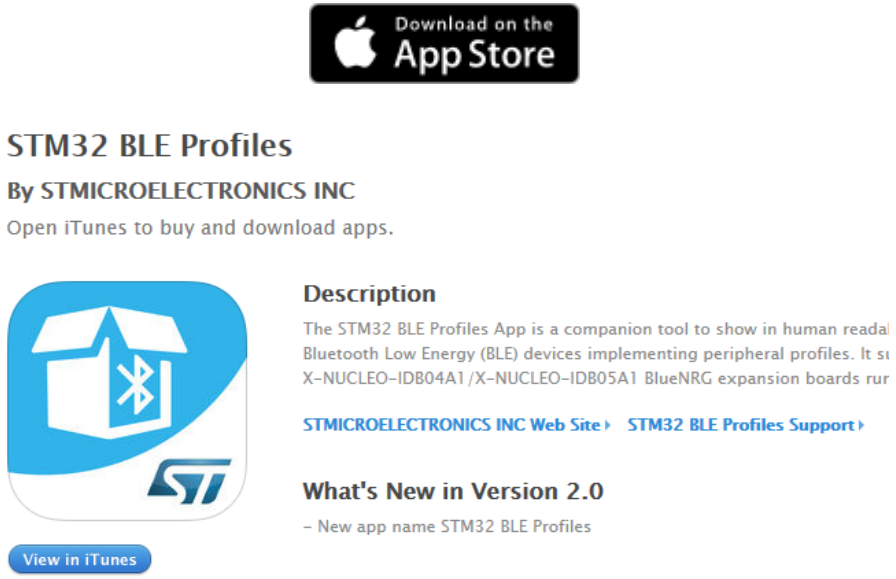
Lab Goal

- This lab will ensure that your BlueNRG device has a unique name and MAC address.
- This lab will demonstrate how to configure a BlueNRG device, communicate with a smartphone and display heart rate data.

Lab Goal

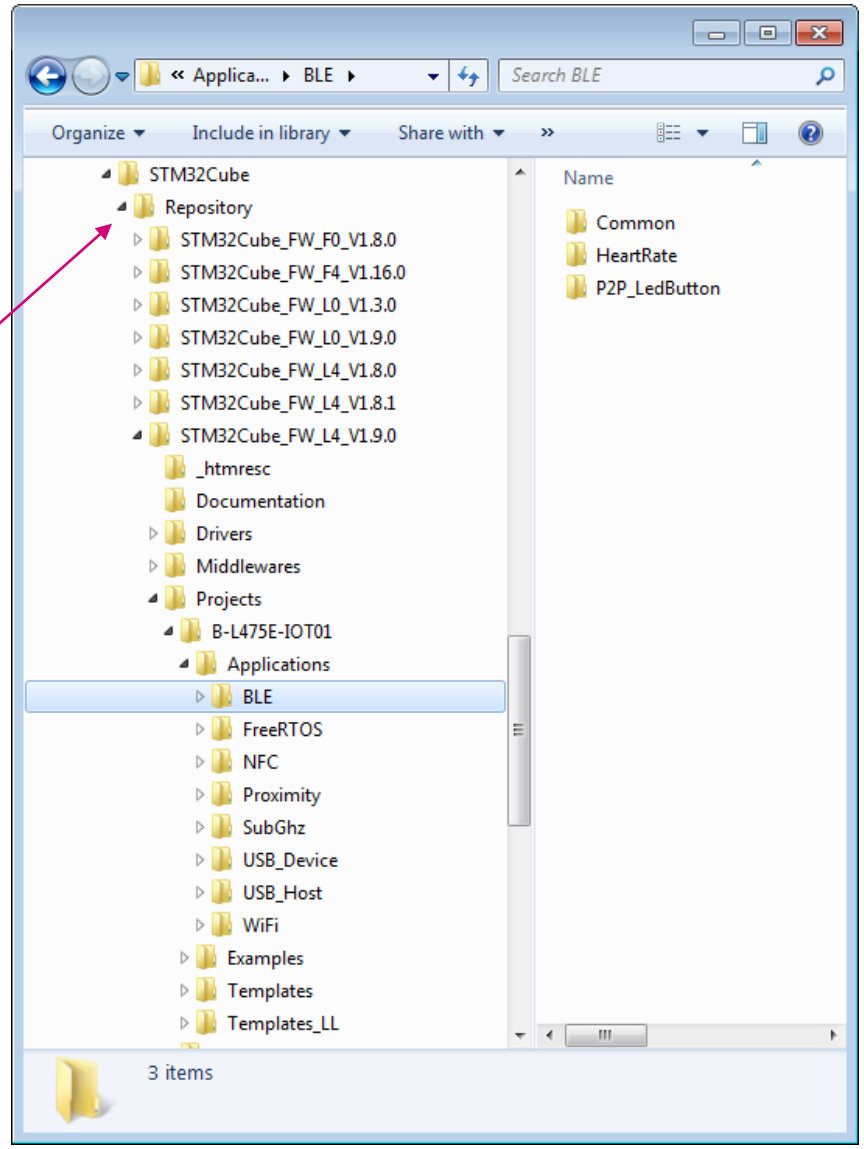
- The DK IoT Node will be used as server while the applet is a client.
- You will need to download the **STM32 BLE Profiles** application available on the Apple App store or the Android Google Play store.

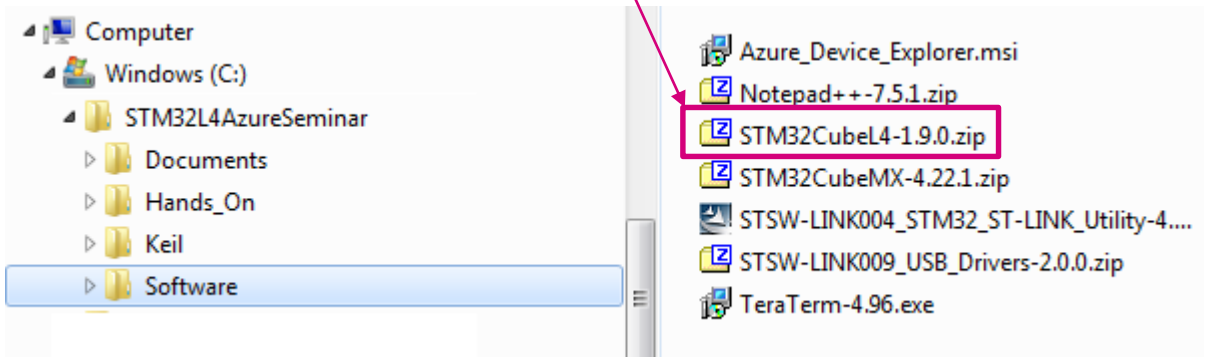




STM32CubeL4 Examples Package

- One of the packages installed on your PC is the STM32CubeL4
- This package includes several applications made for the STM32L4 DK IoT node board
- The zip file is extracted within your STM32Cube repository folder
- The zip file is installed by the Seminar Installer at C:\STM32L4AzureSeminar\Software\





life.augmented

Open the BlueNRG_HandsOn Project

- We are going to configure the **`BlueNRG_HandsOn`** program to give each BlueNRG module a unique MAC address and Unique device name. The device name will be used later to identify your board within the **ST BLE Profiles** app.

1. Close the previous Keil MDK-ARM project.
2. Double click on the **`Project.uvprojx`** file located here:

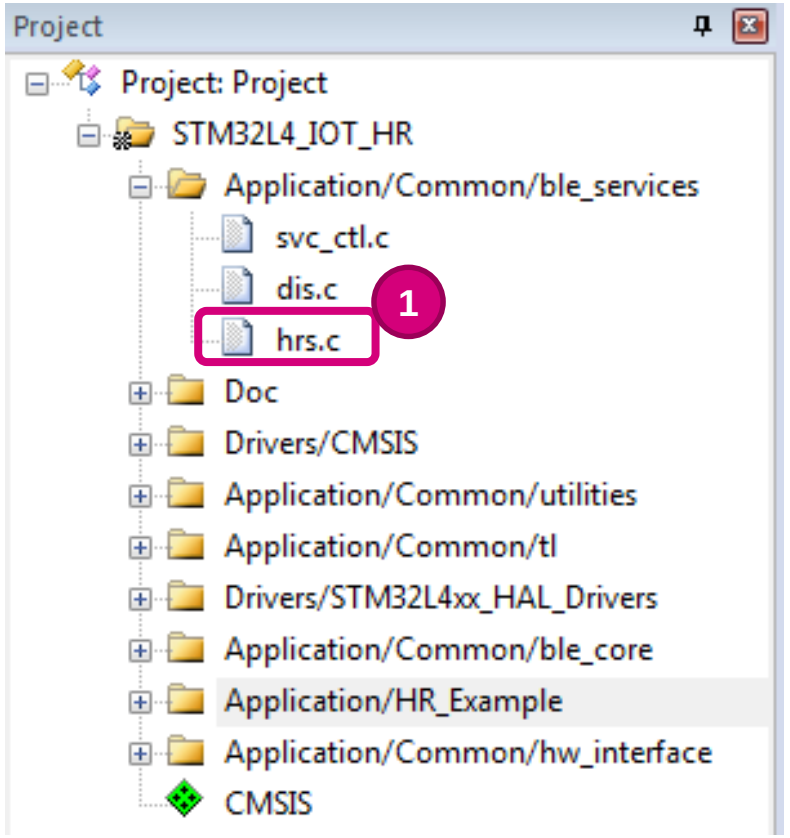
`…\STM32Cube\Repository\STM32Cube_FW_L4_V1.9.0\Projects\B-L475E-IOT01\Applications\BLE\HeartRate\MDK-ARM\`

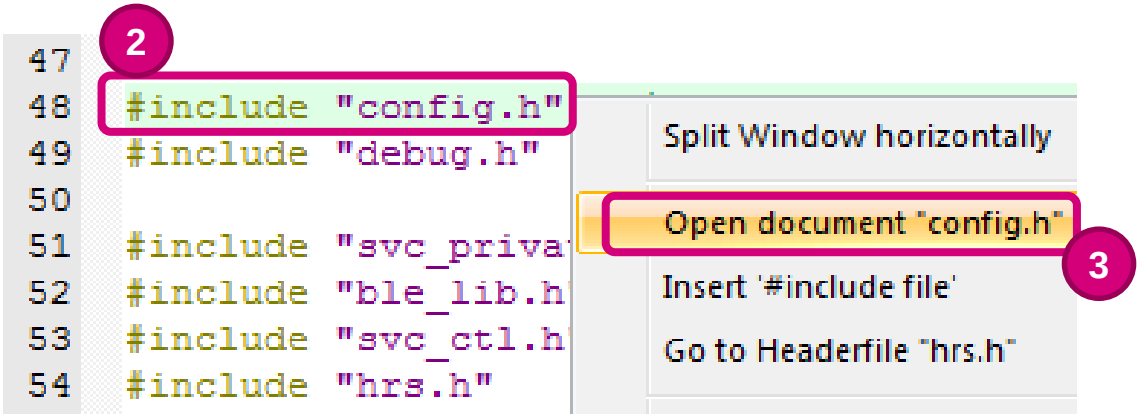
Note: Your STM32Cube Repository folder path can be found by launching STM32CubeMX, and checking Help > Updater Settings

BlueNRG Module configuration (1/4)

Expand the file tree:

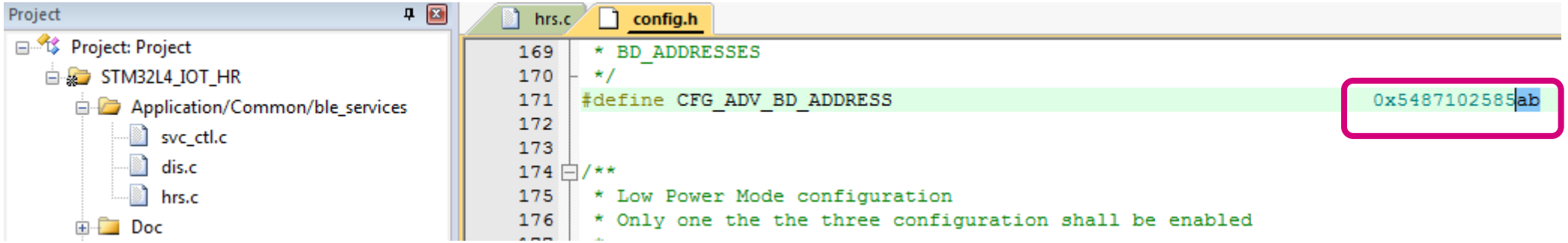
1. Expand **Application/Common/ble_services**
2. Double-click **hrs.c**
3. Right-click **config.h** on line 48
4. Select **Open document “config.h”**





life.augmented

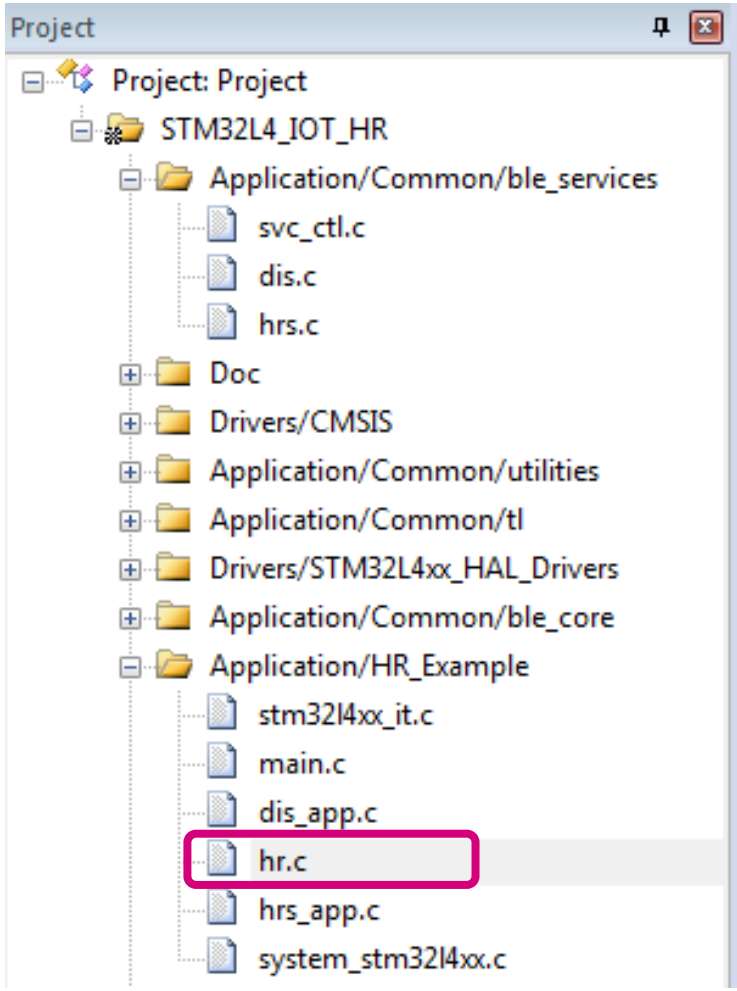
BlueNRG Module configuration (2/4)



