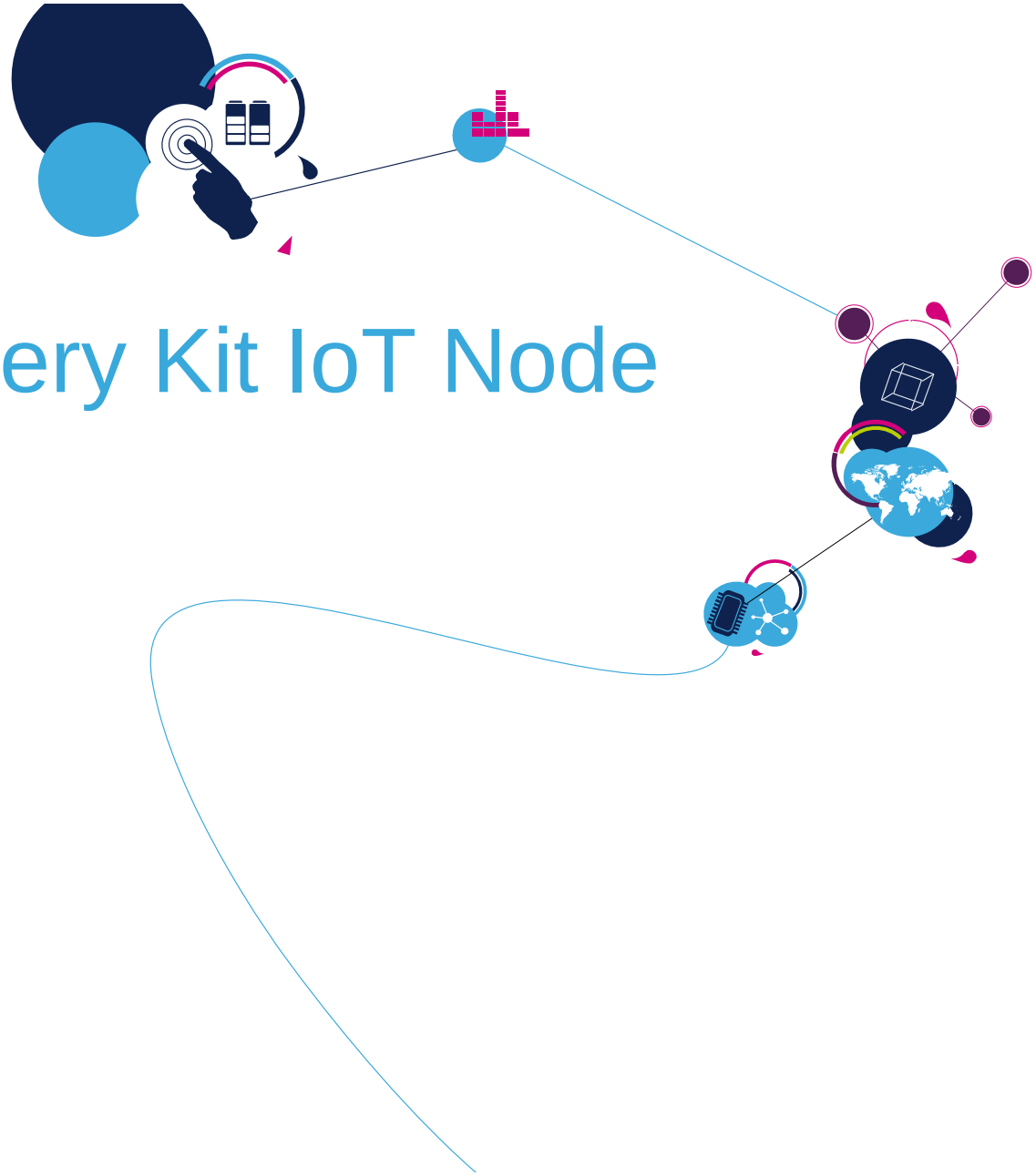


STM32L4 Discovery Kit IoT Node Seminar

Bruno Montanari
Cicero Pompeu

v1.6

life.augmented

Agenda

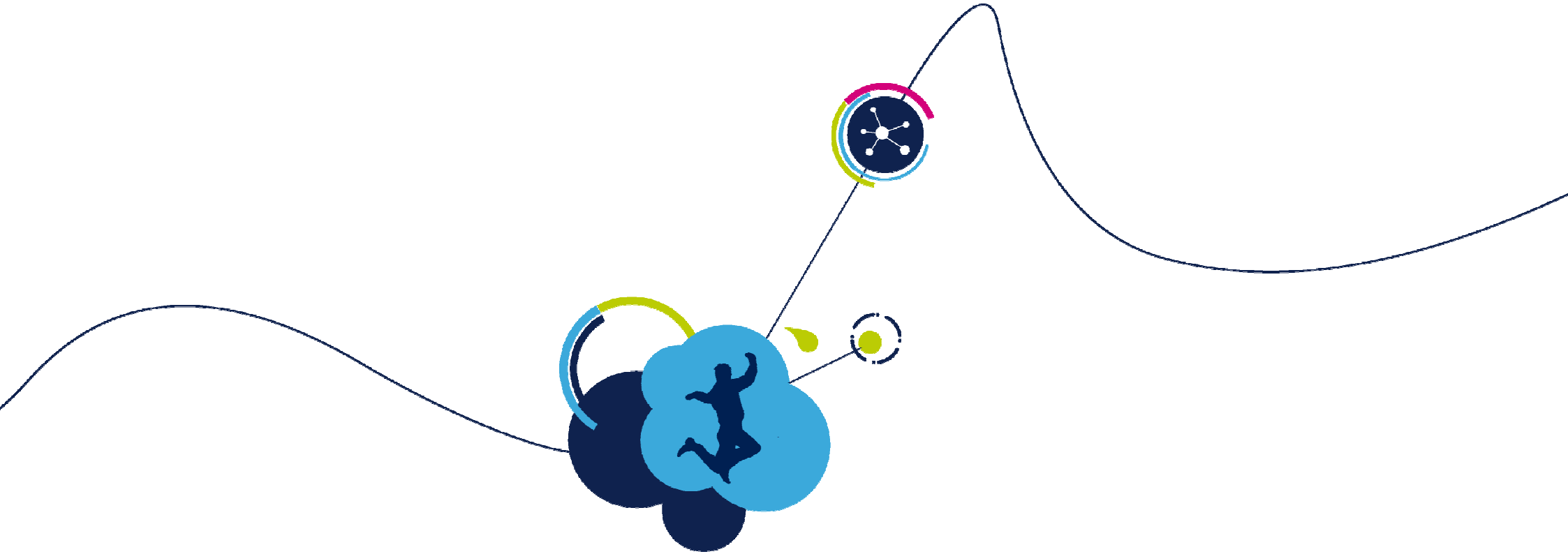
Presentation

- Training Material Check/Installation Help
- Overview of the STM32 Portfolio
- IAR Installation
- Overview of the STM32L475
- Overview of the STM32L475 Discovery Kit IoT Node
- STM32Cube™ Introduction
- IAR License Installation
- ST-Link Utility Installation
- Lab 1 : Getting Started with STM32CubeMX - Blinky LED
- USB Overview
- NFC Overview
- Sensors Overview
- Lab 2 : Real Time Sensor Logging

Agenda

Presentation

- Bluetooth® Low Energy Overview
- Lab 3 : Bluetooth® Low Energy Pairing
- Lab 4 : Bluetooth® Low Energy P2P Communication
- SPIRIT Sub-GHz Overview
- Wi-Fi Module Overview
- Lab 5 : Wi-Fi Client Server
- Lora Overview
- Amazon AWS IoT Overview
- Lab 6 : Creating your Device (“Thing”) on AWS
- Lab 7 : Connect to AWS IoT & Send Sensor Data
- Alexa Voice Demo
- Optional Lab : USB Device Using a Virtual COM Port



Training Material Check / Installation Help

life.augmented

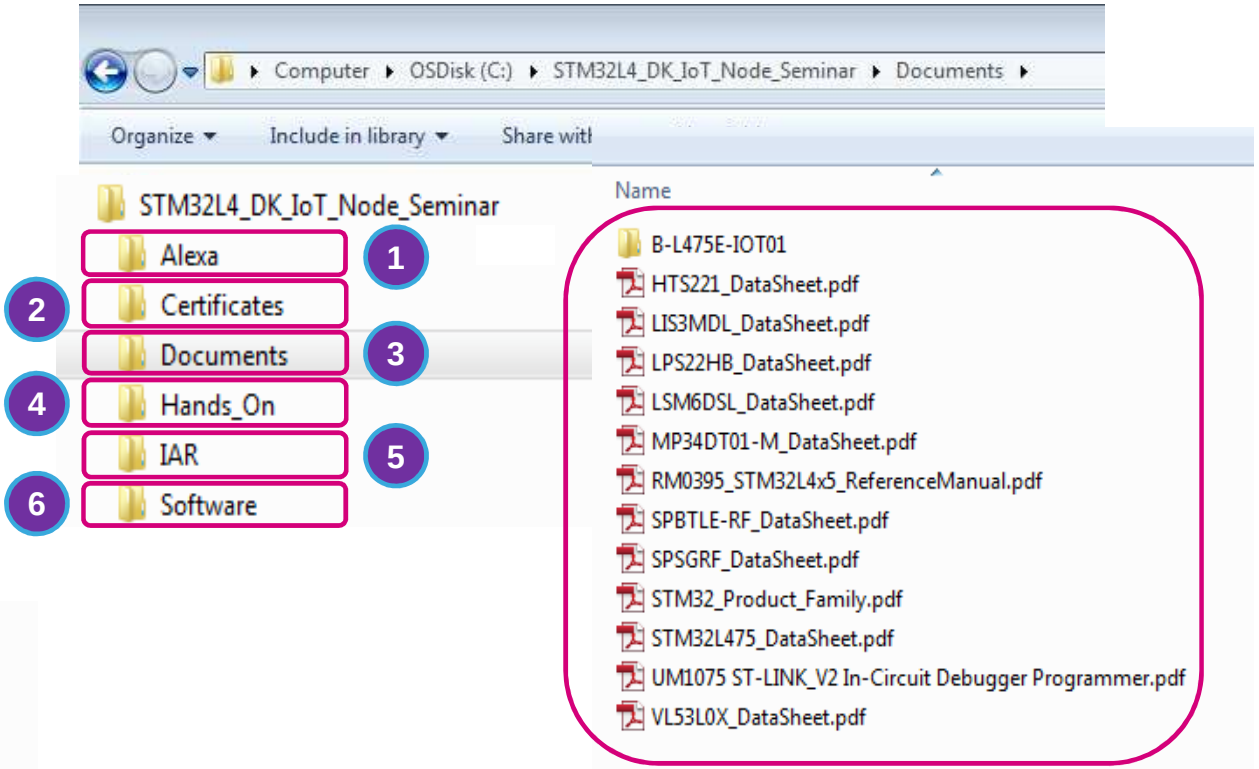
Training Material Installation

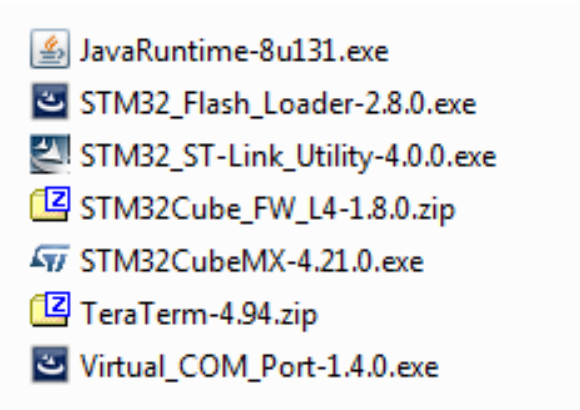
- Each participant should have install STM32CubeMX, the STM32CubeL4 HAL, JAVA, Tera Term, Notepad++. All the material and installers needed and associated lab collateral are available in:

LINK here

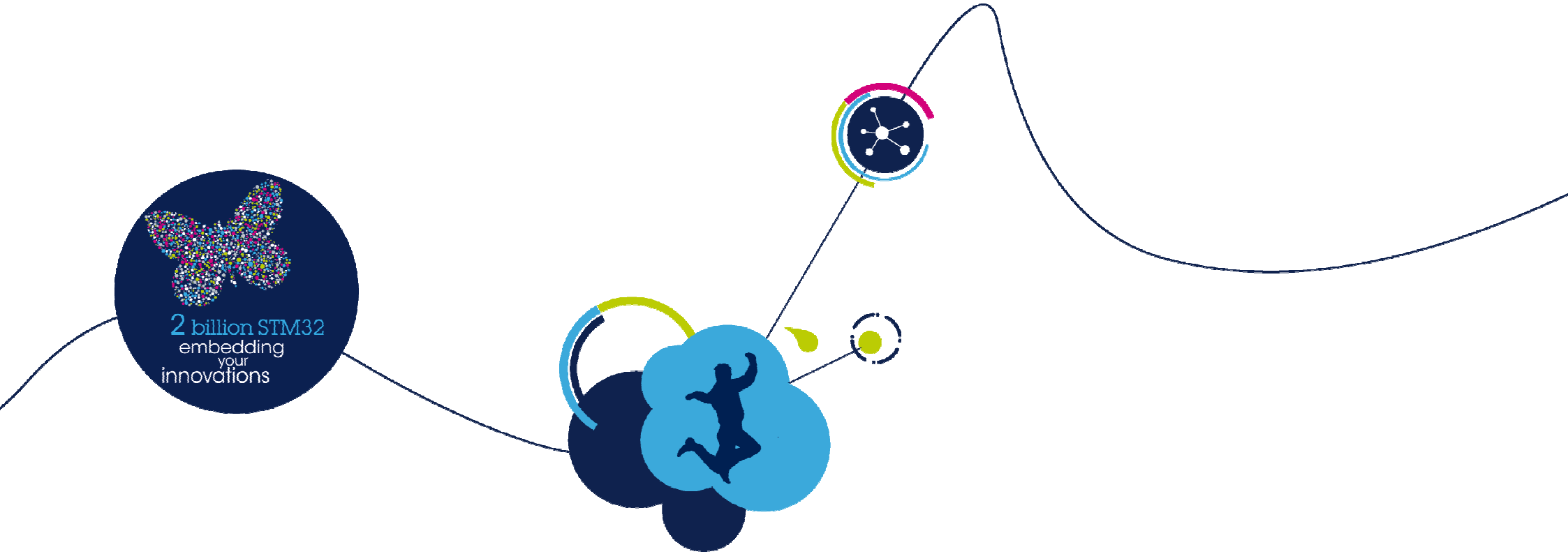
Seminar Directory Content

1. Alexa Skill Code
2. Thing Certificates
3. Documents
4. Hands on Labs
5. IAR
6. Software





life.augmented

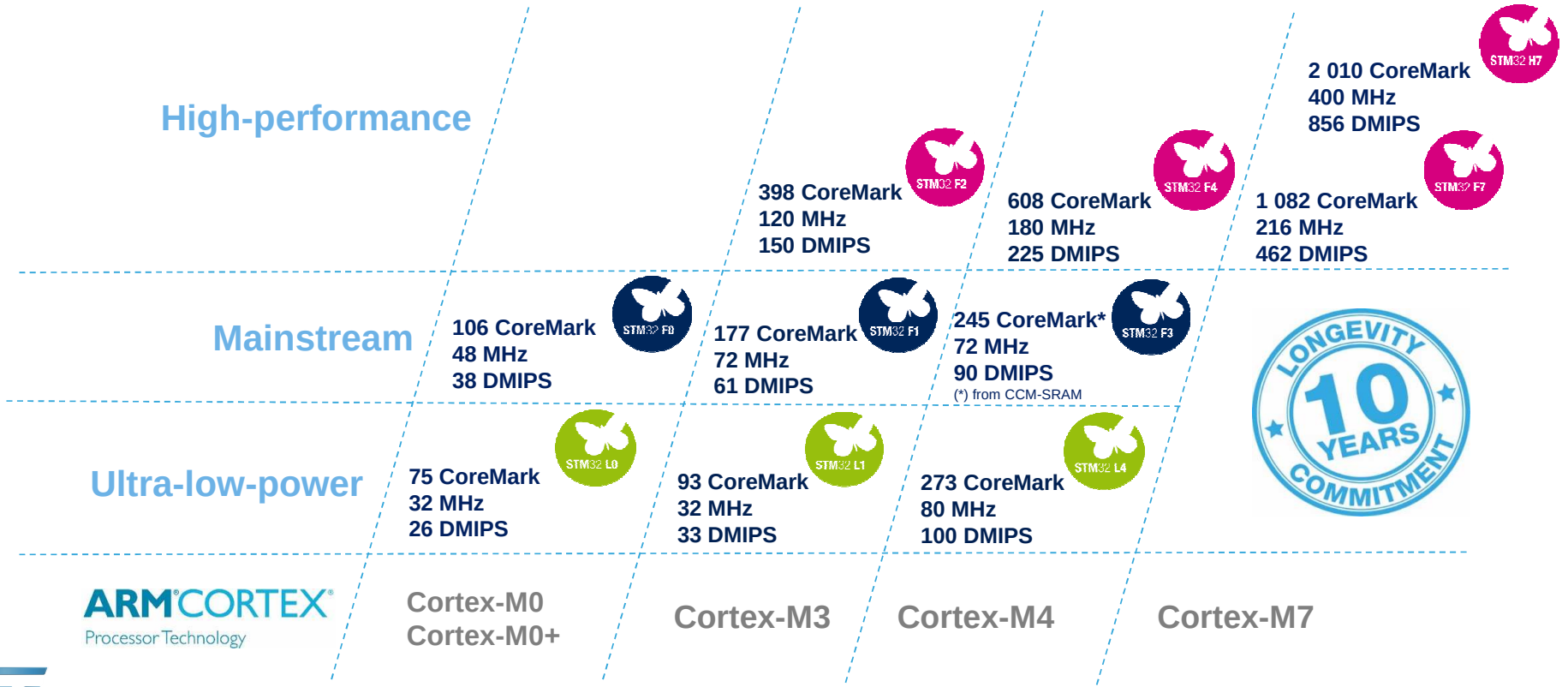


Overview of the STM32 Portfolio

life.augmented

STM32 Portfolio

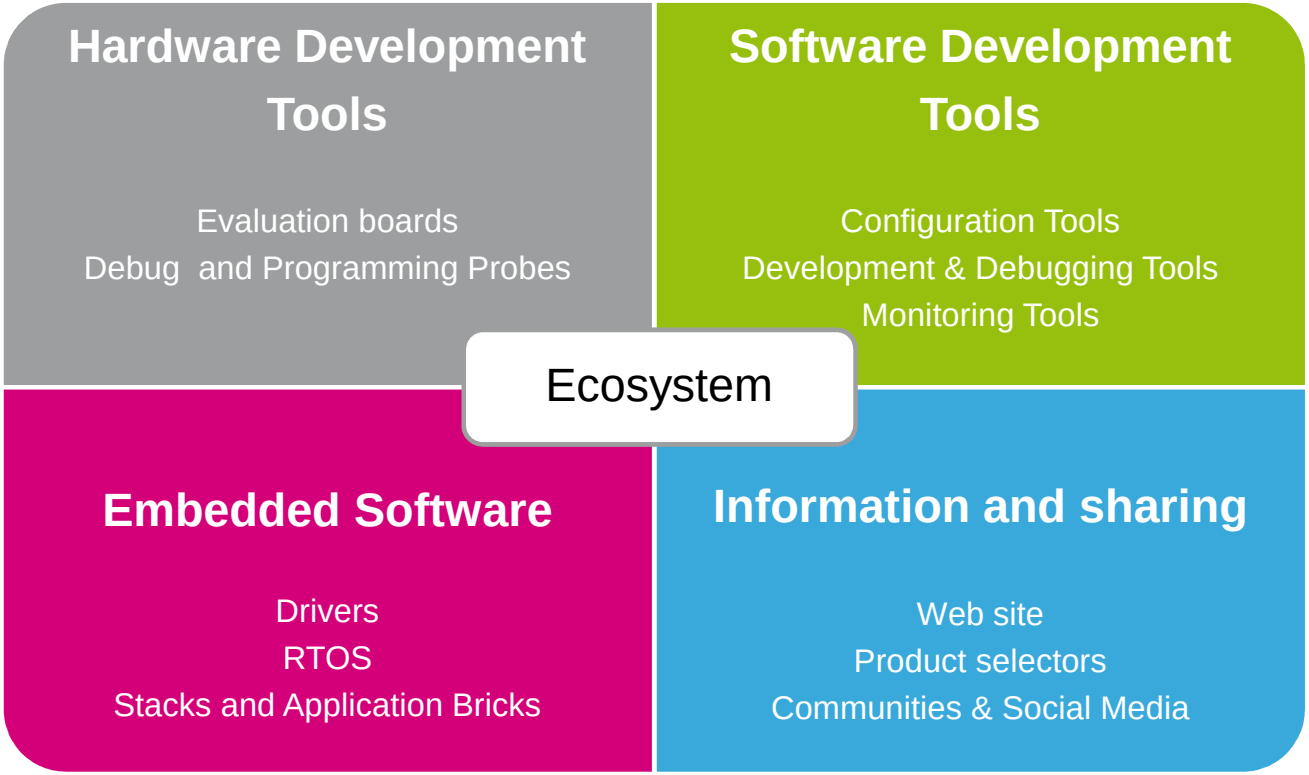
10 product series / More than 40 product lines



life.augmented

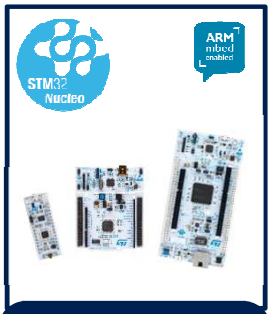
What is the STM32 MCU Ecosystem?

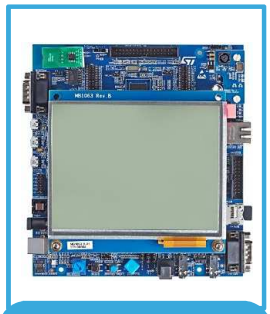
All collaterals required to develop with an MCU



life.augmented

Hardware Development Tools



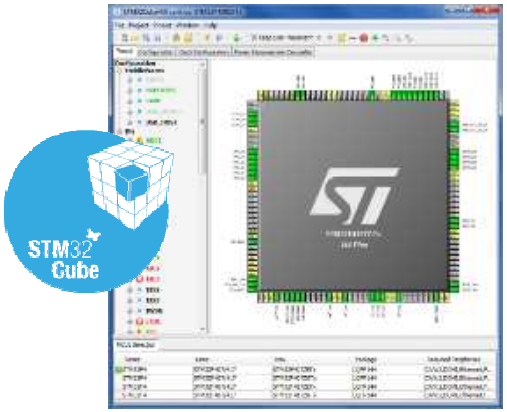


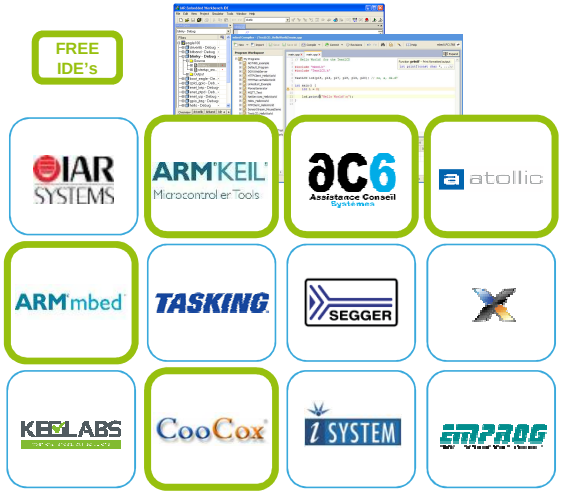
	STM32 Nucleo	Discovery kits	Evaluation boards	3rd parties
Typical use case	Flexible prototyping, Community	Prototyping, Creative demos	Full feature evaluation	From full evaluation to open hardware
Extension possibilities	+ + +	+ +	+	
Connectivity	Arduino™ ST Morpho	ST	ST	

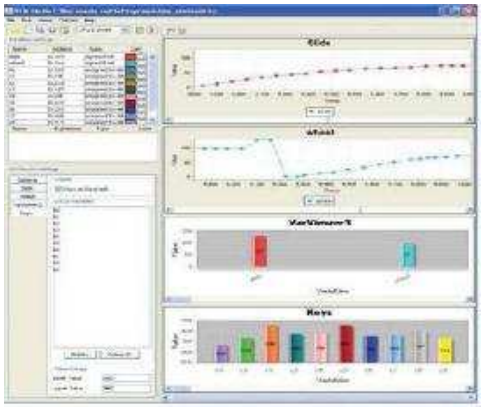
life.augmented

Software Development Tools

A complete flow, from configuration up to monitoring







STM32CubeMX
Configure & Generate Code

Partners IDEs
Compile and Debug

STMStudio
Monitor

life.augmented

STM32 ODE Platform

Hardware

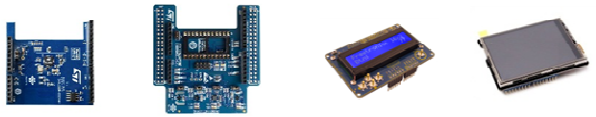
STM32 Nucleo development boards

Software

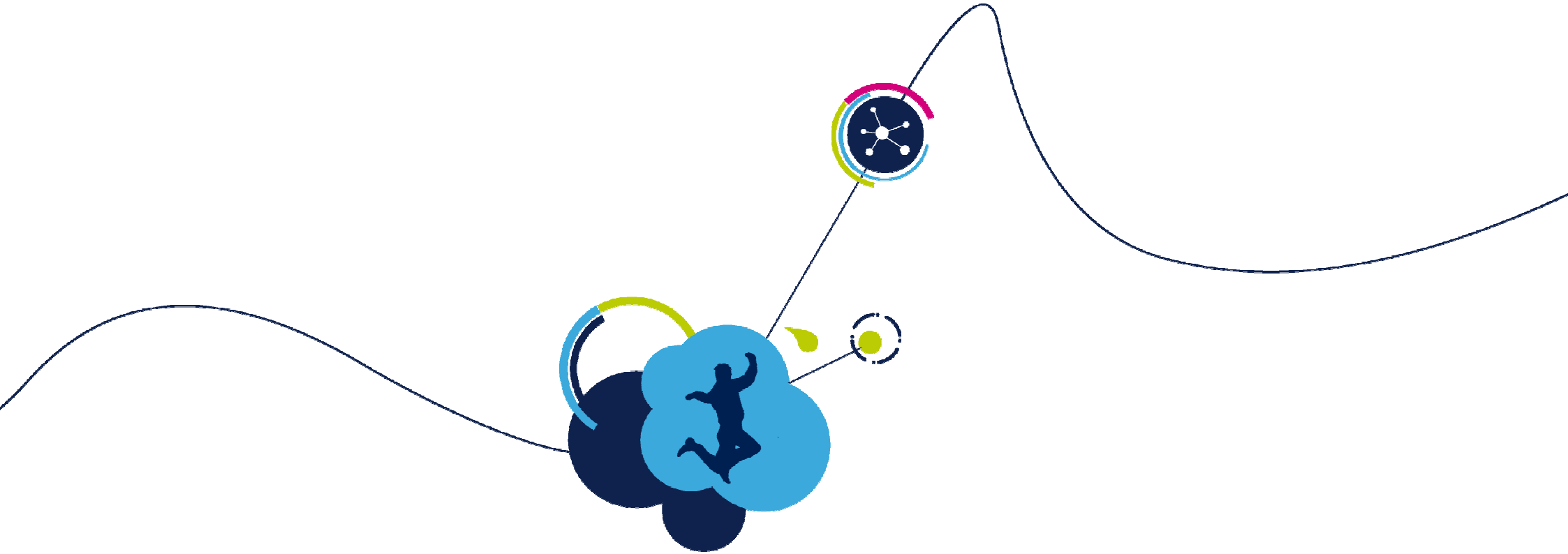
STM32Cube software library

Expansion

STM32 Nucleo expansion boards from ST and third parties



STM32Cube expansion SW



IAR Installation

life.augmented

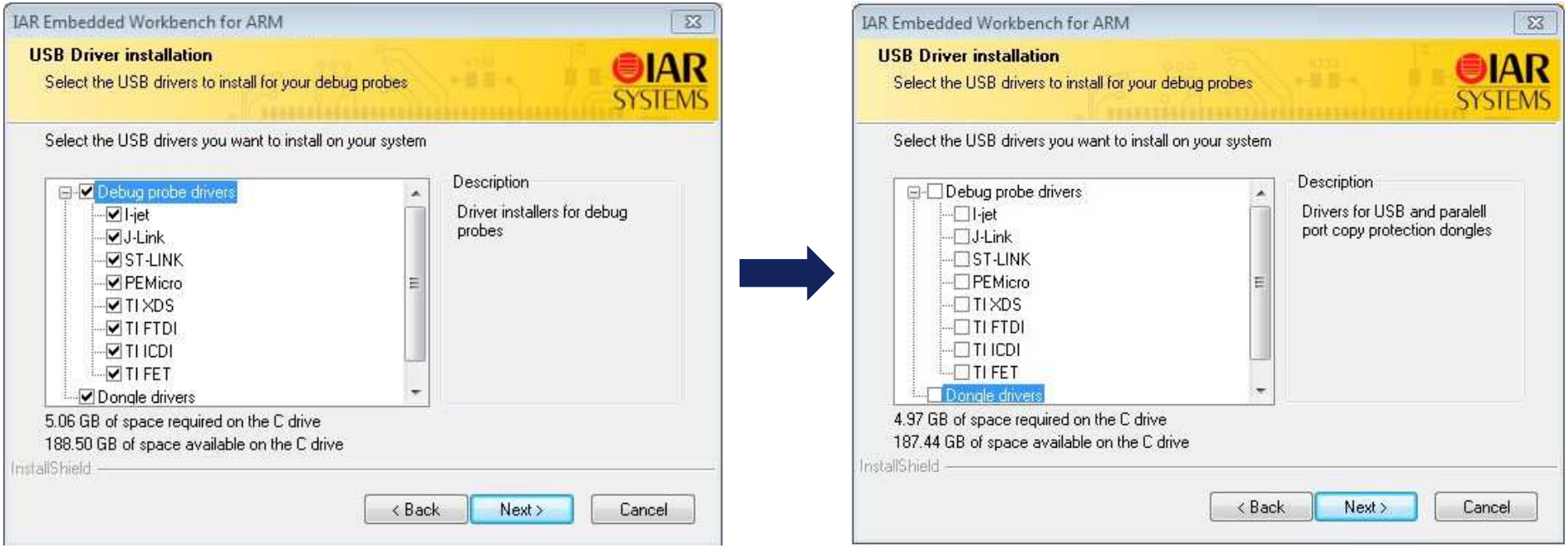
IAR Installation

- Run the IAR professional tool suite installer:
  `C:\STM32L4_DK_IoT_Node_Seminar\IAR`