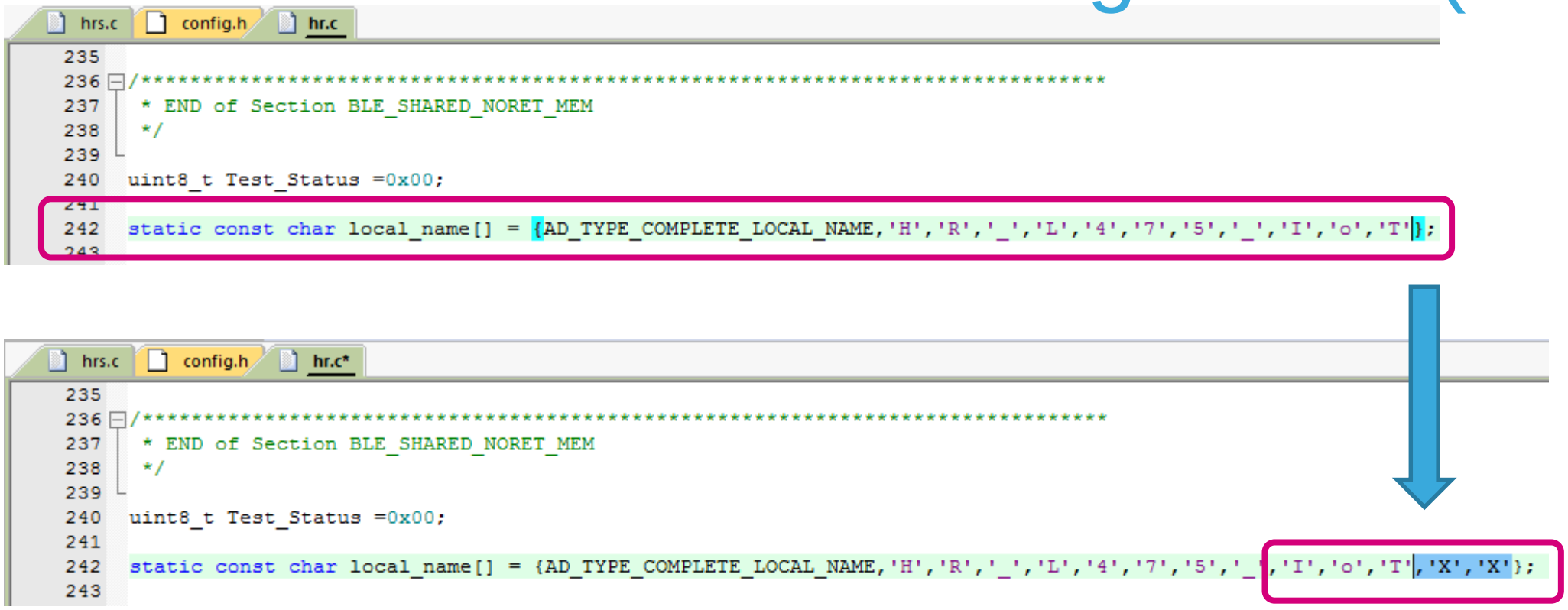
- Replace the '**ab**' in the **CFG_ADV_BD_ADDRESS** (line 171) with your participant number found on your box (2 characters).

BlueNRG Module configuration (3/4)

1. Expand **Application/HR_Example**
2. Double-click **hr.c**



BlueNRG Module configuration (4/4)



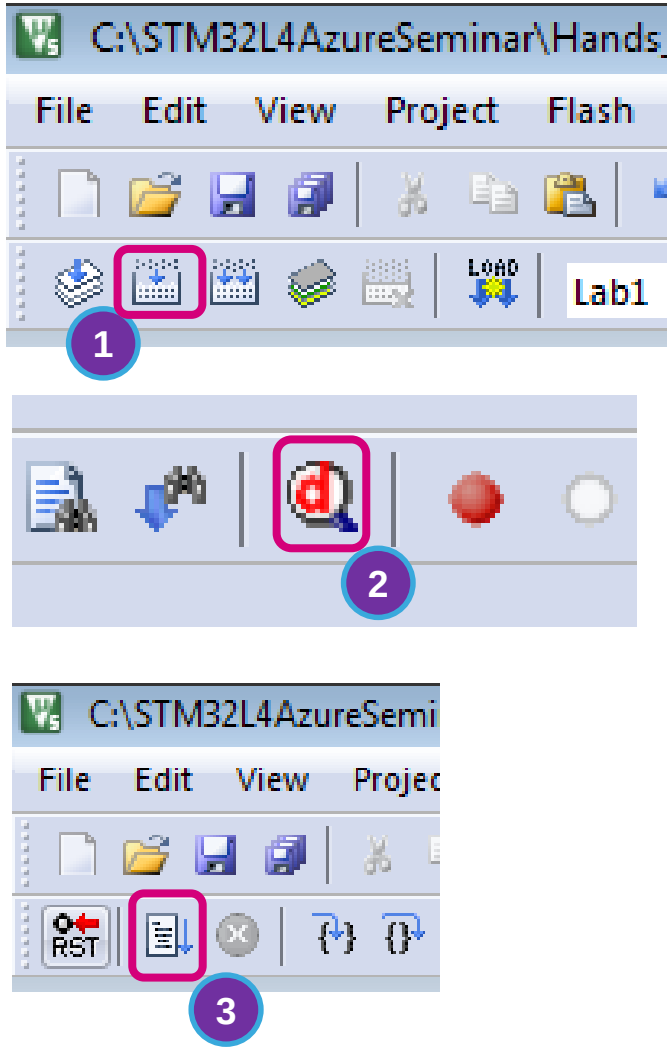
- Add your participant number within single quotes in place of the **'X','X'** in the **local_name** (line 242) table as shown above.

Load and Run

1. Click on the Build button or choose Project > Build Target

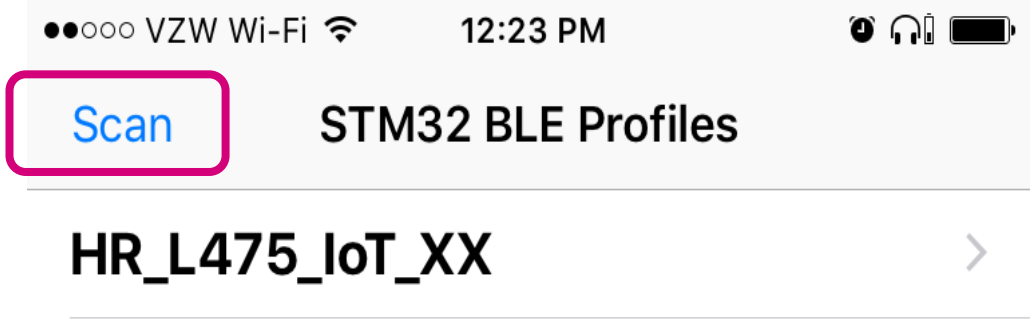
2. Click the Start/Stop Debug Session button

3. Click the Run button

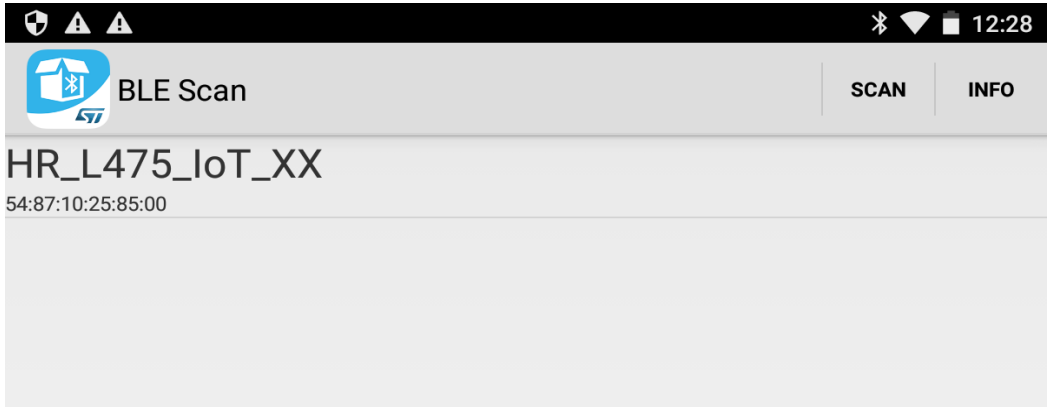


Pair with **STM32 BLE Profiles** App

iOS



Android

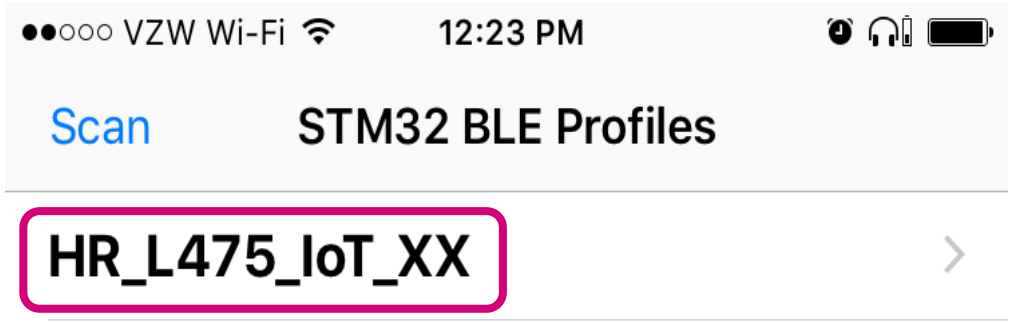


- Make sure Bluetooth® is active on your phone
- Using your phone, open the **STM32 BLE Profiles** app.
- For iOS users click on **Scan**.

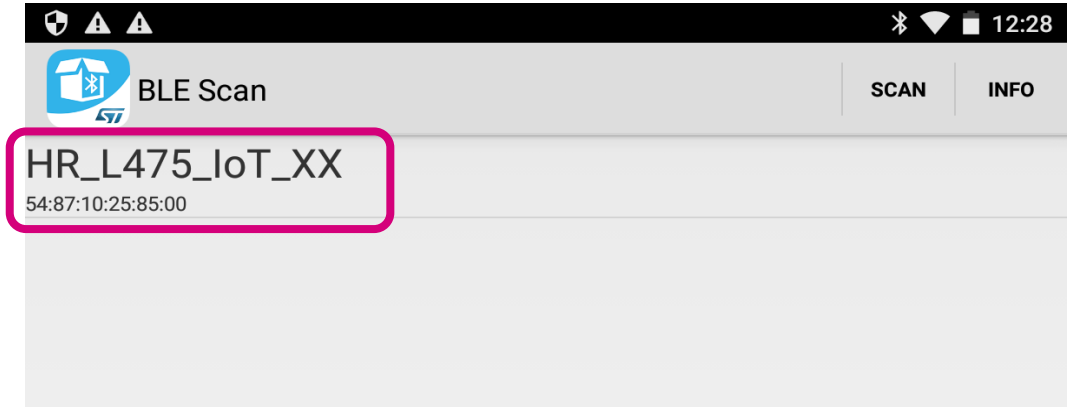
life.augmented

Pair with **STM32 BLE Profiles** App

iOS



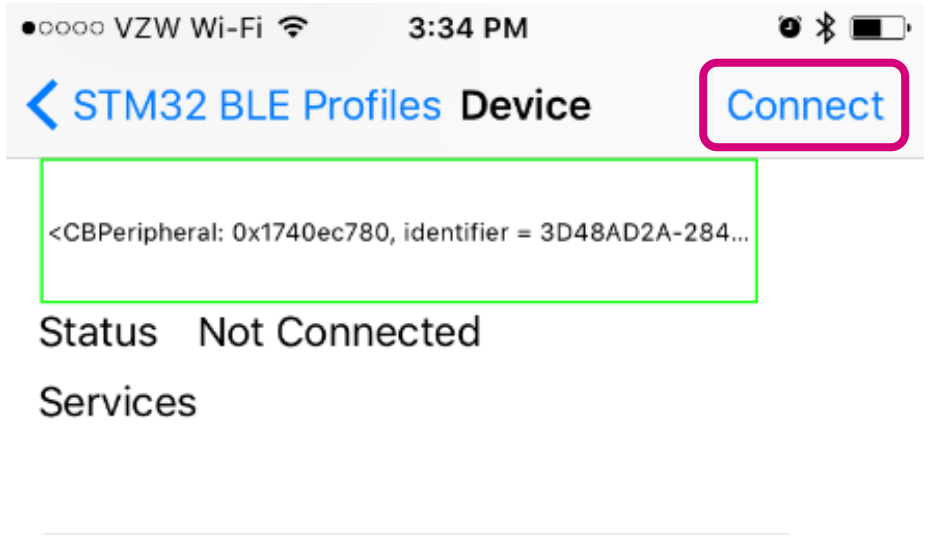
Android



- Identify your device with device name **HR_L475_IoT_XX**, (**XX** is the number you entered during the board configuration. Click on your device name.