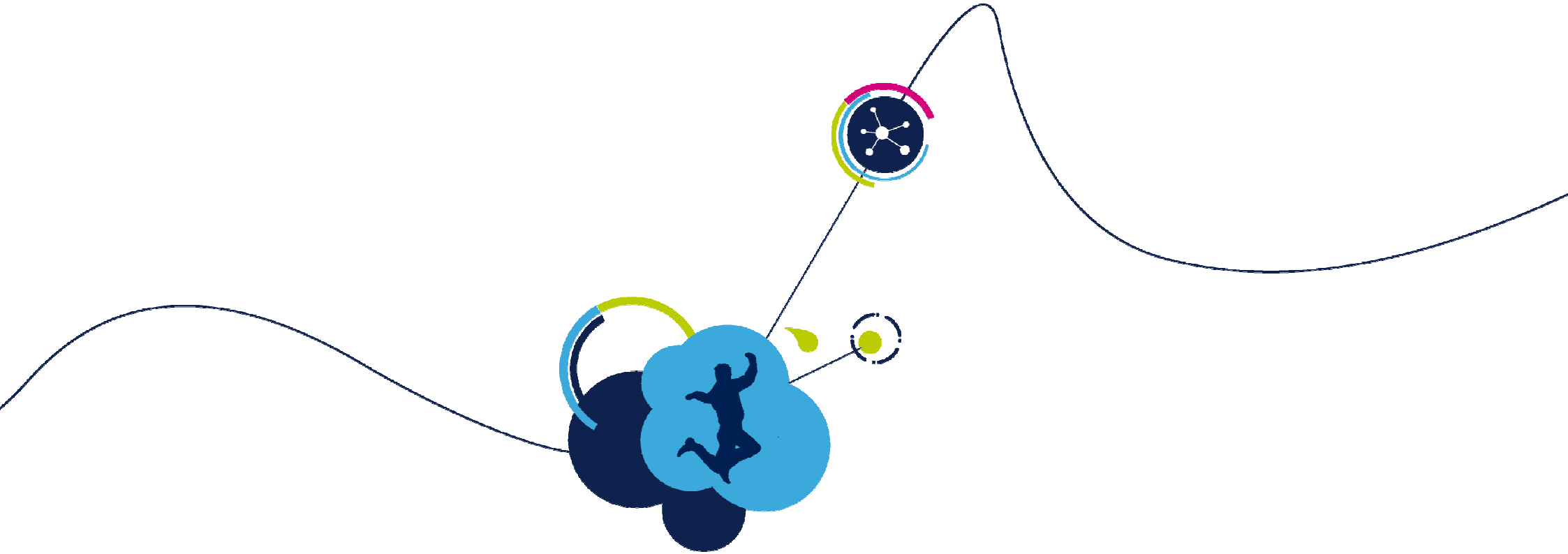
- From the installer menu select Install IAR Embedded Workbench.

IAR USB Driver Installation

- De-select **ALL** the USB drivers when IAR prompts you:



life.augmented



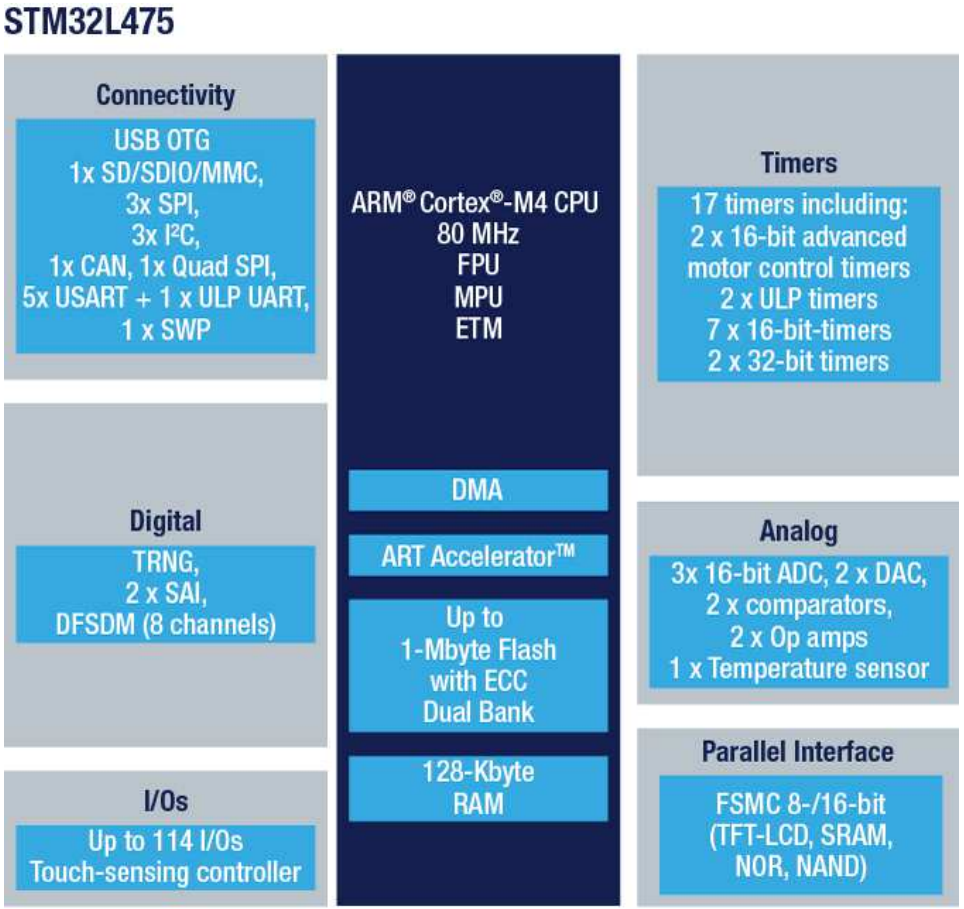
Overview of the STM32L475

life.augmented

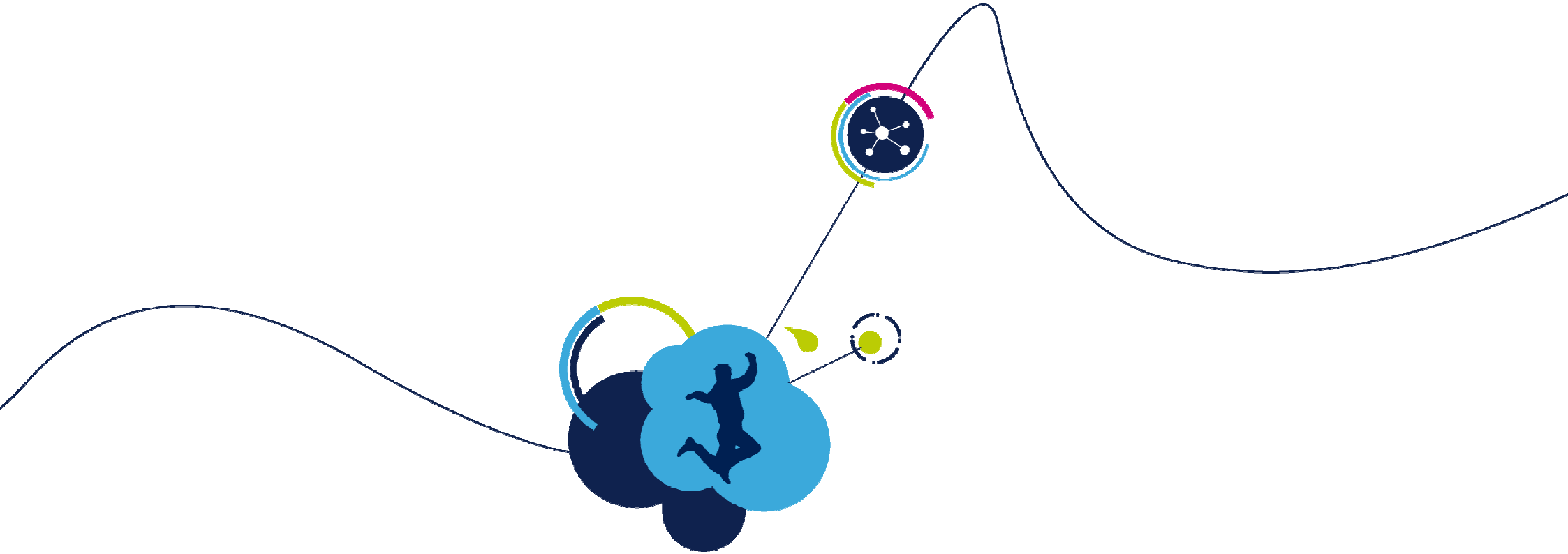
Overview of the STM32L475

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

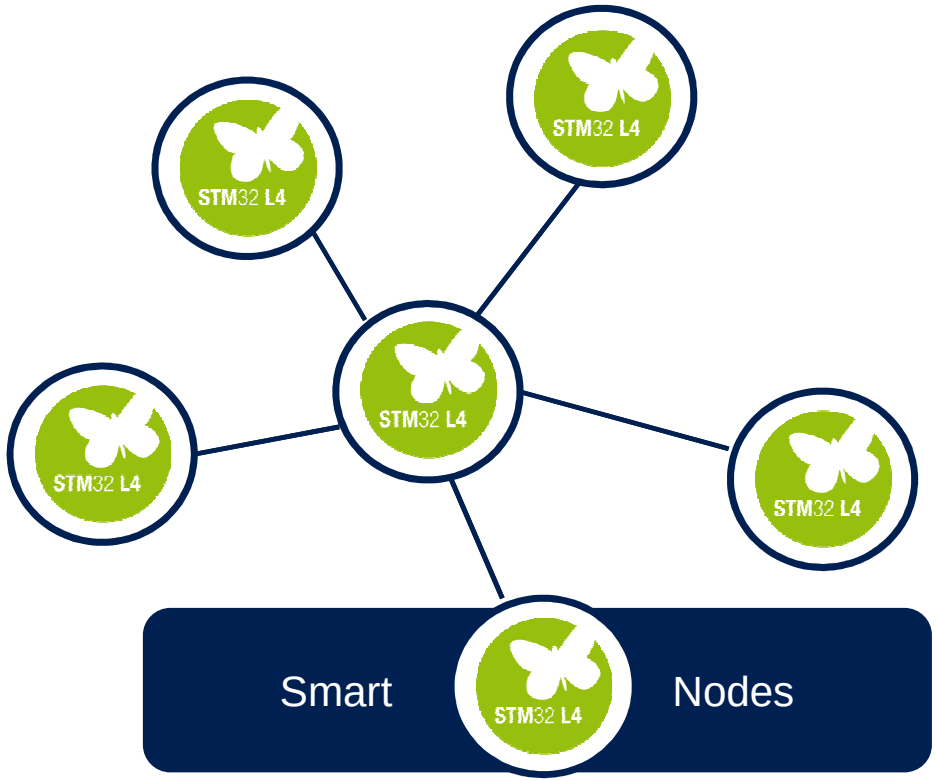


Overview of the STM32L4 Discovery Kit IoT Node

life.augmented

STM32L4 Discovery Kit IoT Node

Get connected seamlessly!



Network Servers

life.augmented

STM32L4 Discovery Kit IoT Node

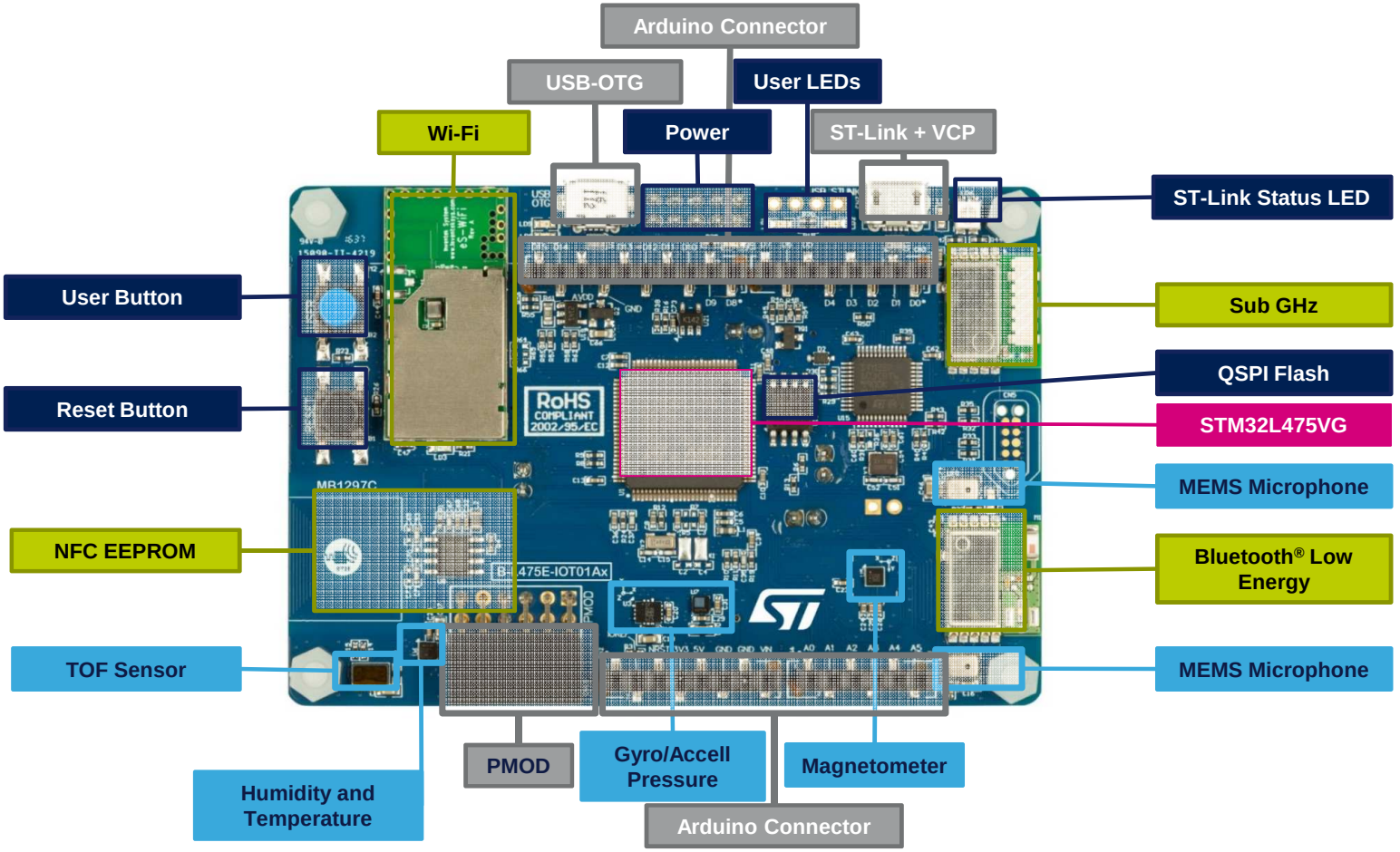
Open the door to remote services

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

life.augmented

STM32L4 Discovery Kit IoT Node

Multi-link communication, multiway sensing

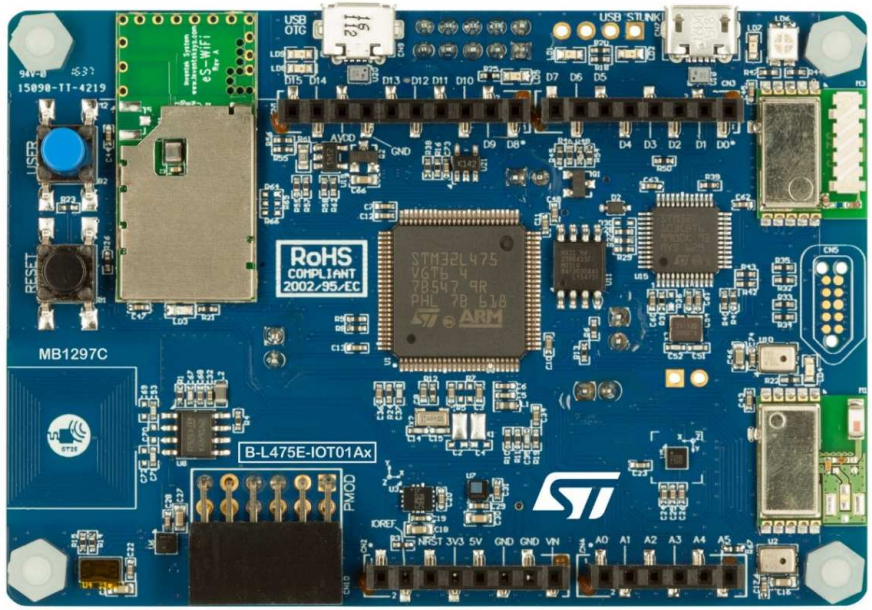


life.augmented

Comprehensive Software Libraries

Instant showcase

- SW Libraries for STM32L4 MCU & sensors
- Connectivity SW protocol stacks
- Cloud service connectors (AWS)
- Demo examples (X-CUBE-AWS)



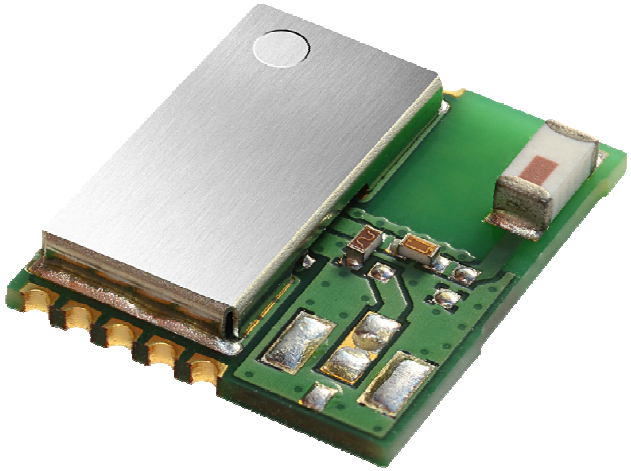
life.augmented

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



life.augmented

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

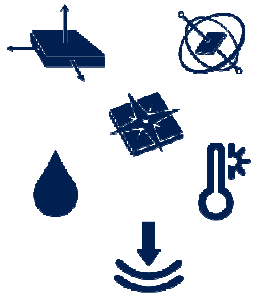
Wired Connectivity Features

- 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)

life.augmented

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

life.augmented

User Resource Features

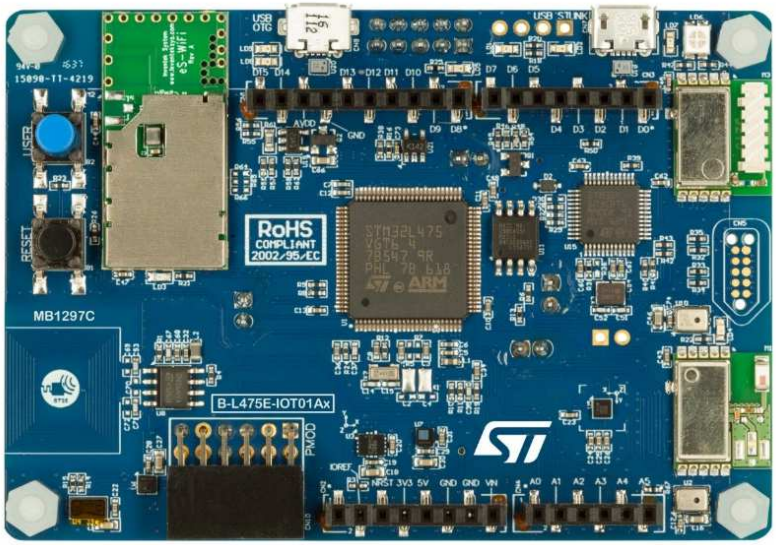
- Reset and User Buttons
  - Board Reset and Programmable Application Buttons
- User LEDs
  - Programmable Application LEDs
- QSPI Flash
  - 64Mbit for Data Storage and Program Execution
- Selectable Power Supply
  - ST-Link, USB-OTG, Arduino or External Power

Advantages of a Single Board

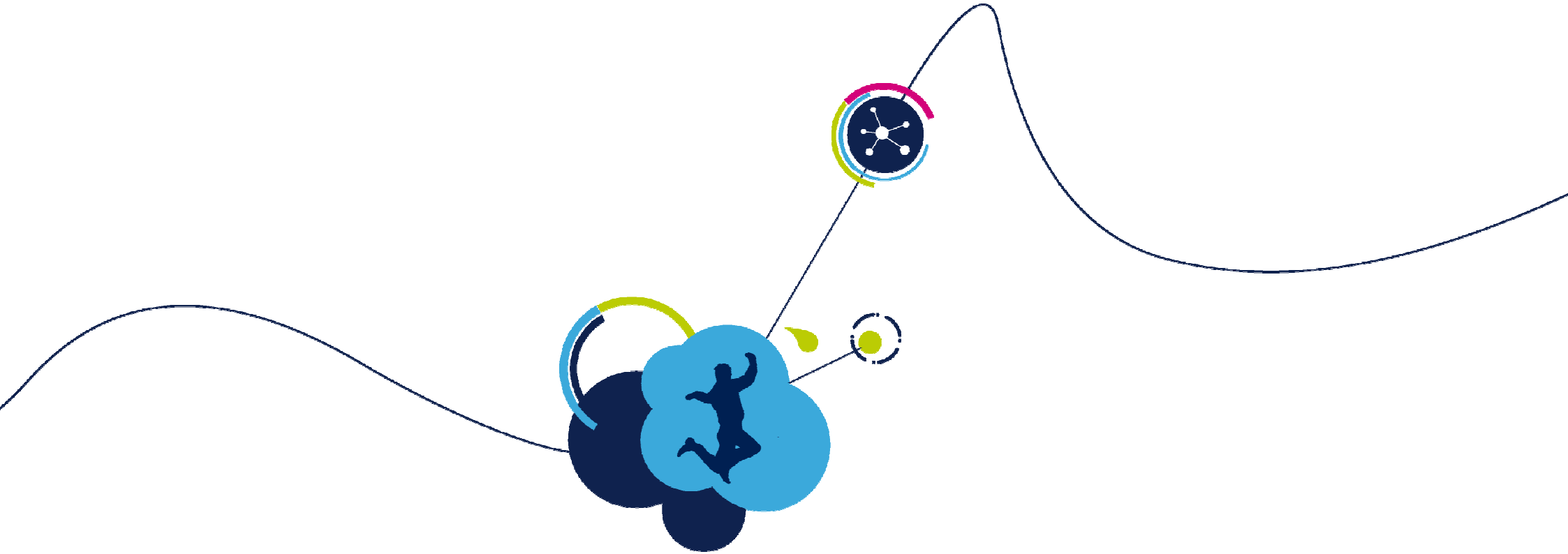
- Easily Debug Hardware Issues.
- Included Collateral is Tightly Coupled
  - BSP Included for all Board Components
  - Cloud Connectivity Reference Solutions Included
- Cost Effective Development Solution (~$60)
- No Need to Manage & Order Multiple Board SKUs.

Availability

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



life.augmented

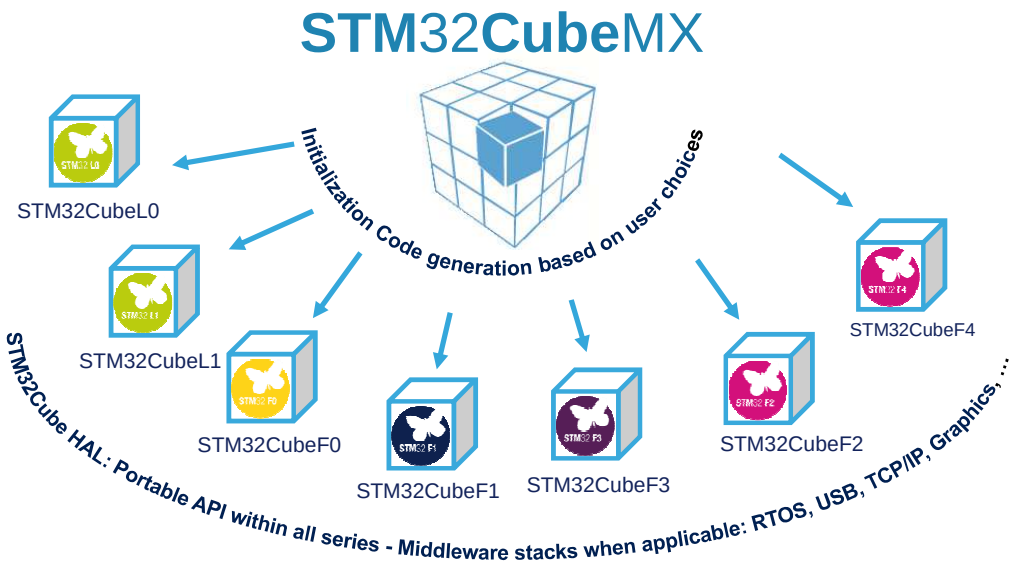


STM32Cube™ Introduction

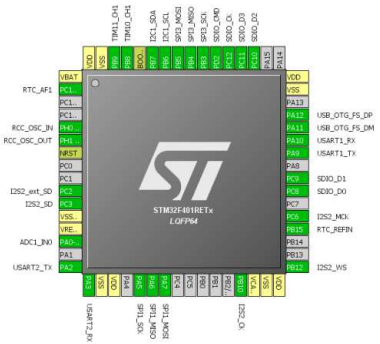
life.augmented

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



Pinout Wizard

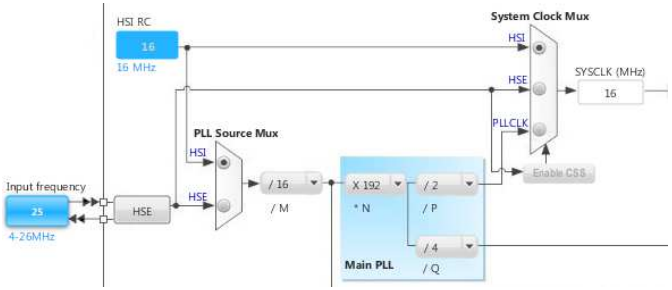
STM32CubeMX

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

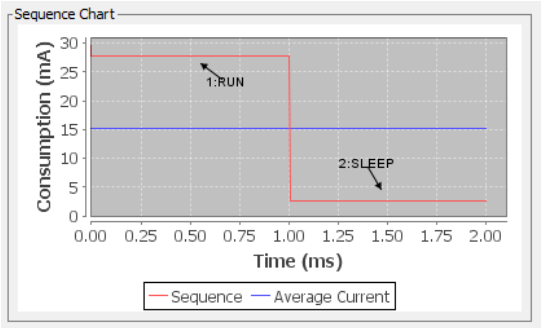
Baud Rate
BaudRate must be between **110 Bits/s** and **10.5 MBits/s**.

Peripherals & Middleware Wizard

Clock Tree wizard

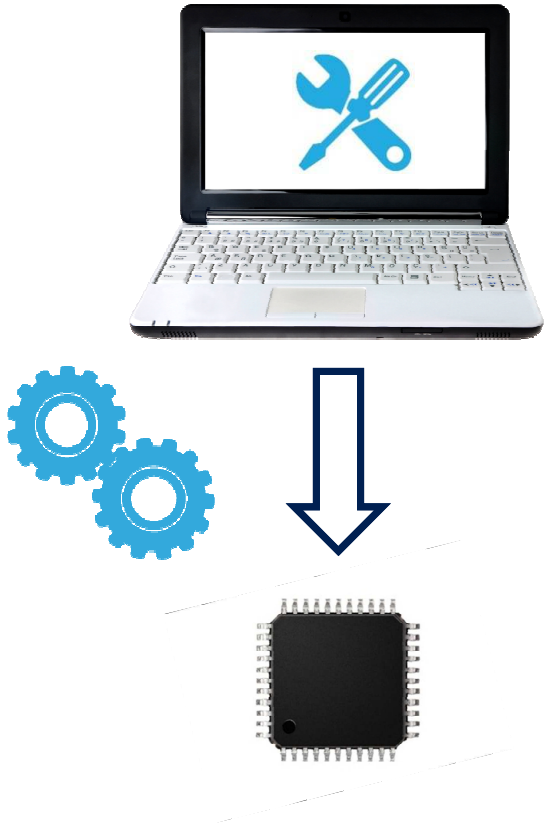


Power Consumption Wizard



life.augmented

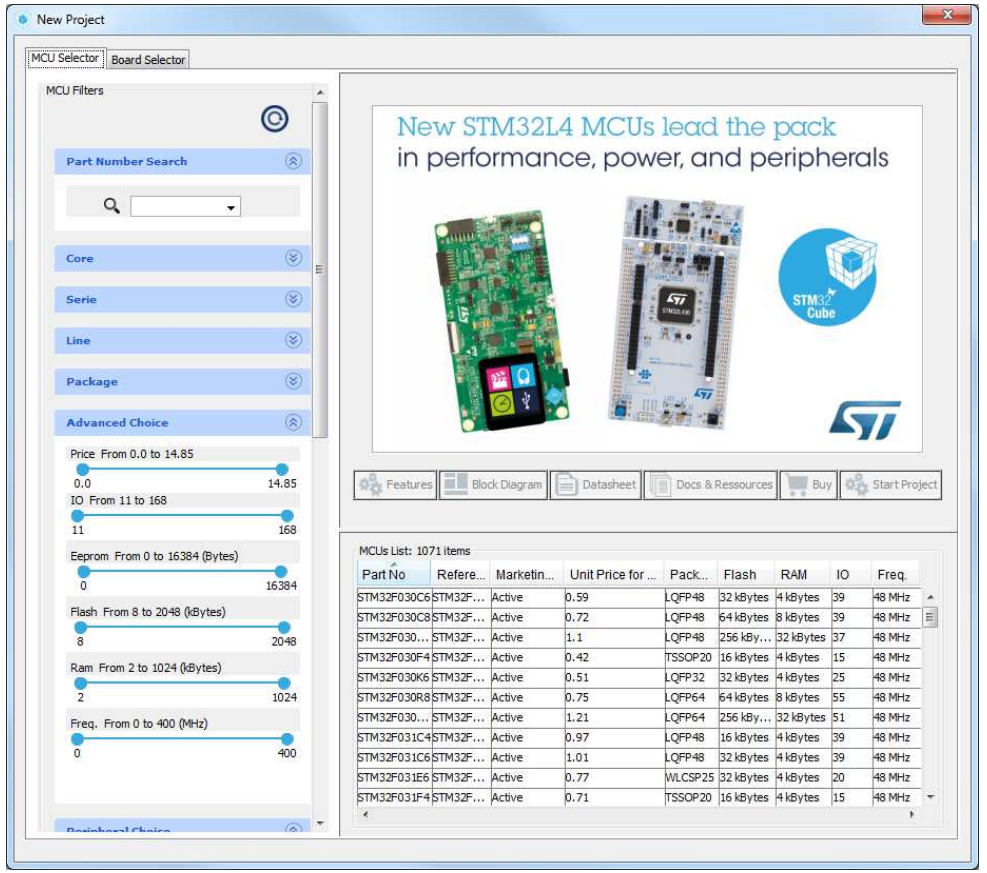
STM32CubeMX : User Code Configuration



Generates Initialization C Code based on user driven configuration!