Connect to your Device

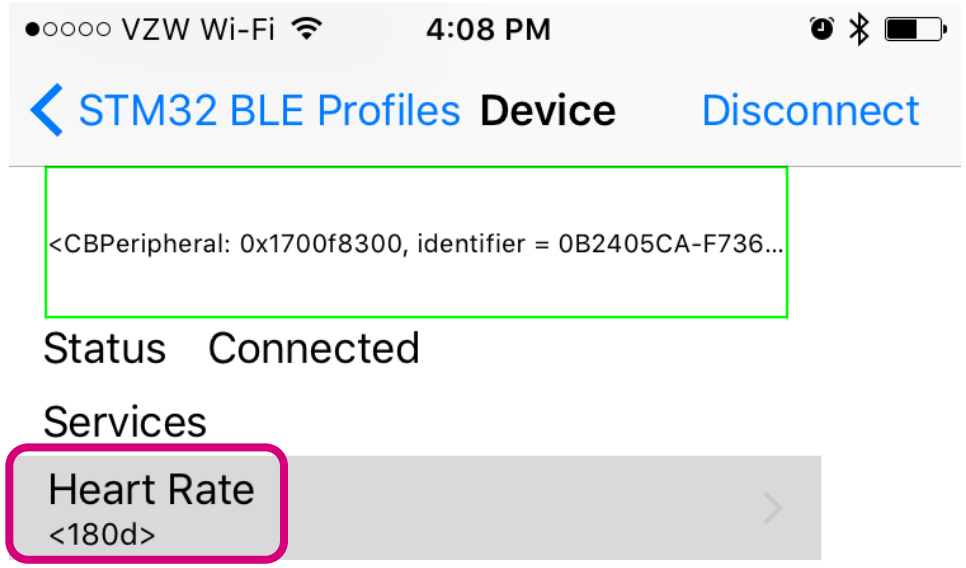
iOS



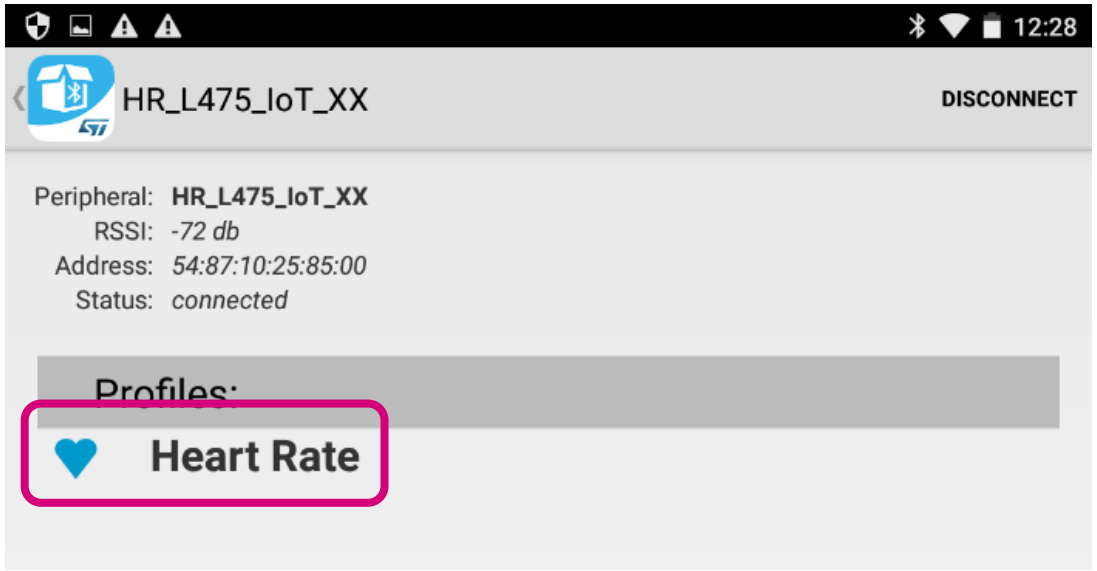
- iOS users, please click **Connect** on the next screen (iOS)

Select the Heart Rate Profile

iOS



Android

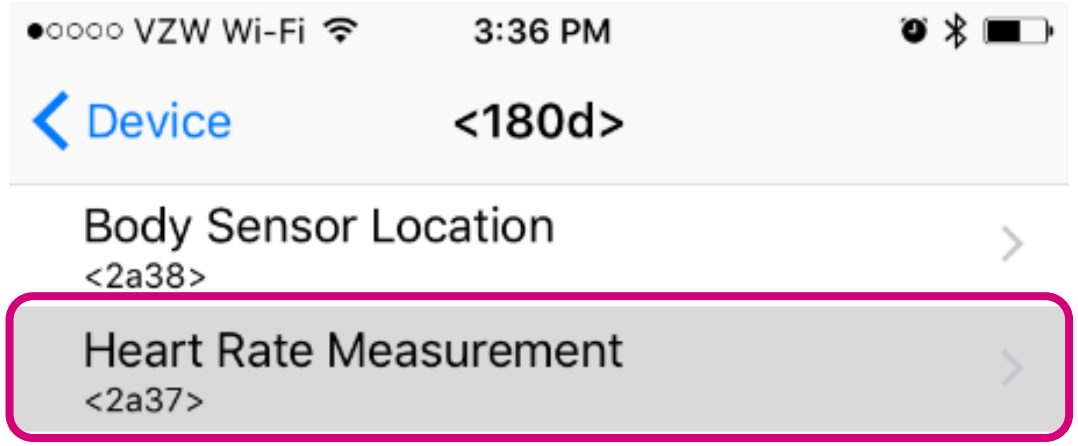


- Click on **Heart Rate** under **Services** (iOS) or **Profiles** (Android)

life.augmented

Select the Heart Rate Profile

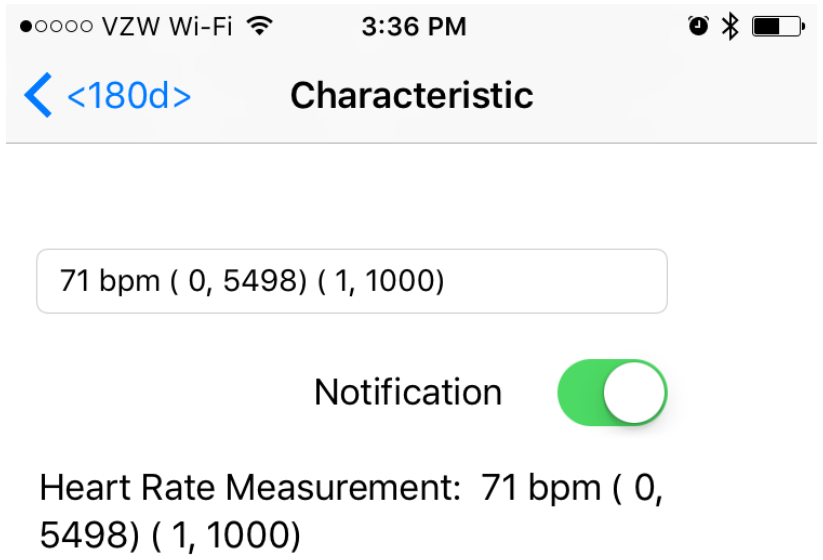
iOS



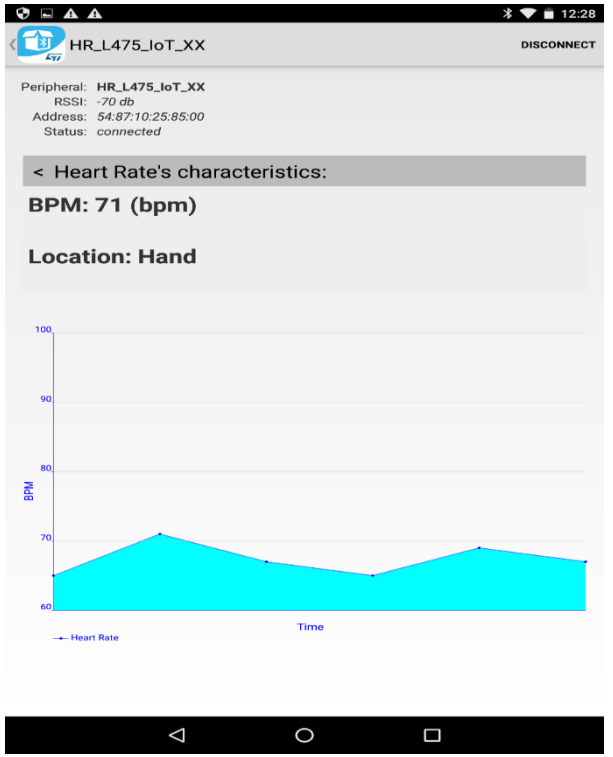
- iOS users, please click on **Heart Rate Measurement**

Display HR Data

iOS



Android



- You should see the simulated heart rate.

STM32L4 Discovery Kit IoT Node Azure IoT Connectivity Summary

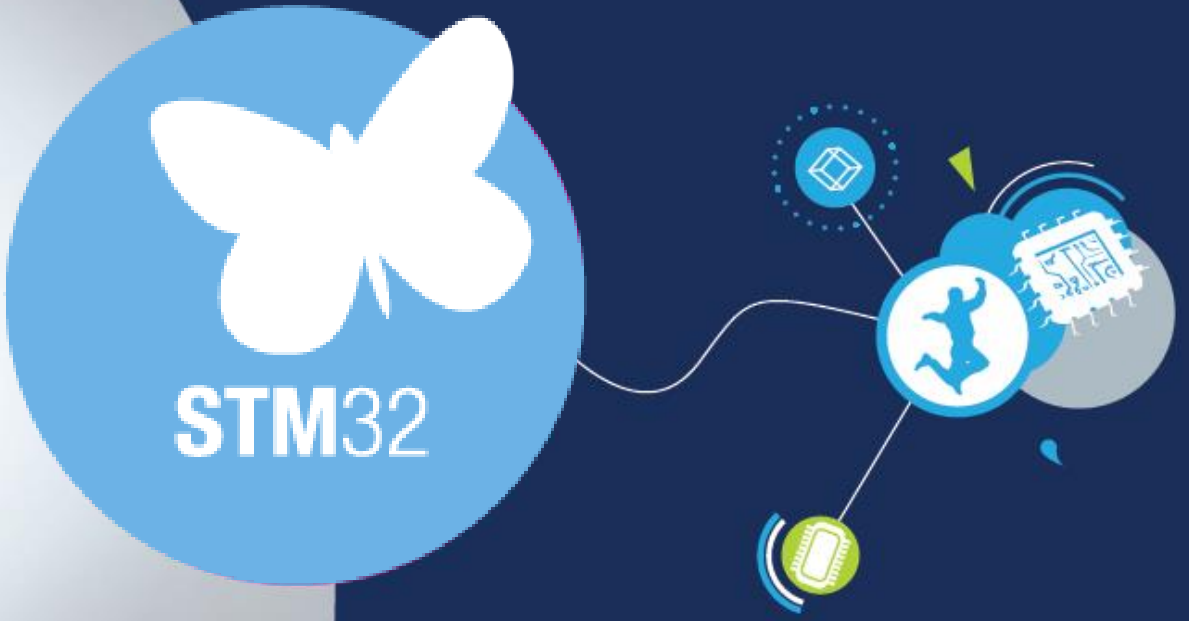
- STM32L4 MCU enables your IoT projects with the combination of ultra-low power and high performance
- STM32L4 Discovery Kit IoT is a compact, yet powerful board to explore various connectivity options including Microsoft Azure IoT
- Next Steps
  - You can add BLE and NFC features to the Azure project to create an application that can connect to the cloud and also has short range connectivity with your smartphone
  - Make your own custom web application by following the Azure guide linked earlier
  - Use the rich set of sensors on the board to get creative!
  - Post your projects or ideas on the ST Community website to gather feedback and get support: https://community.st.com/community/share-your-activities/pages/overview