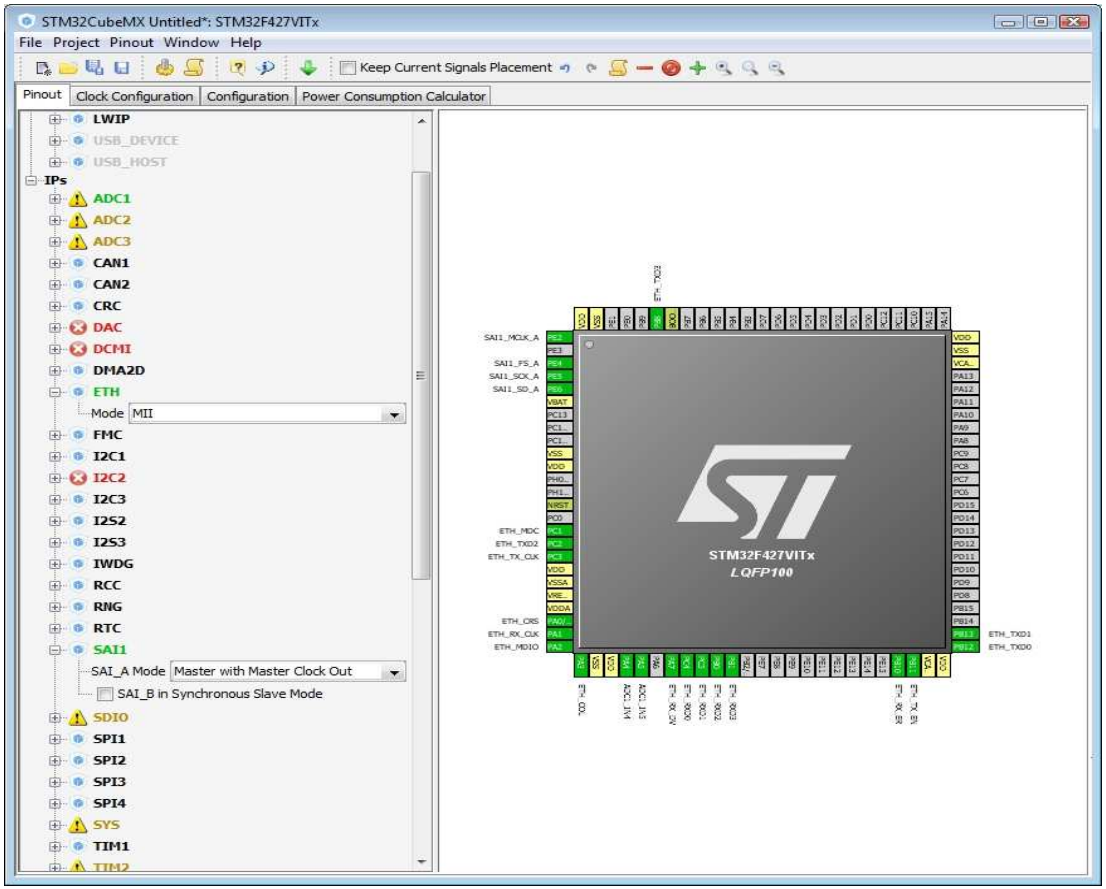
STM32CubeMX : MCU Selector

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



STM32CubeMX : Pin-out Configuration

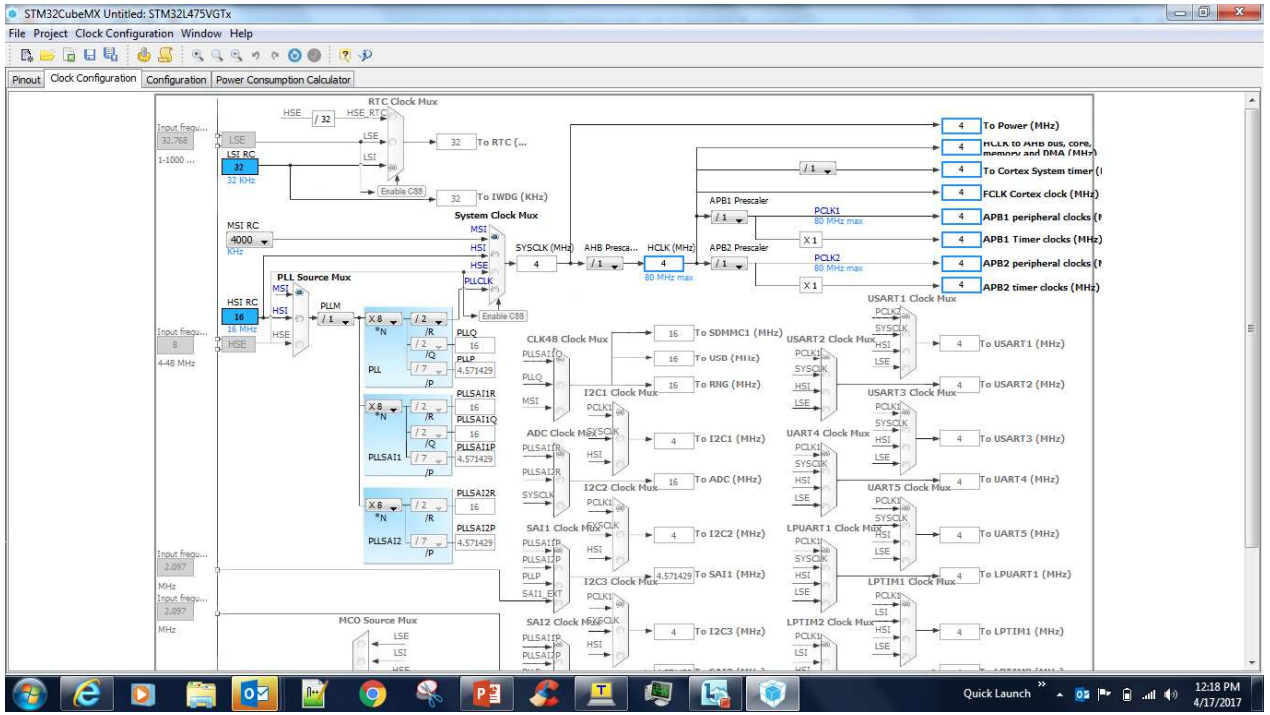
- Pinout from:
  - Peripheral tree
  - Manually
- Automatic signal remapping
- Management of dependencies between peripherals



life.augmented

STM32CubeMX : Clock Tree

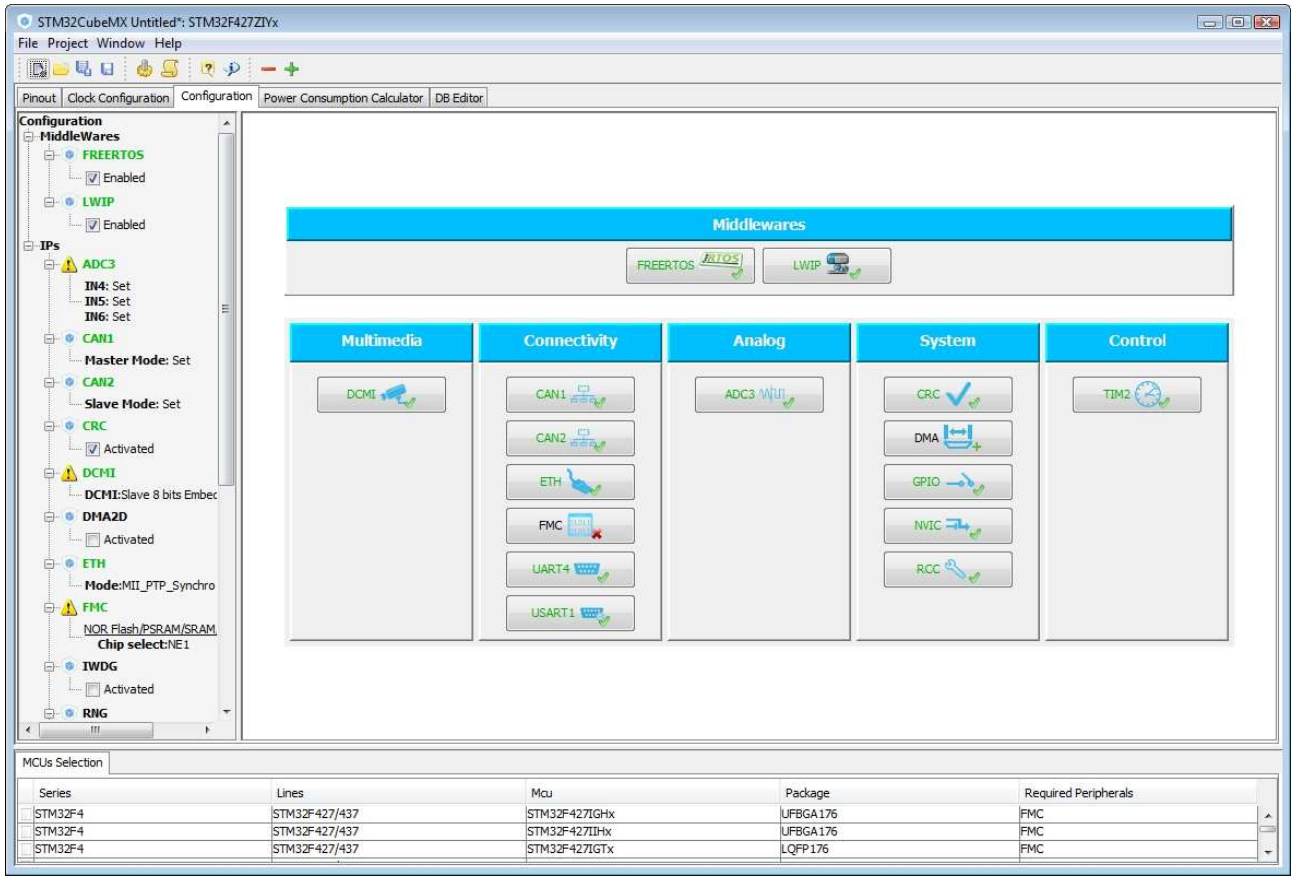
- Immediate display of all clock values
- Management of all clock constraints
- Highlight of errors



life.augmented

STM32CubeMX : Peripheral Configuration

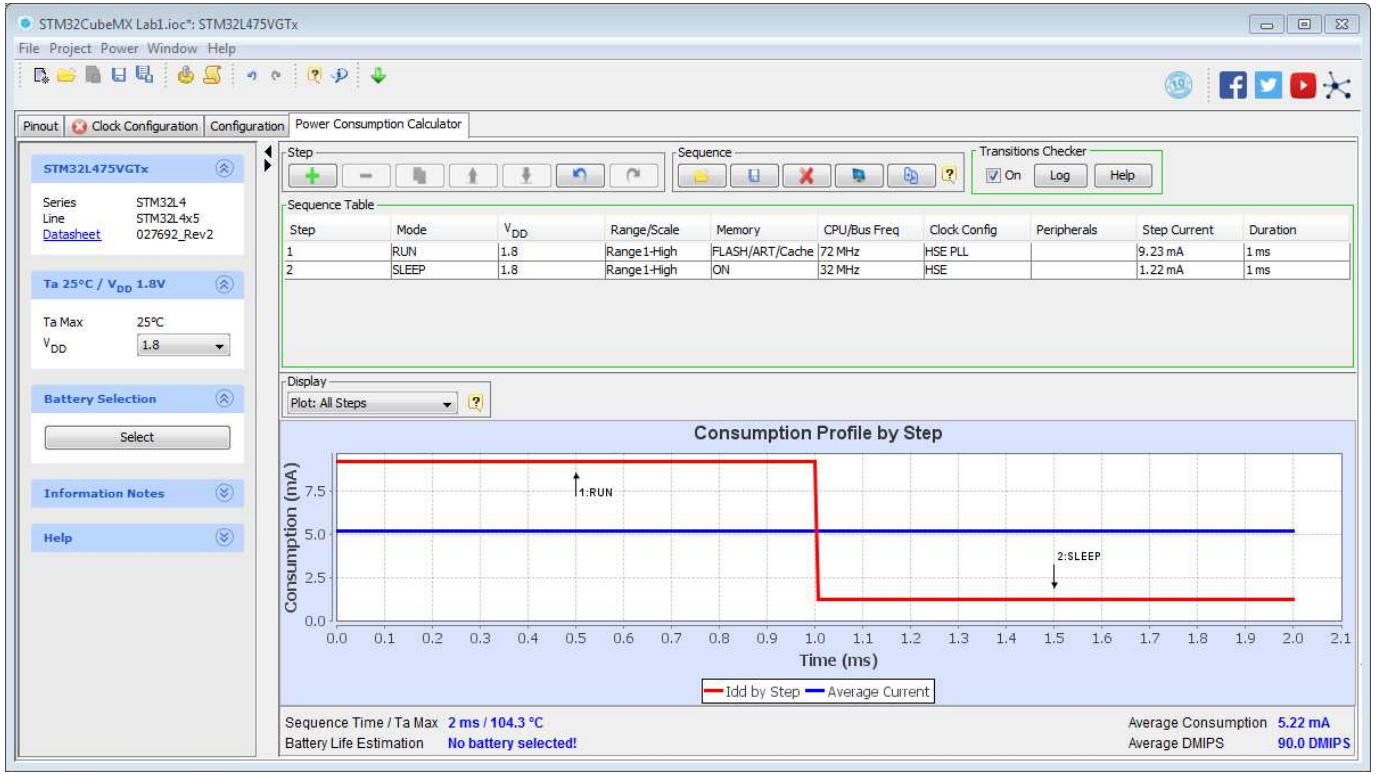
- Global view of used peripherals and middleware
- Highlight of configuration errors
- Manage:
  - GPIO
  - Interrupts
  - DMA



life.augmented

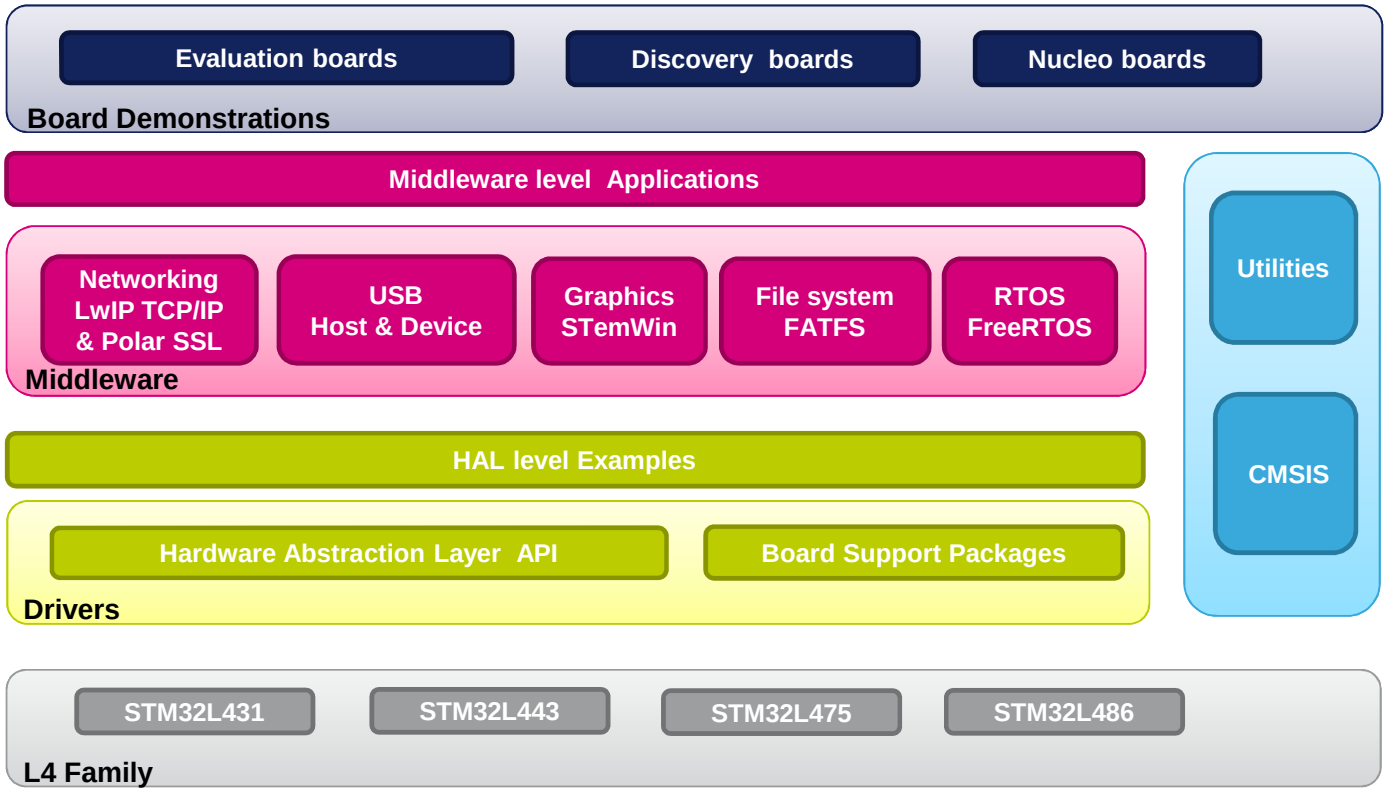
STM32CubeMX : Power Consumption Calculator

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

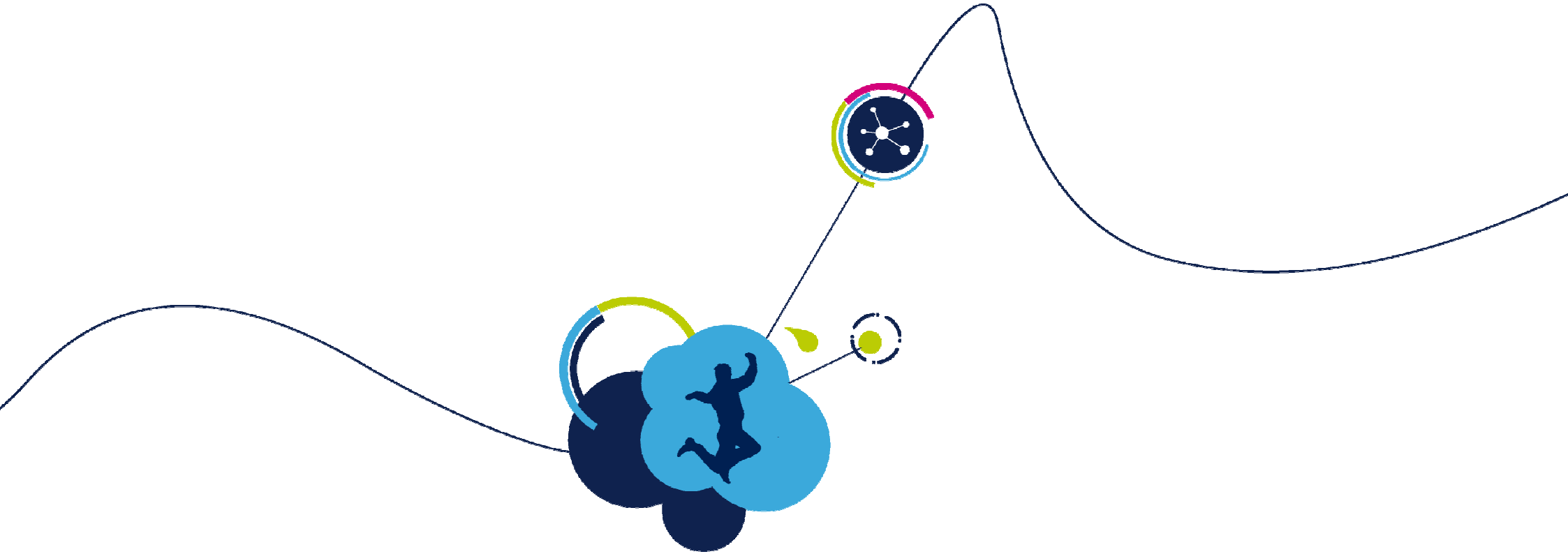


life.augmented

STM32Cube : Firmware Components



life.augmented



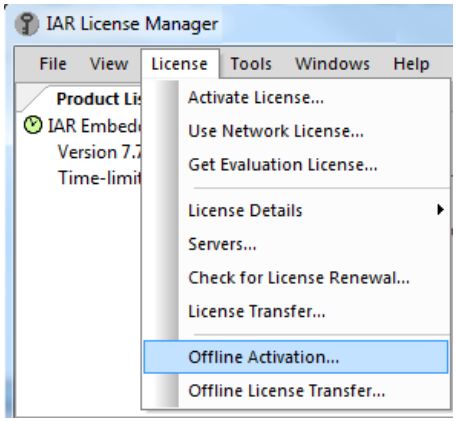
IAR License Installation

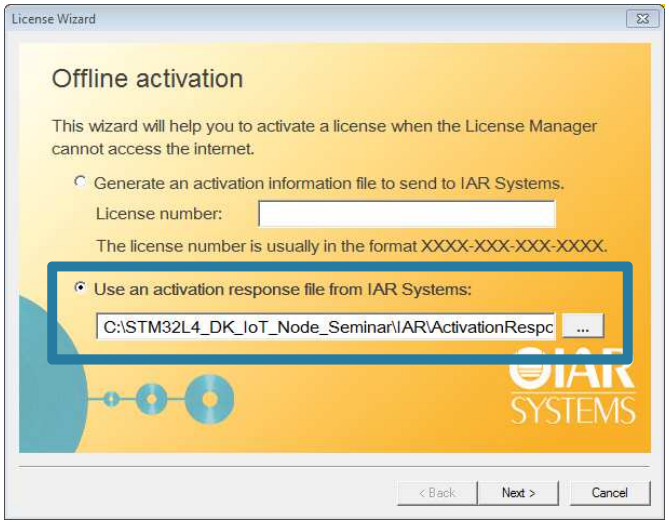
life.augmented

IAR License Installation

- Open IAR
- Go to `Help->License Manager`
- Go to `License->Offline Activation…`
- The activation response file is located here:

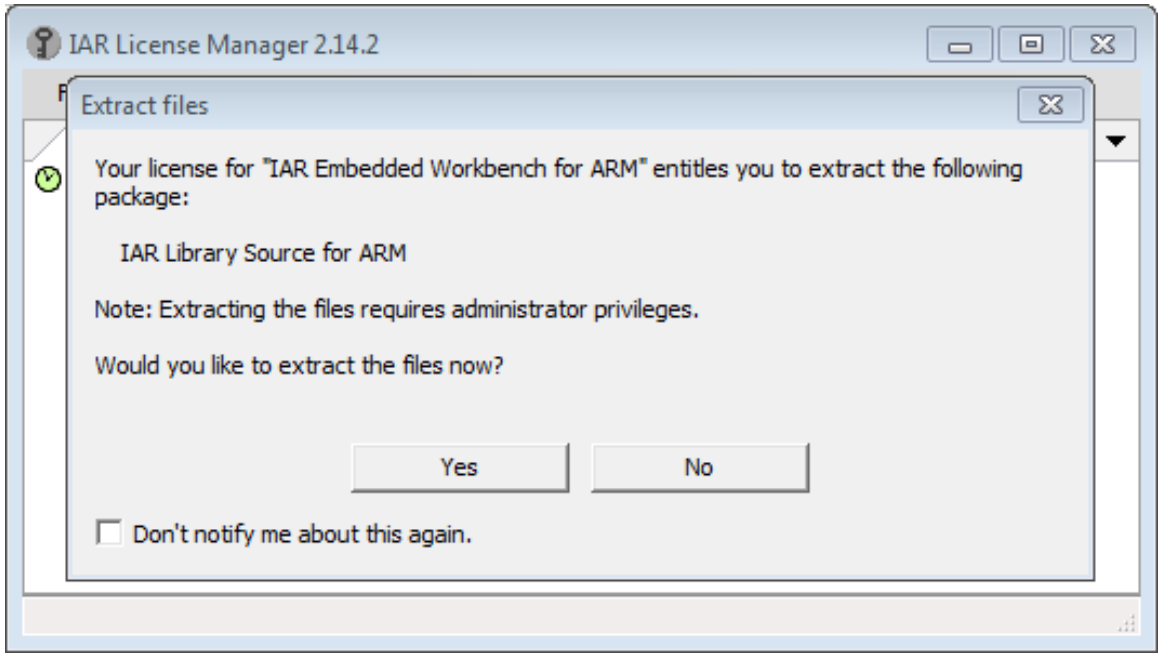
`C:\STM32L4_DK_IoT_Node_Seminar\IAR\ActivationResponse.txt`





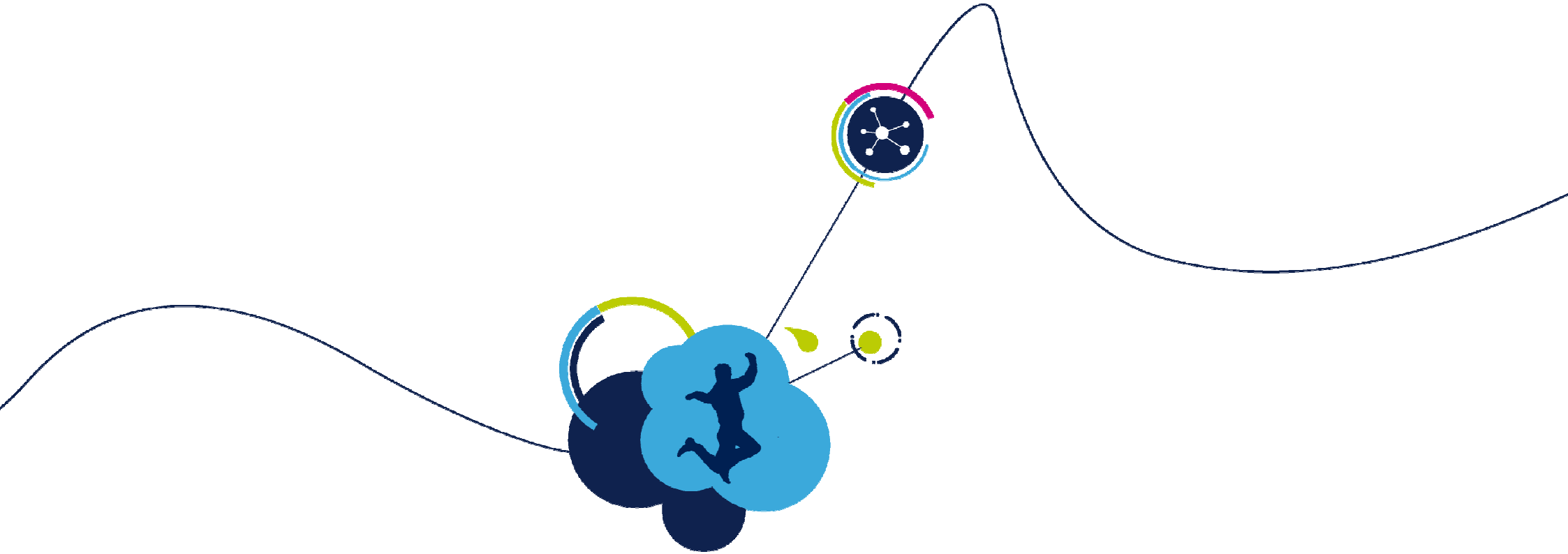
life.augmented

IAR License Installation



If you see this pop-up, select "Don't notify me about this again." and click on "No".

life.augmented



ST-Link Utility Installation

life.augmented

ST-Link Utility Installation

- The ST-Link Utility provides typical flash program / erase / upload / download functions via the ST-LINK/V2 debugger on-board the STM32L475 Discovery Kit IoT Node Board. It also installs the Windows device drivers necessary for the ST-LINK/V2 debugger.

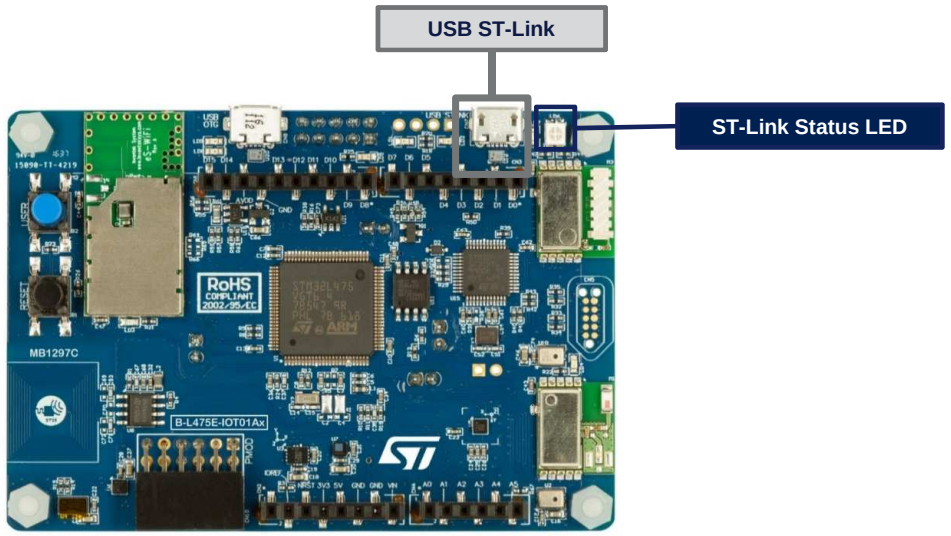
- The ST-Link installer is located here:

C:\STM32L4_DK_IoT_Node_Seminar\Software

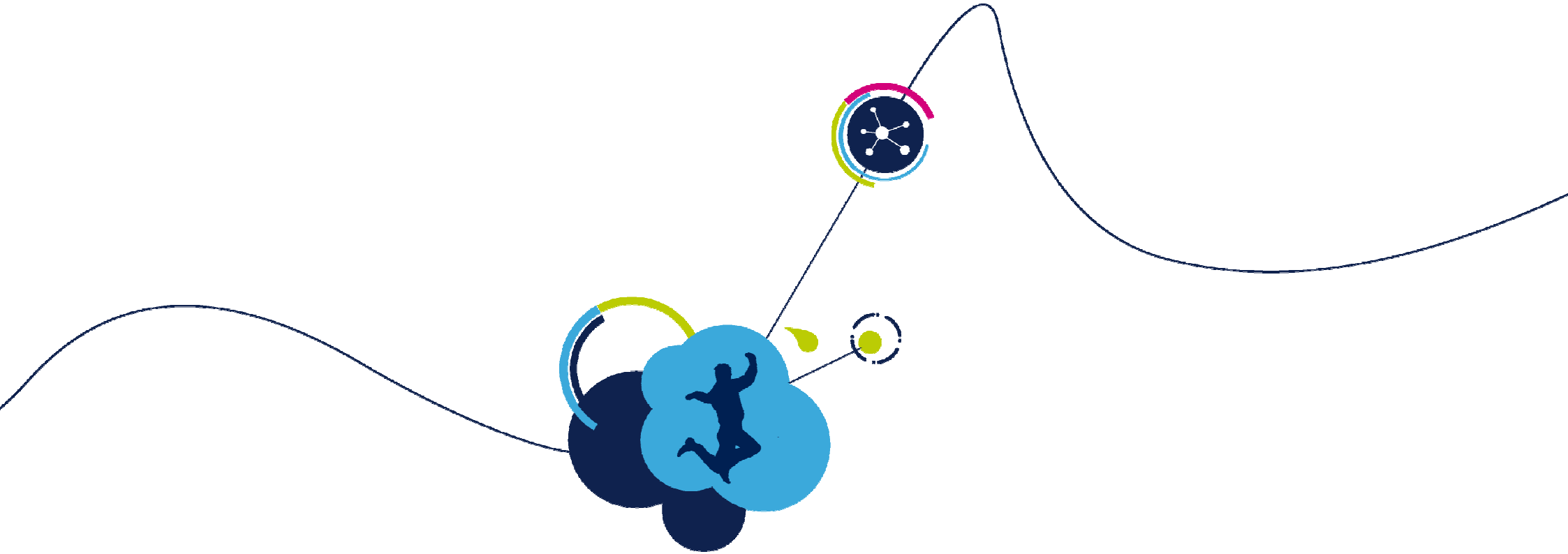
STM32_ST-Link_Utility-4.0.0.exe

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.



life.augmented



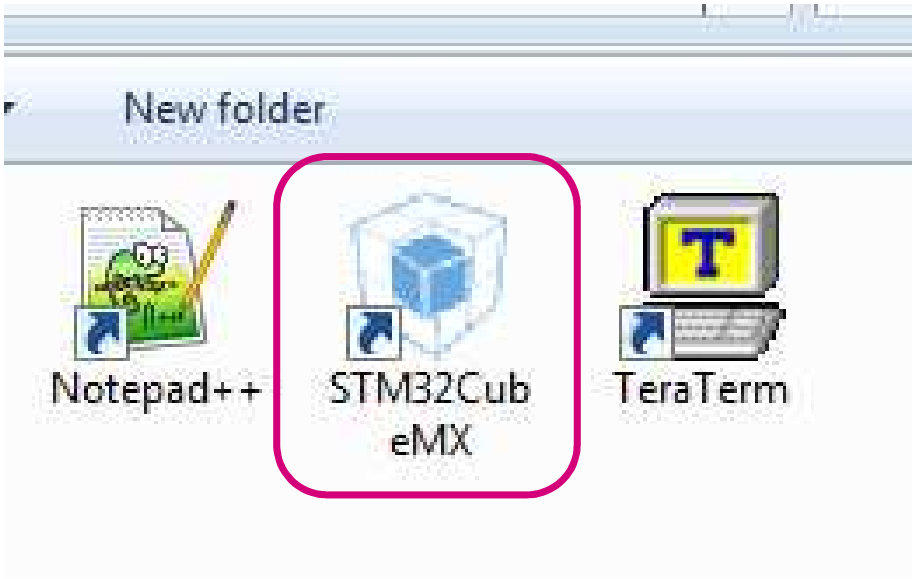
Lab 1 : Getting Started with STM32CubeMX – Blinky LED

life.augmented

Create New STM32CubeMX Project

- Open the **STM32L4 DK IoT Node Seminar** folder on your Desktop.