Releasing Your Creativity

/STM32

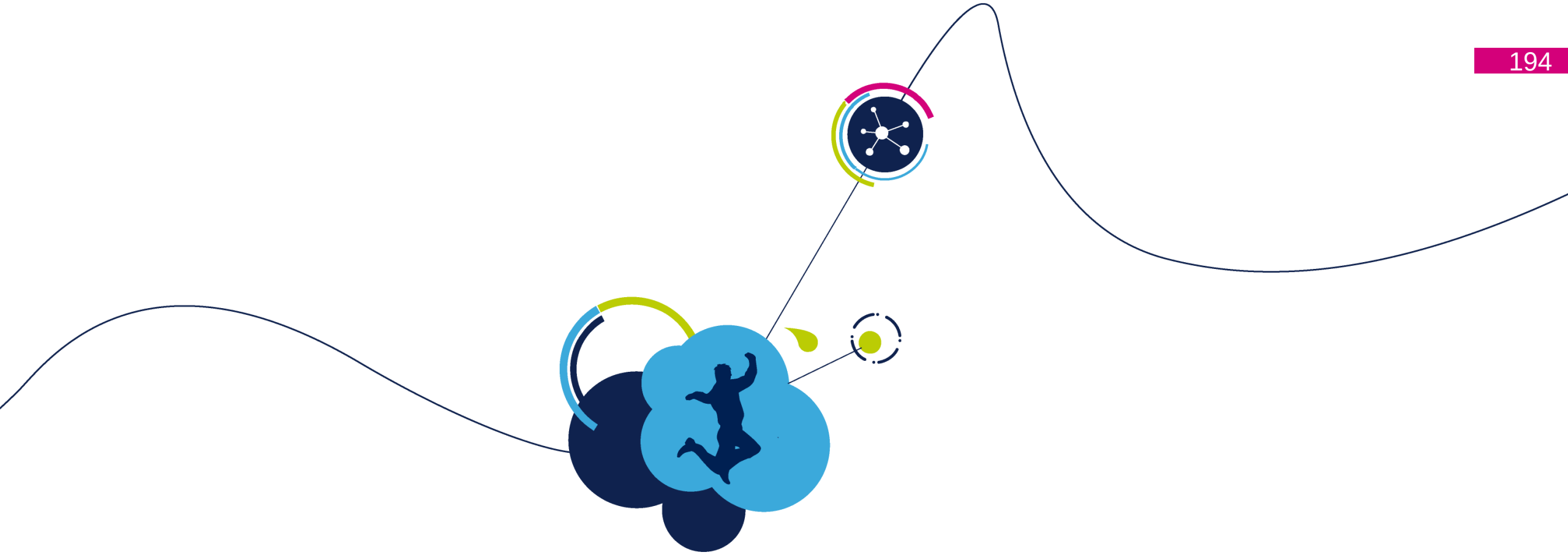
@ST_World

st.com/e2e



www.st.com/stm32

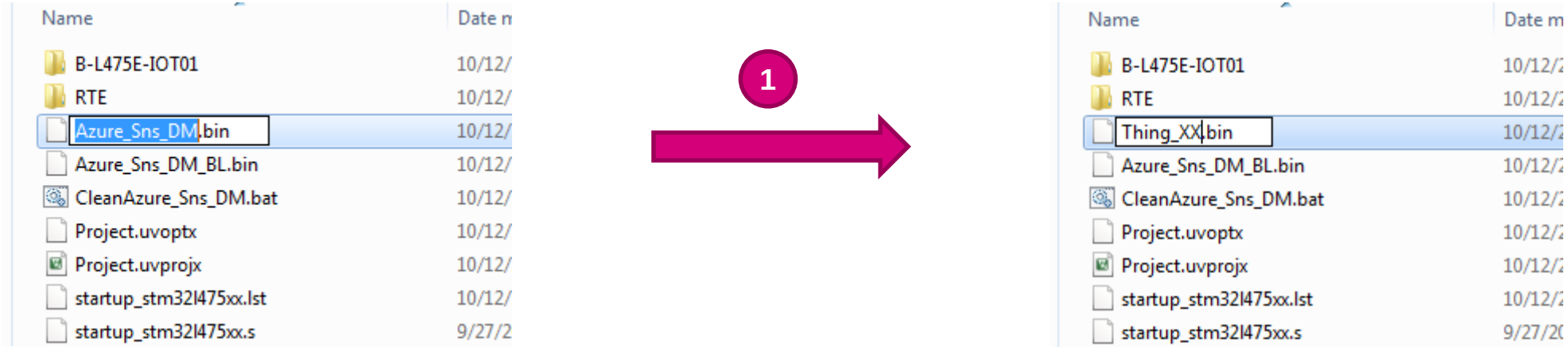
life.augmented



Appendix 1: Upload firmware to Azure storage

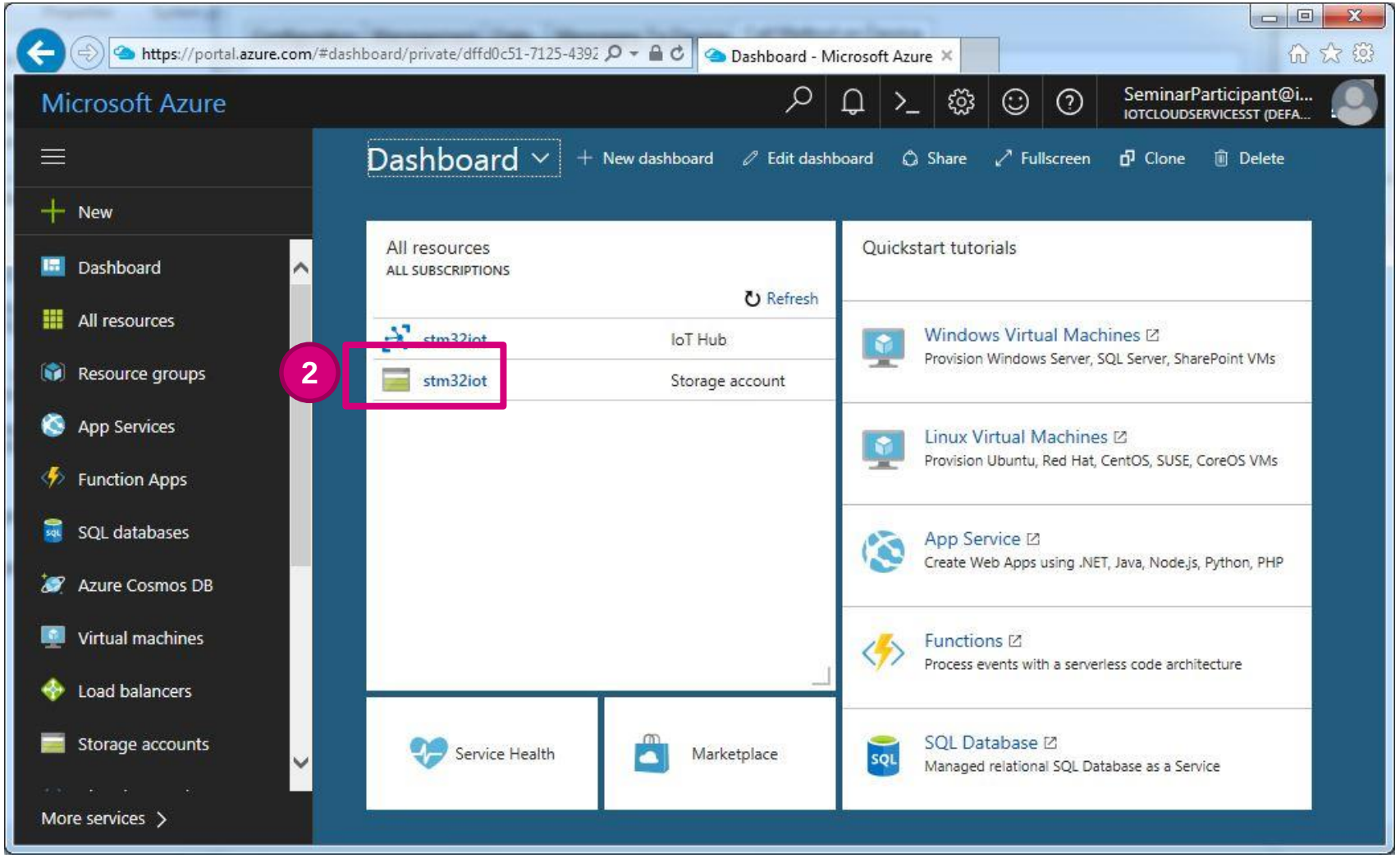
Step 1: Prepare Firmware binary for Upload to Azure Cloud Storage

1. In your project directory, rename your **`Azure_Sns_DM.bin`** to **`Thing_XX.bin`** where **XX** is your participant number



Step 2: Upload firmware to Azure storage

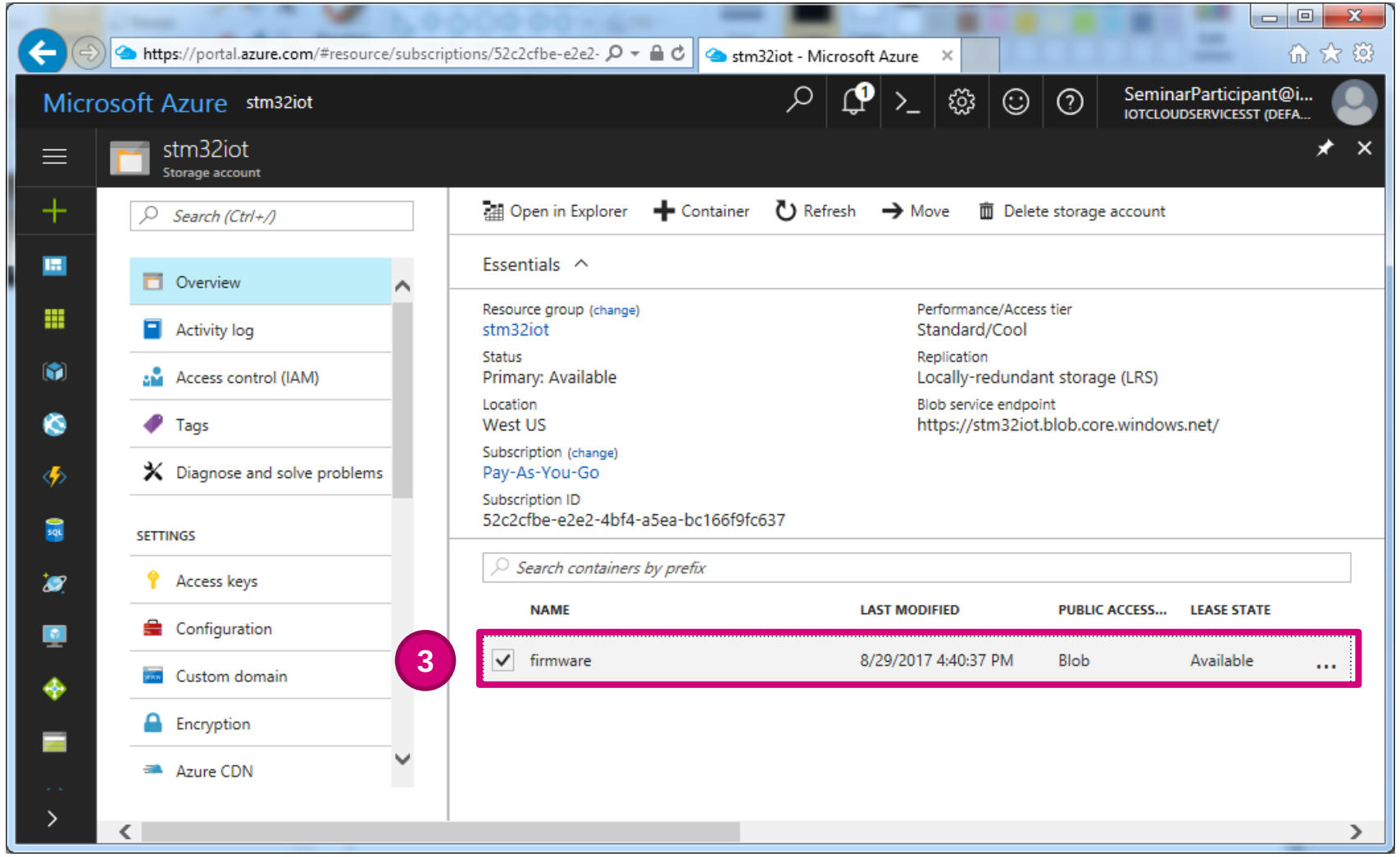
2 Open the stm32iot Storage account from the Azure Portal Dashboard