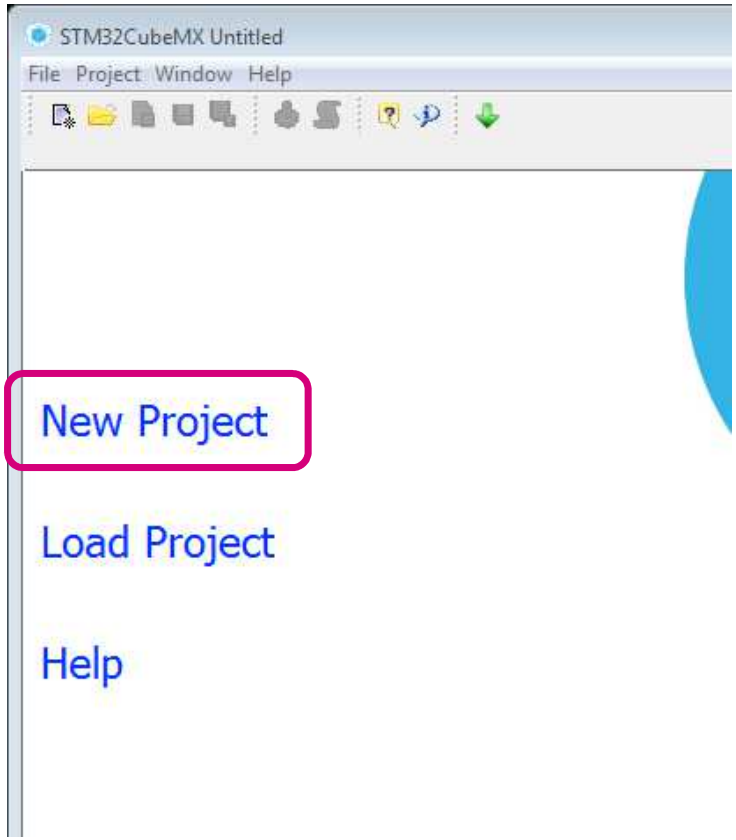
Create New STM32CubeMX Project



- Double-click on the **STM32CubeMX** shortcut

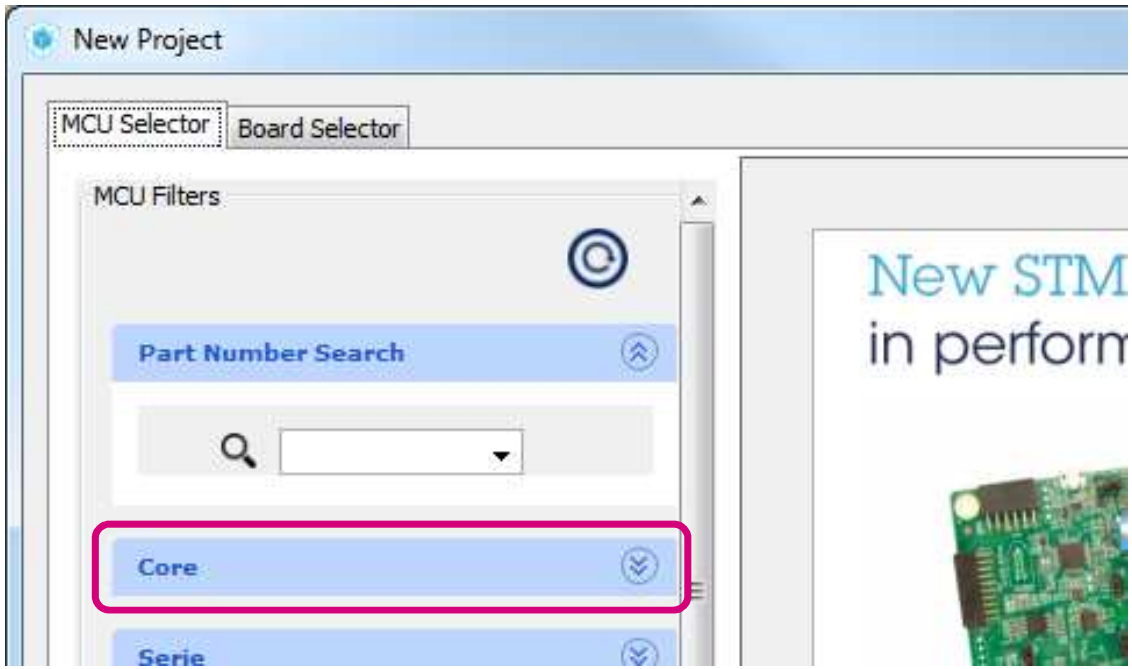
Create New STM32CubeMX Project

- Click on **New Project**



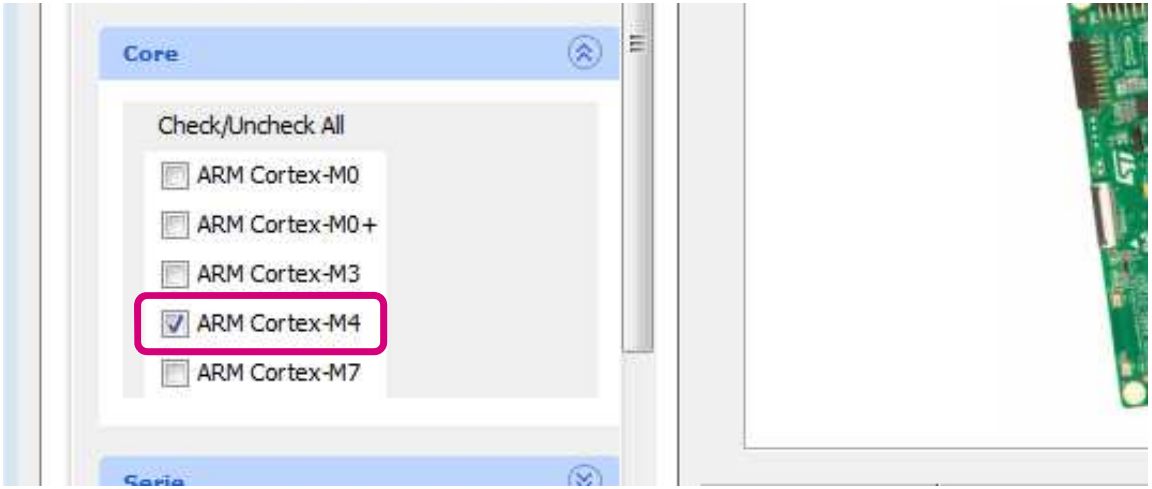
Select the Microcontroller

- Expand **Core**



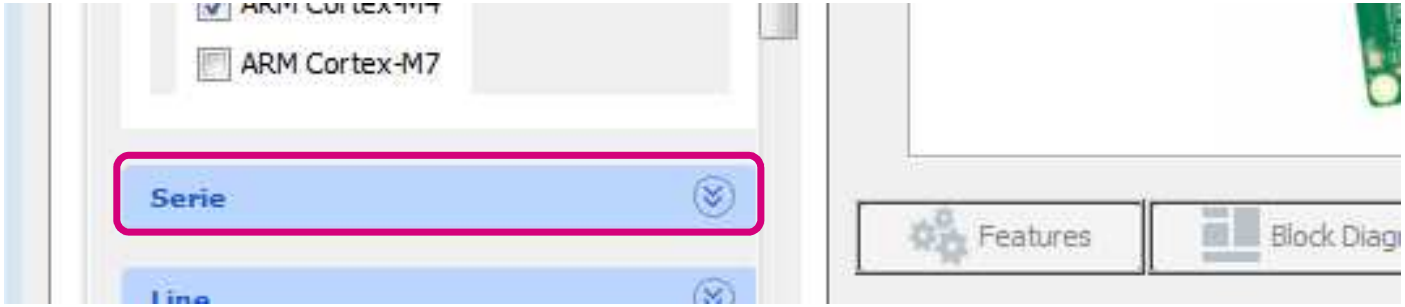
Select the Microcontroller

- Select **ARM Cortex-M4**



Select the Microcontroller

- Expand **Series**

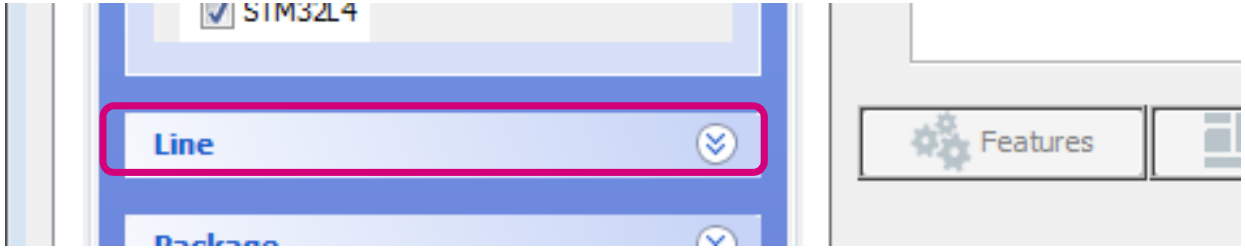


Select the Microcontroller

- Select **STM32L4**

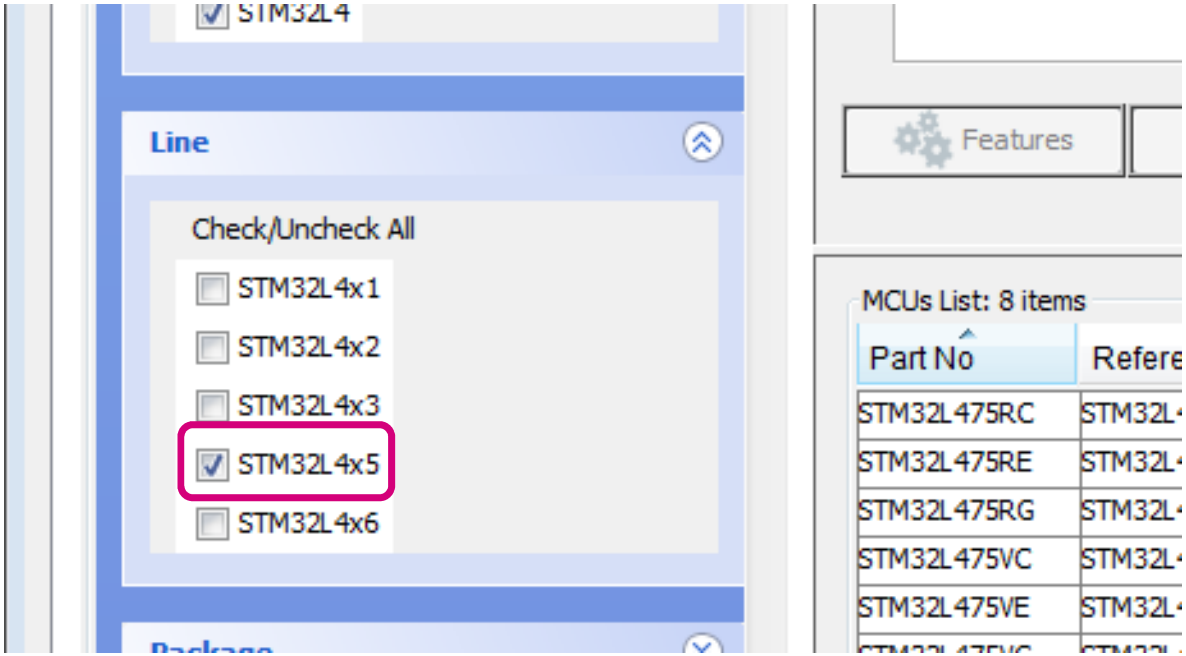
Select the Microcontroller

- Expand **Line**



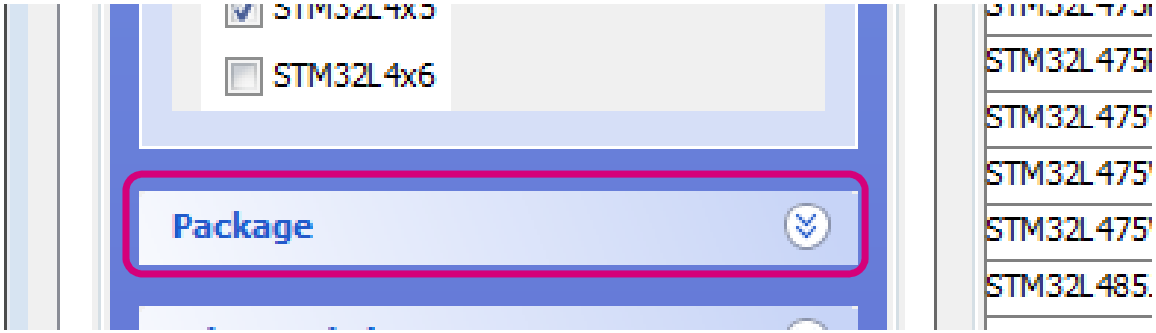
Select the Microcontroller

- Select **STM32L4x5**

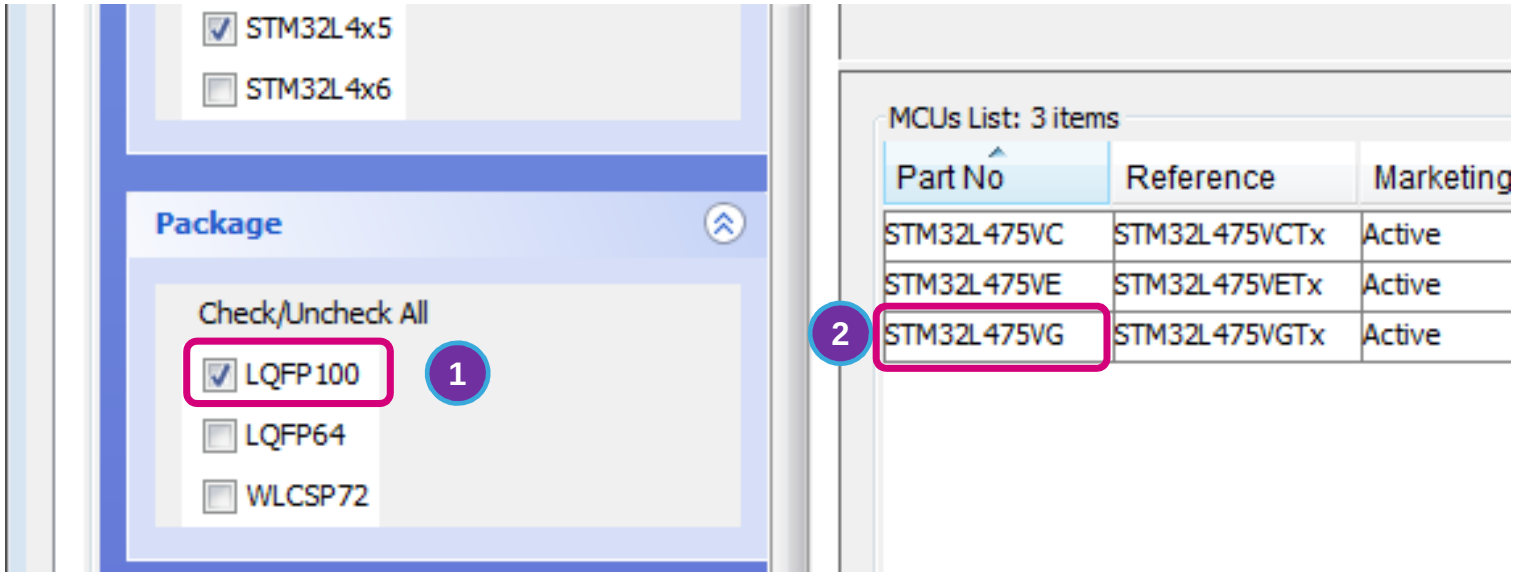


Select the Microcontroller

- Expand **Package**



Select the Microcontroller

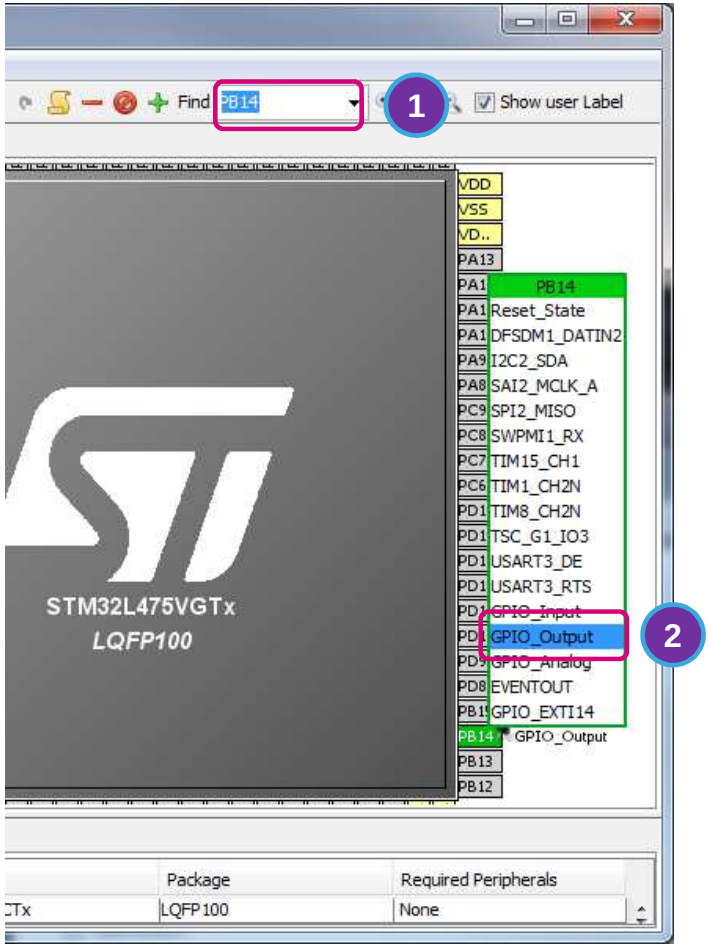


1. Select **LQFP100**