life.augmented

Step 2: Upload firmware to Azure storage

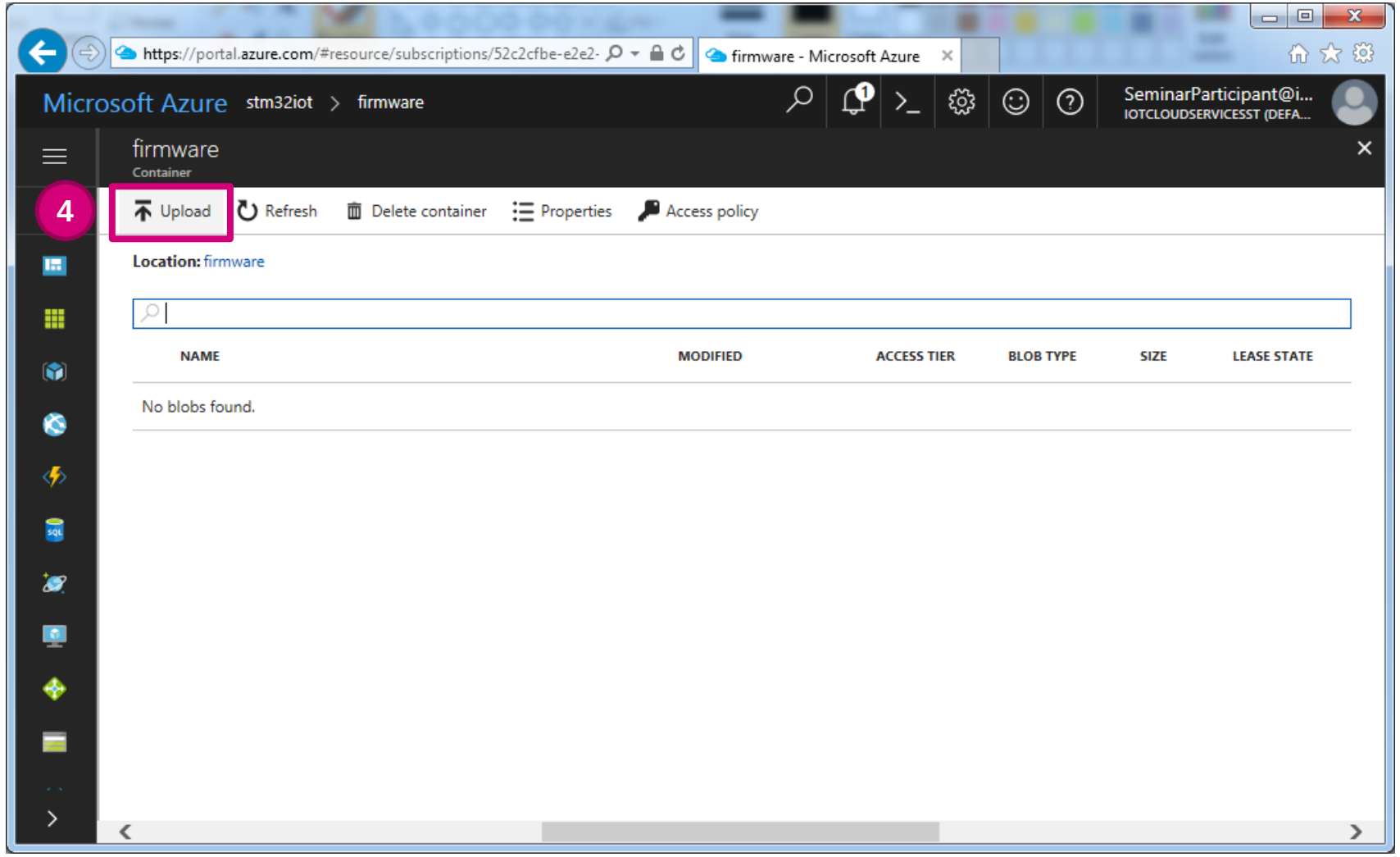
3 Click on the firmware container



life.augmented

Step 2: Upload firmware to Azure storage

4 Click on Upload

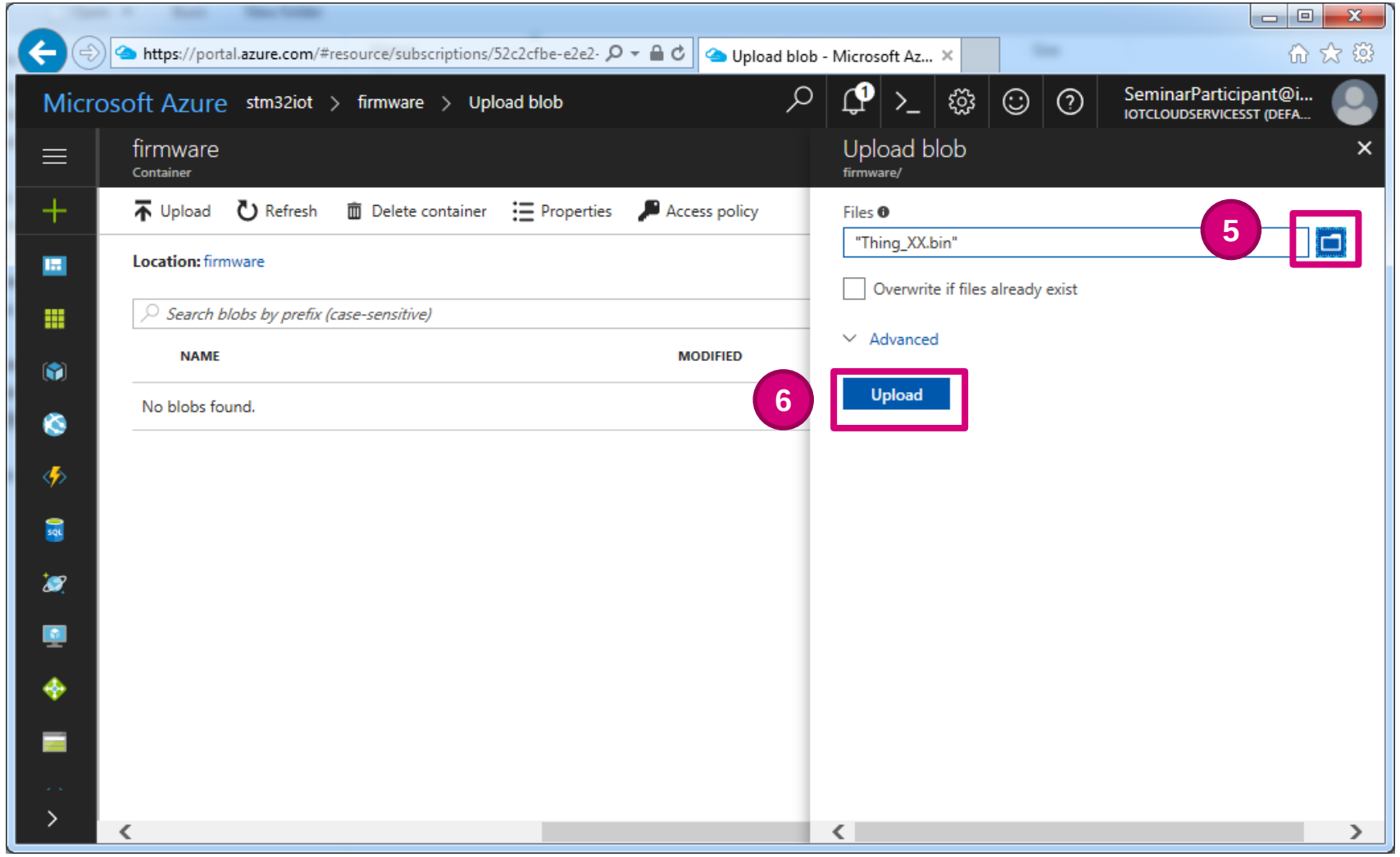


life.augmented

Step 2: Upload firmware to Azure storage

5 Browse to the location of your binary (located in your project folder)

6 Click on Upload



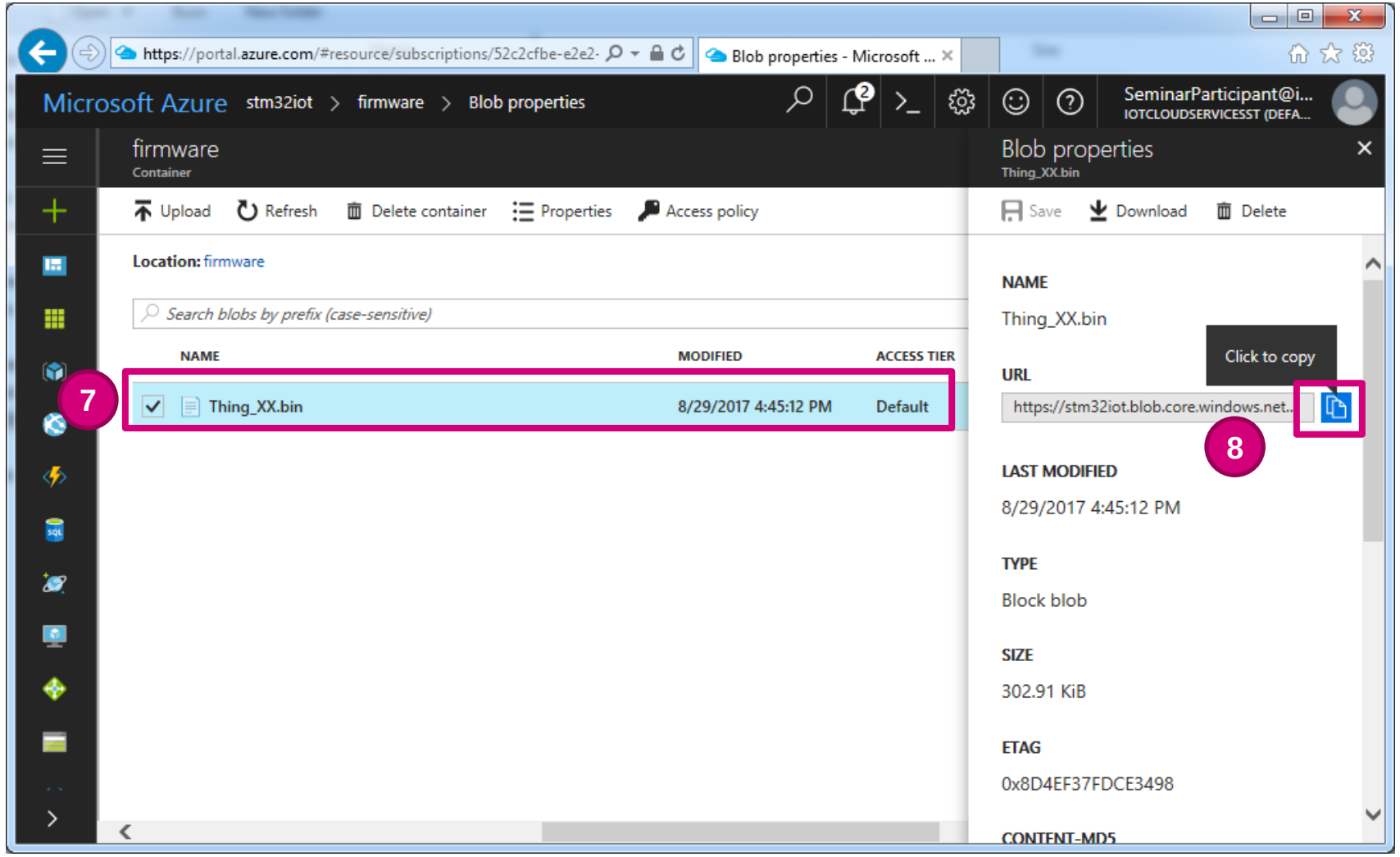
life.augmented

Step 3: Retrieve firmware URL

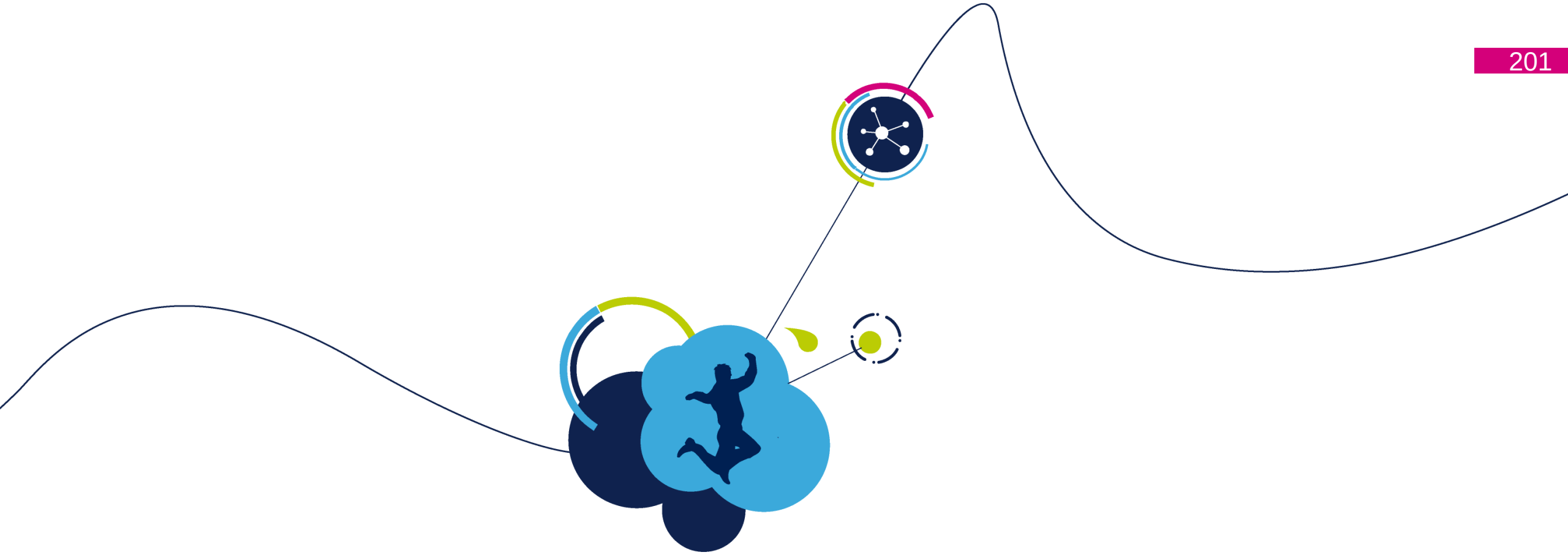
Once the binary has been uploaded:

7. Double-click on your Thing_XX.bin blob

8. Click to copy the blob URL



life.augmented



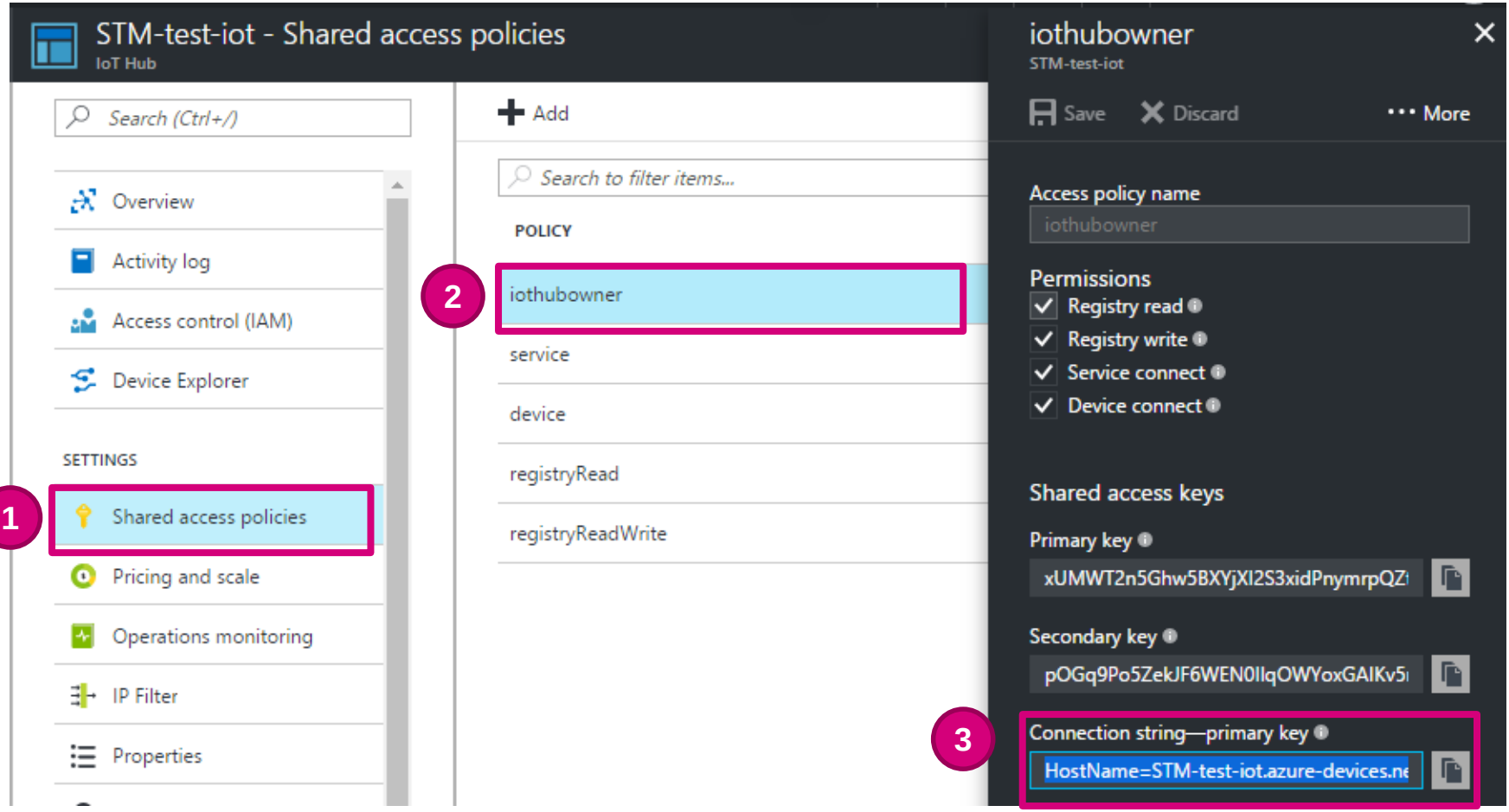
Appendix 2: Using Device Explorer tool to manage devices connecting to your IoT hub

Step 1: Visualize IoT Hub Credentials and Retrieve Connection String

1. Within settings pane for the IoT Hub, click on Shared access policies pane

2. Click on iothubowner in the left side bar

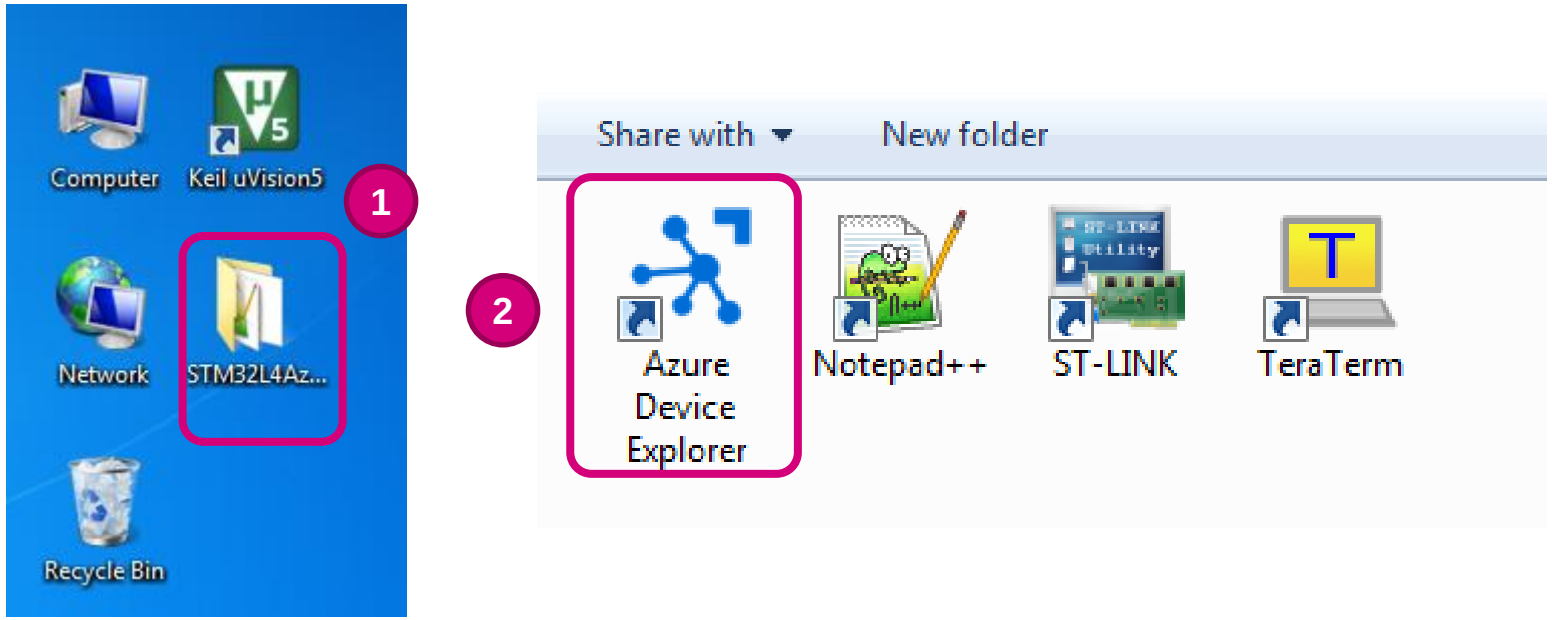
3. Copy the Connection string to Notepad++



life.augmented

Step 2: Launch Device Explorer Desktop

Application

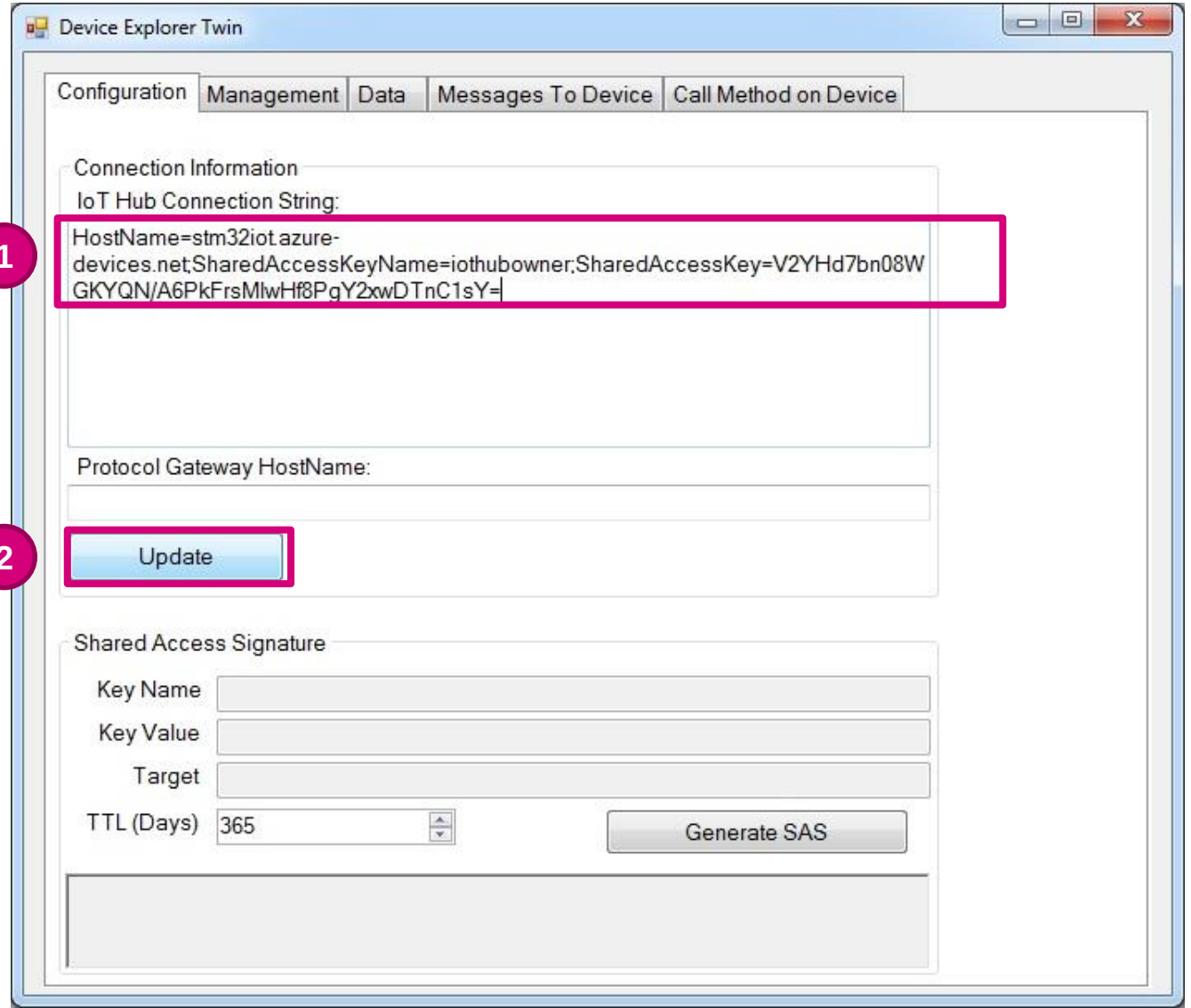


1. Open STM32L4AzureSeminar folder in your Desktop

2. Double-click on the **Device Explorer** shortcut

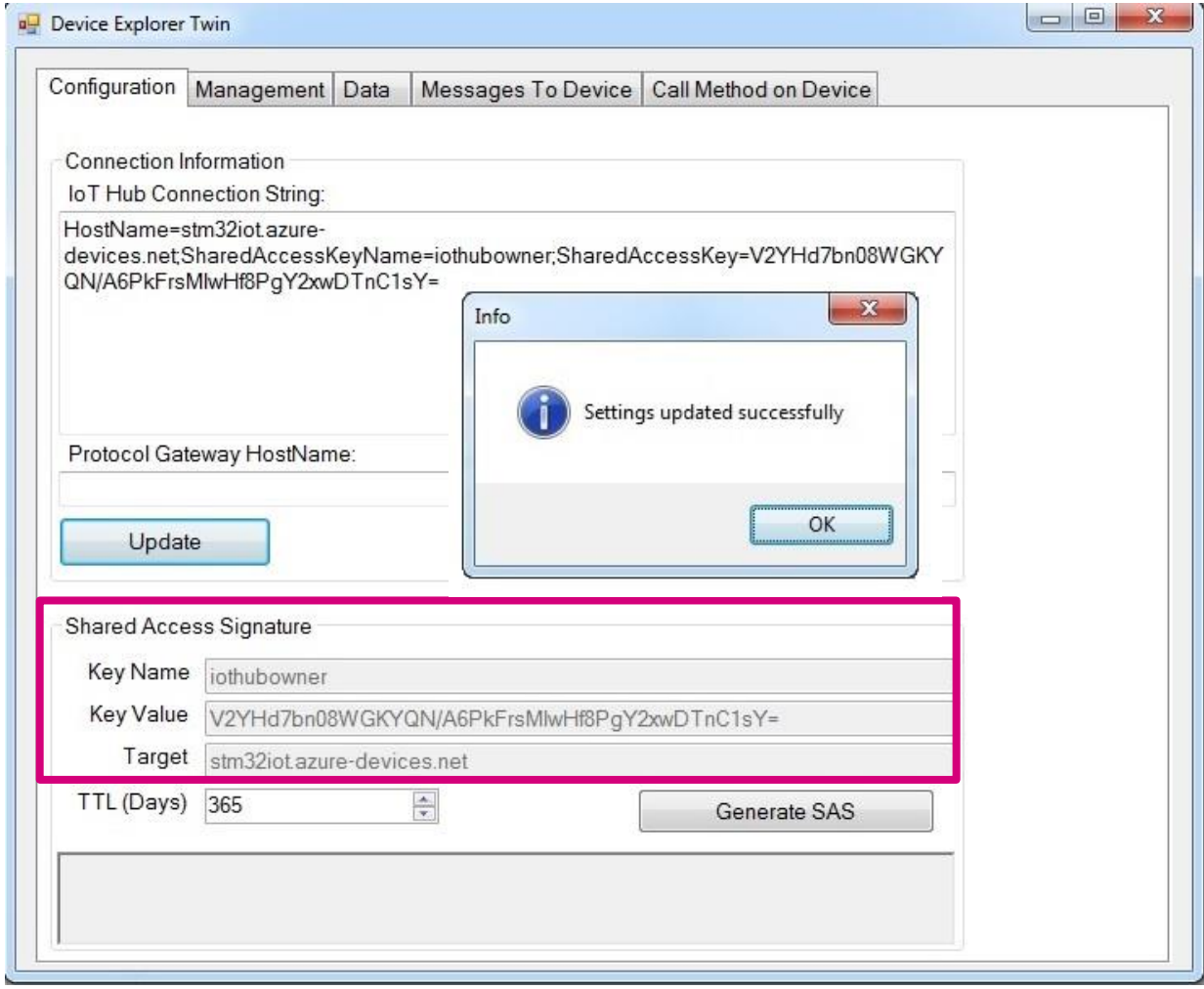
Step 3: Update IoT Hub Connection String in Device Explorer

1. Paste the IoT Hub connection string (from Page 64) in the IoT Hub Connection String
2. Click on Update



Step 3: Update IoT Hub Connection String in Device Explorer (Cont'd)

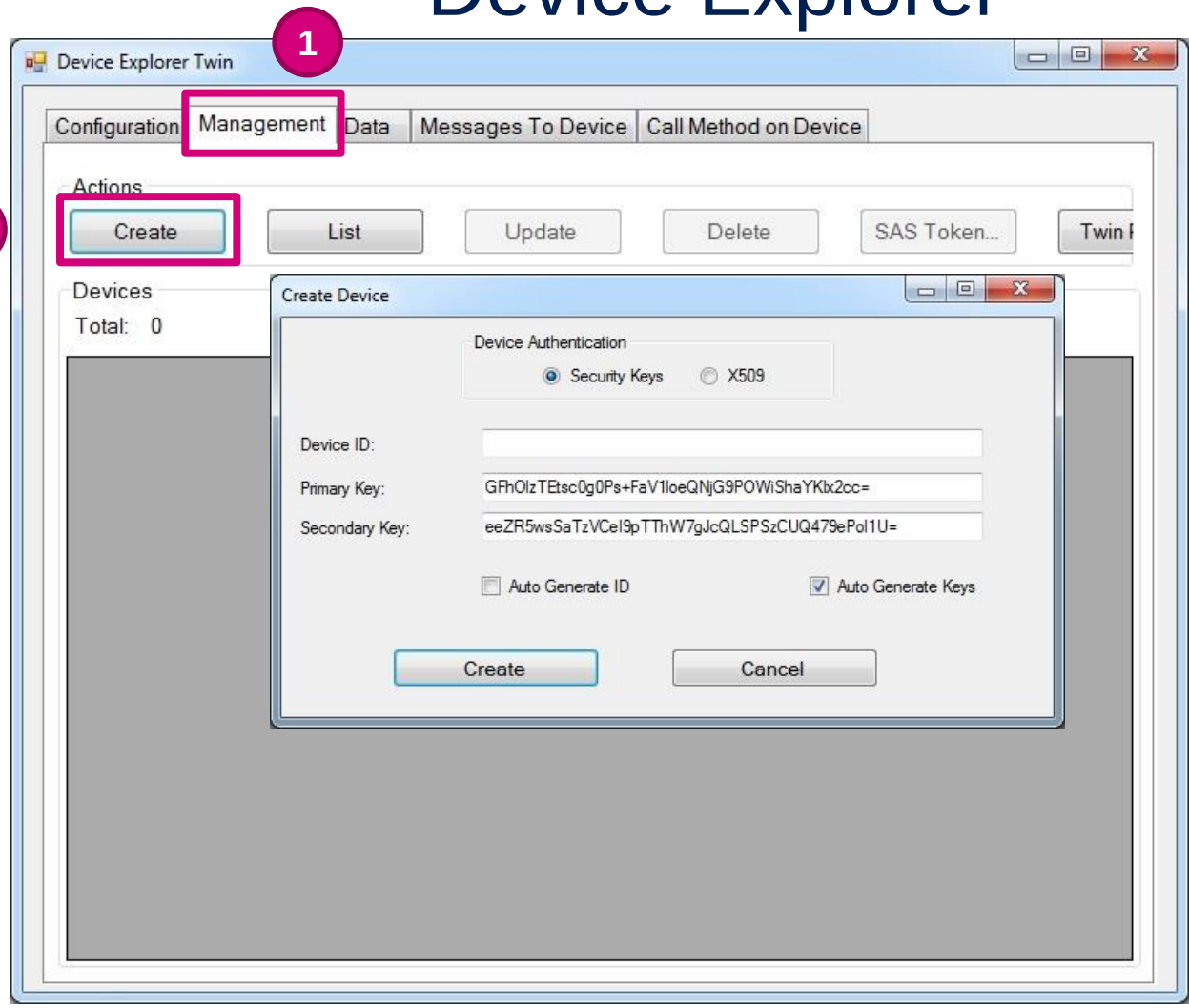
- On successful update the section Shared Access Signature is populated with the three components of the IoT Hub Connection String:
  - Key Name
  - Key Value
  - Target



life.augmented

Step 4: Create a New Device in Device Explorer

- You are now ready to create your IoT device in the IoT Hub:
  1. Go to tab Management
  2. Click on Create
- A Create Device window will appear showing Primary and Secondary keys which are automatically generated



life.augmented

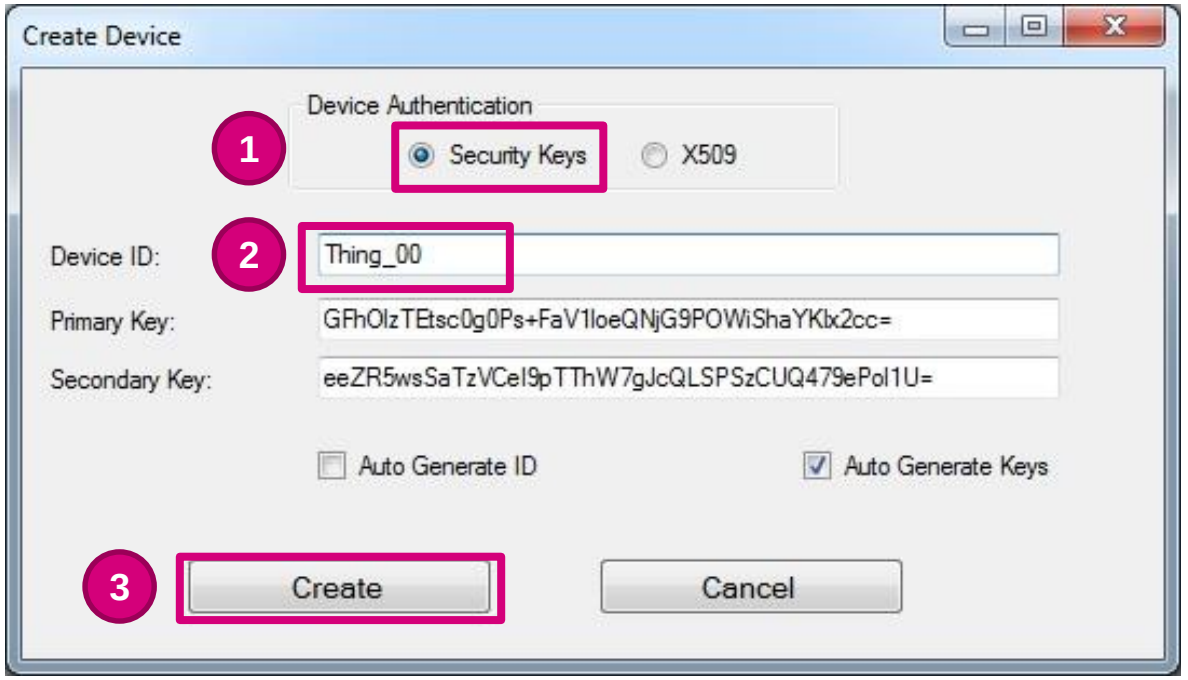
Step 4: Create a New Device in Device Explorer (Cont'd)

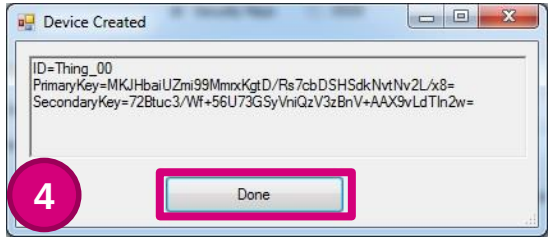
1. For this tutorial we rely on device authentication with keys. However, also device identification with X509 certificated is supported by NucleoL4

2. Insert your device ID "Thing_**XX**": "**XX**" is your participant number (2 characters) printed on the box

3. Click on Create to create the device in your IoT Hub

4. A Device Created window appears, indicating that your device has been successfully registered with this IoT Hub. Click "Done"

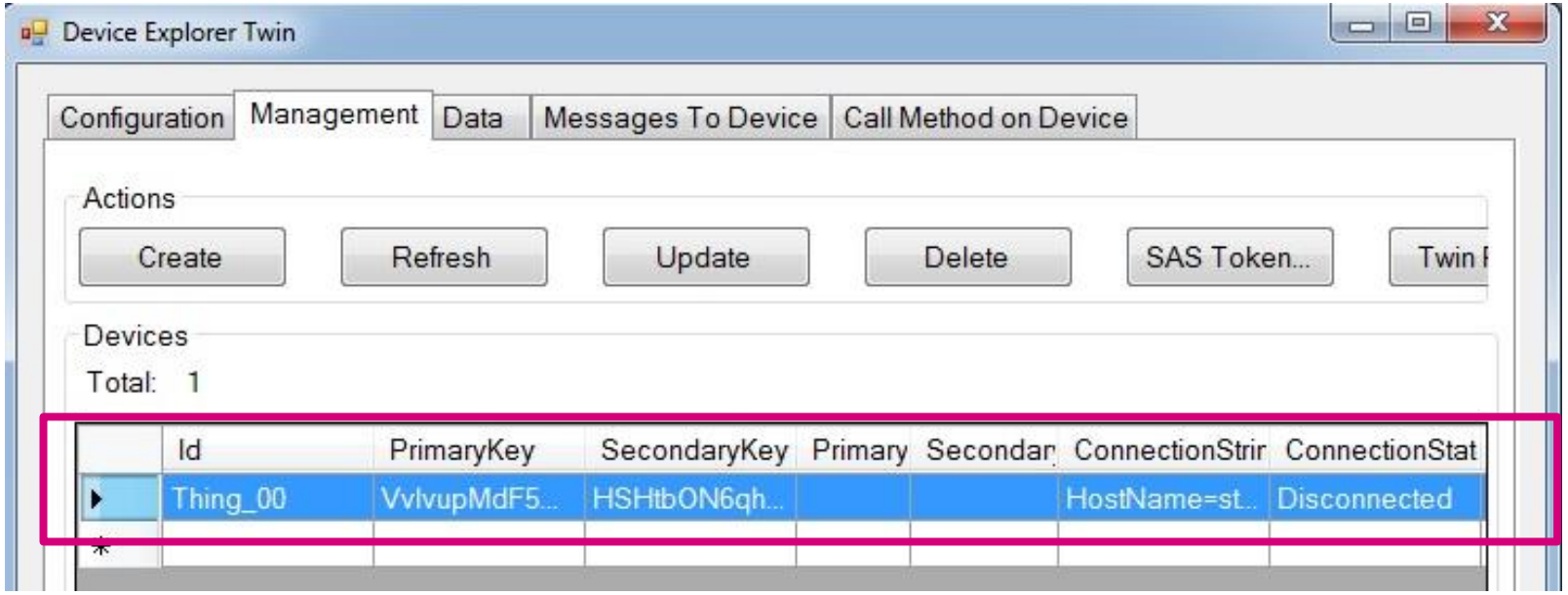




life.augmented

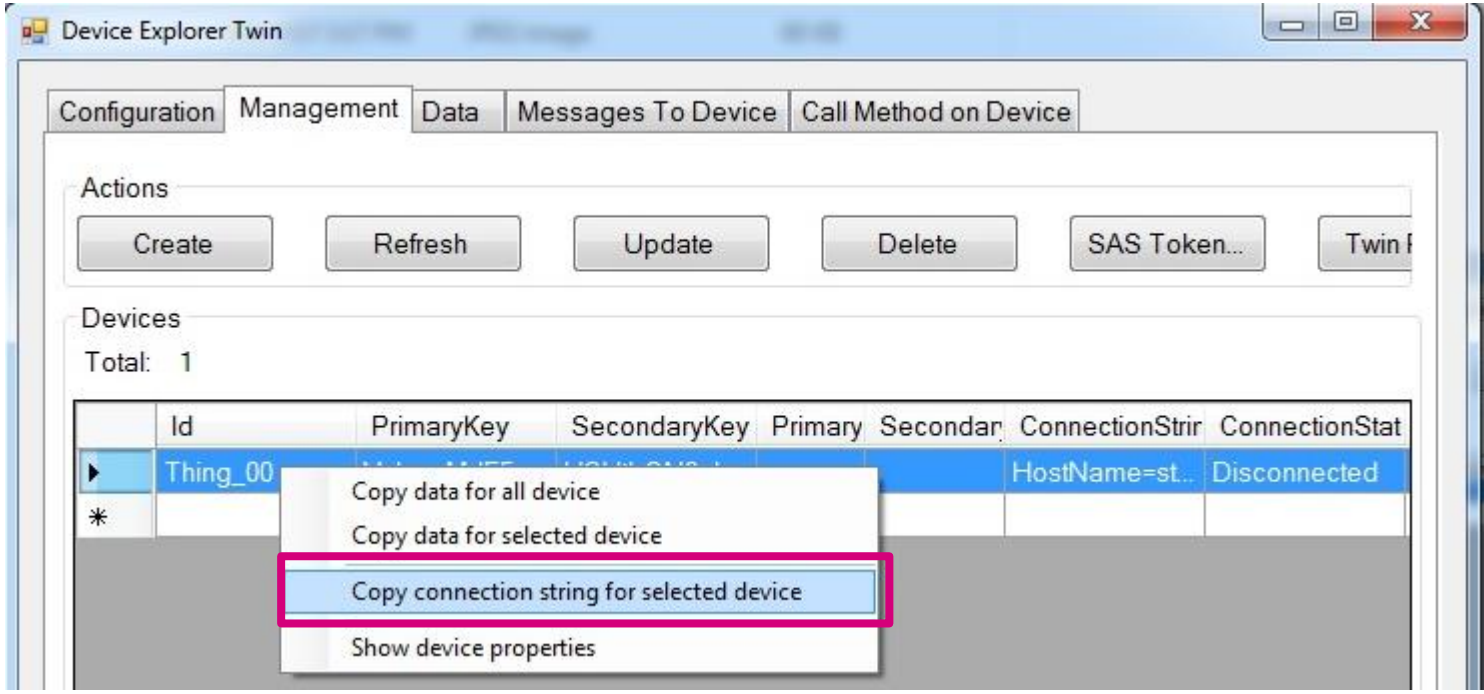
Step 4: Create a New Device in Device Explorer (Cont'd)

- An entry for your device has been created and listed in Management tab of your device explorer. This entry contains the connection string to be stored in the device to authorize access to your IoT Hub



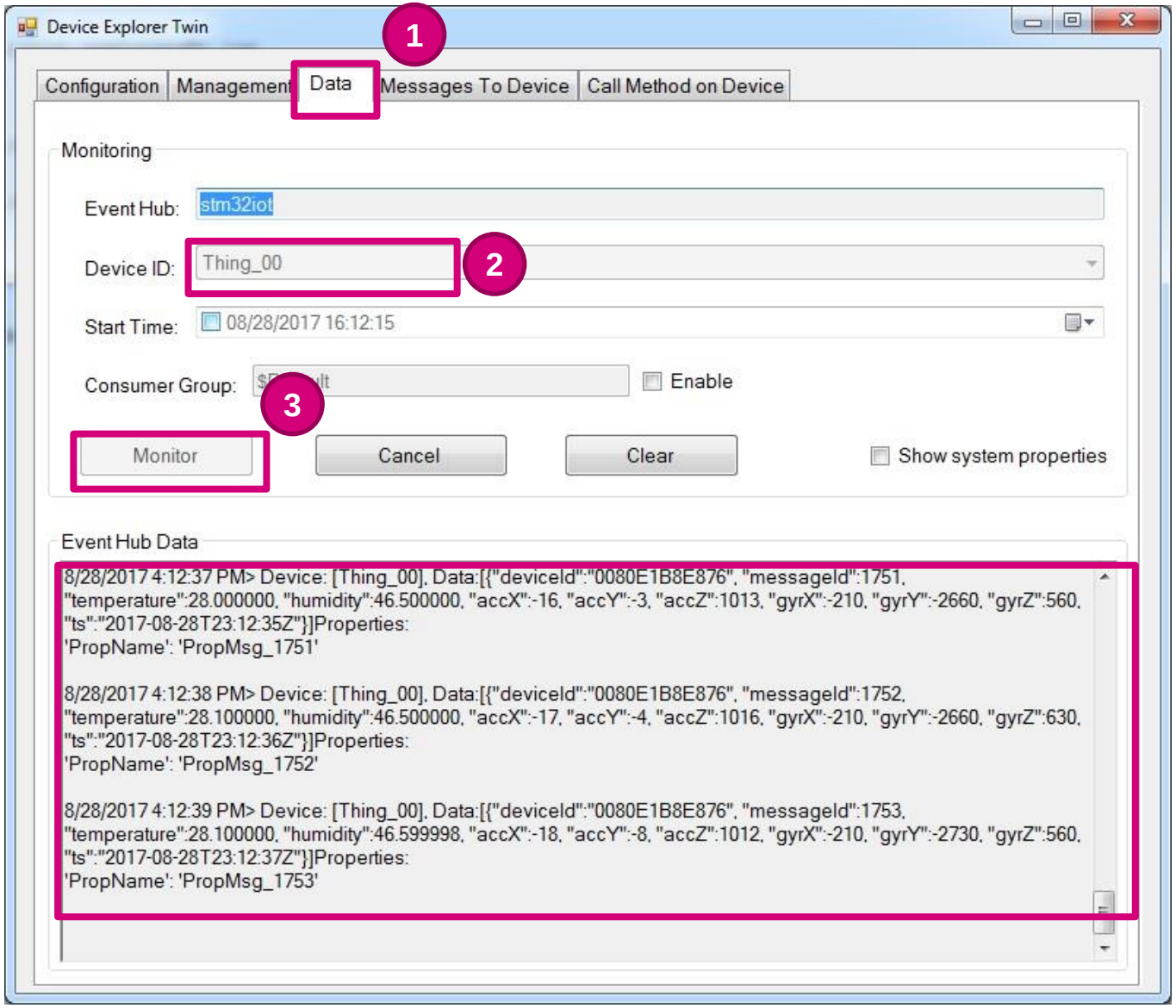
Step 5: Retrieve Connection string for added device

- Select your device ID “Thing_**XX**”: “**XX**” is your participant number (2 characters) printed on the box
- Use the mouse right click for context menu then select Copy connection string for selected device
- Paste the connection string in Notepad++. It will be used later



Step 6: Visualize Messages in Device Explorer

1. Select Data tab
2. In Device ID, Select your device “Thing_**XX**”: “**XX**” is your participant number (2 characters)
3. Click on Monitor to start visualizing the log of messages received by IoTHub

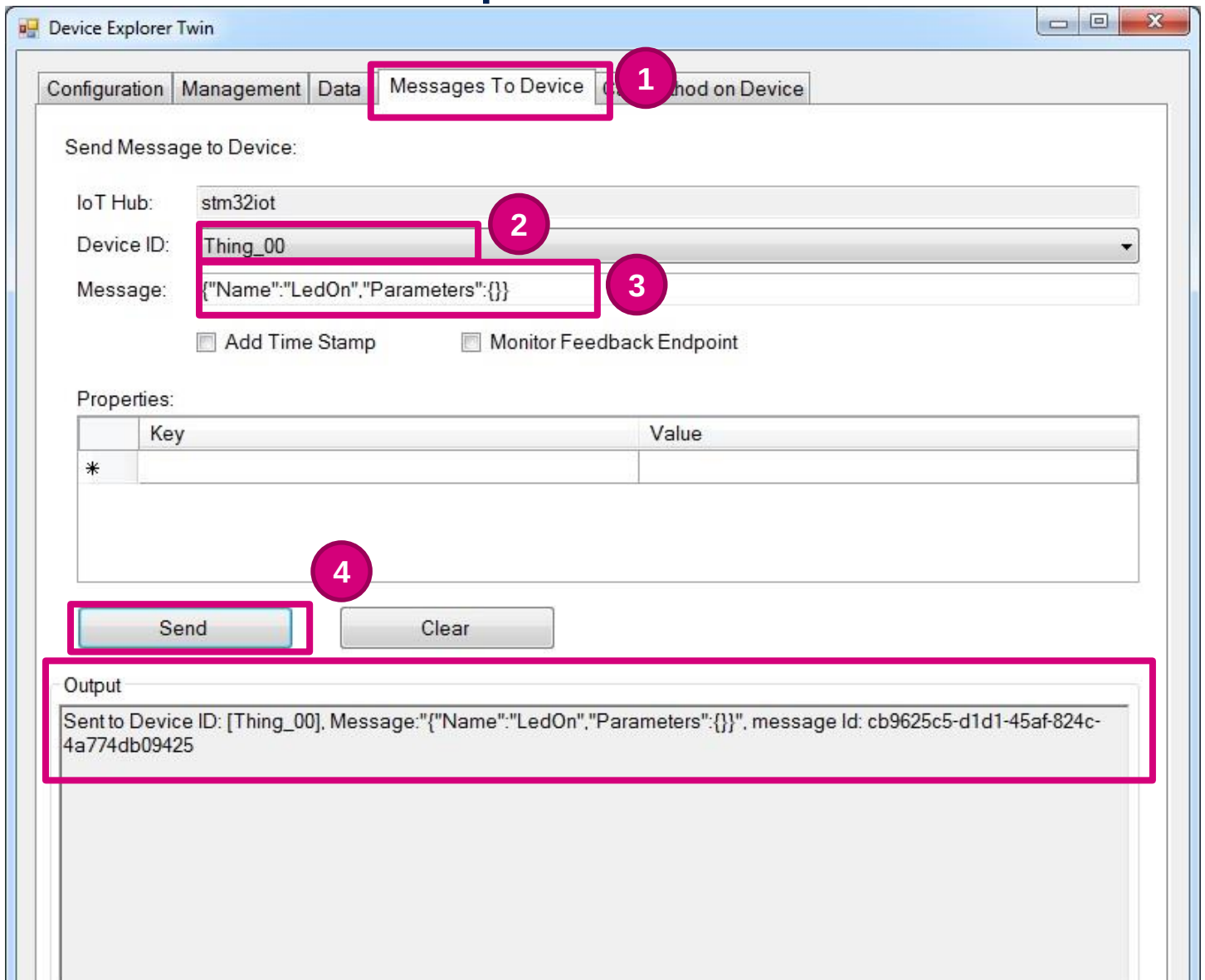


life.augmented

Step 7: Send Message to Device

with Device Explorer: Turn LED On

1. Select Message To Device tab
2. In Device ID, Select your device “Thing_**XX**”: “**XX**” is your participant number (2 characters)
3. Copy-paste the message below to the Message field to turn the green LED on:
   - {"Name":"LedOn", "Parameters":{}}
4. Click on **Send**



life.augmented

Step 8: Call FW-update Method on Device with Device Explorer

1. Select Call Method on Device tab
2. Select your device “Thing_**XX**”: “**XX**” is your participant number (2 characters)
3. Copy-paste the message below to Method name

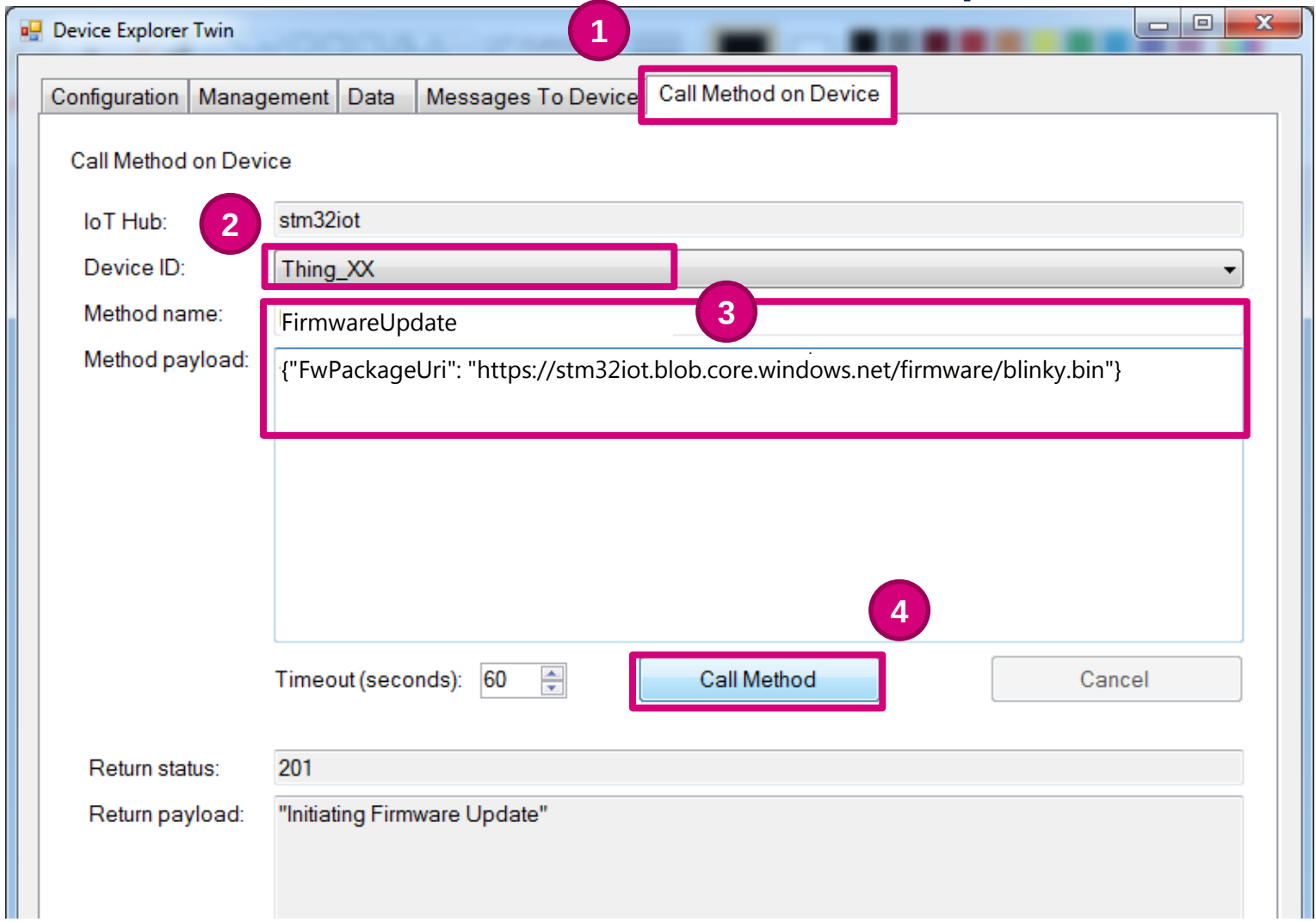
```
FirmwareUpdate
```

   Copy-paste the message below to Method payload

```
{"FwPackageUri":
"https://stm32iot.blob.core.windows.net/
firmware/blinky.bin"}
```

- Click on Call Method

4.



life.augmented

Releasing Your Creativity

www.st.com/stm32

life.augmented