2. Double-click on **STM32L475VG**

GPIO Selection

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

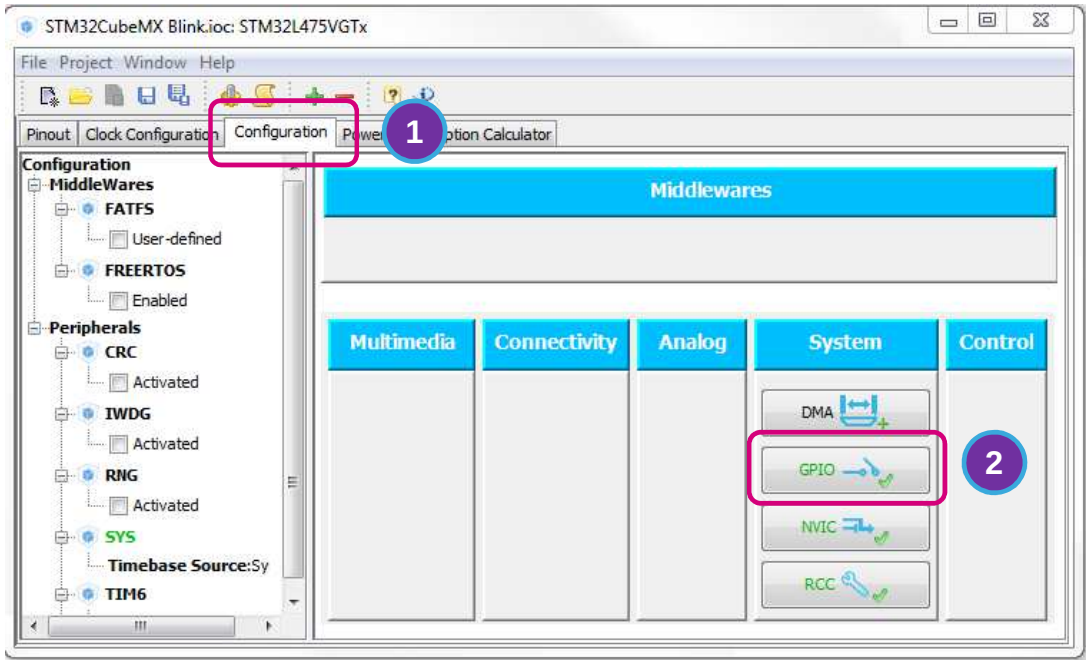
1. Use the find toolbar and type **PB14**
2. Select **PB14** and set it to **GPIO_Output** mode.



life.augmented

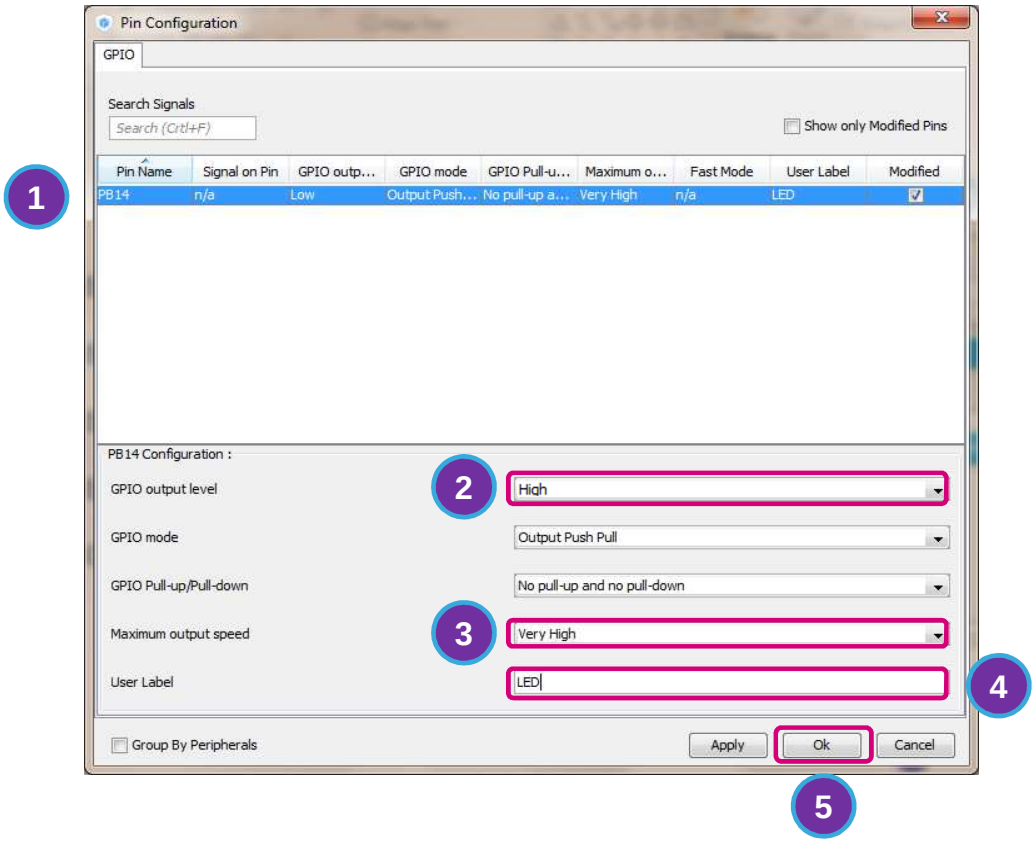
GPIO Configuration

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



GPIO Configuration

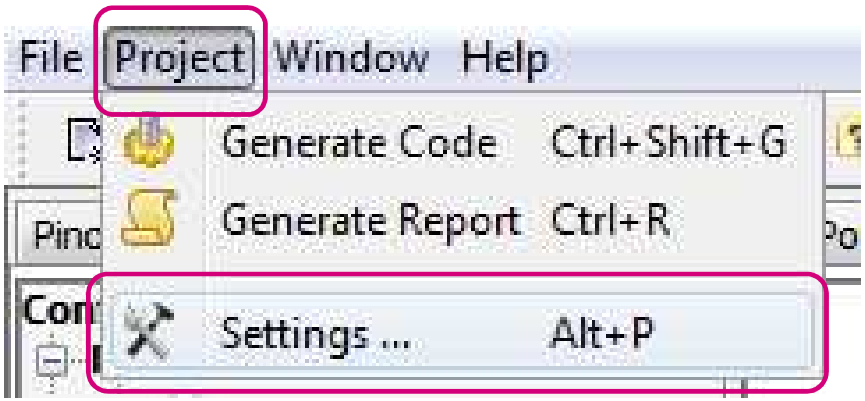
1. Select **PB14**.
2. Set the GPIO output level to **High**.
3. Set the Maximum output speed to **Very High.**
4. Set the User Label to **LED.**
5. Click OK



life.augmented

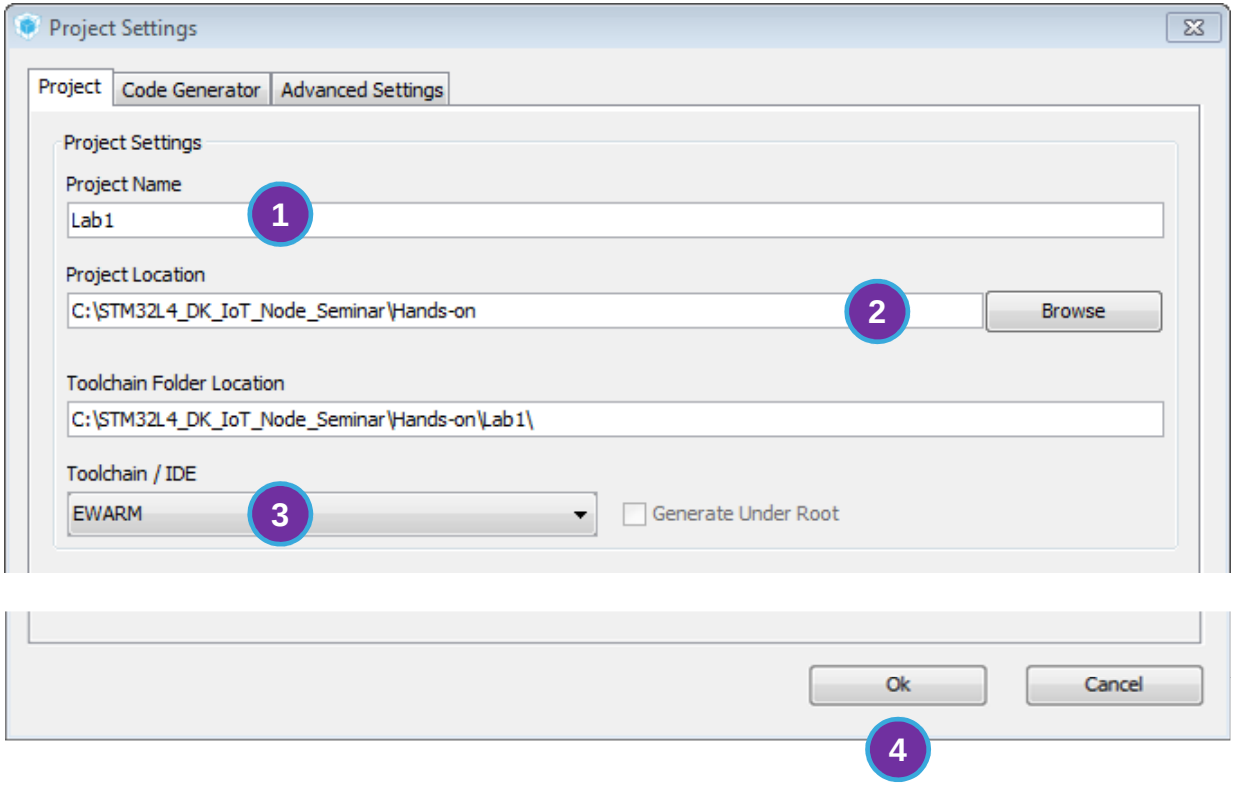
Project Settings

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

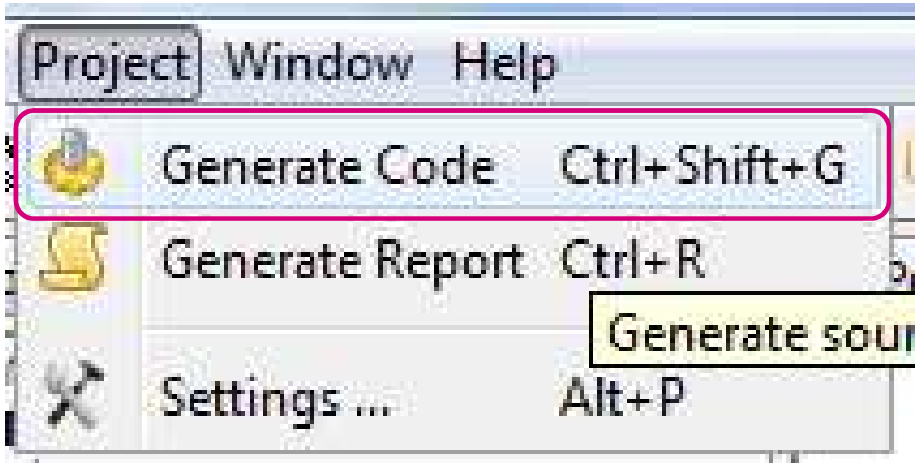


Project Settings

1. Set the project name to **Lab1**.
2. Set the project location: **C:\STM32L4_DK_IoT_Node_Seminar\Hands-on**
3. Set the IDE Toolchain to **EWARM**.
4. Click **OK**.

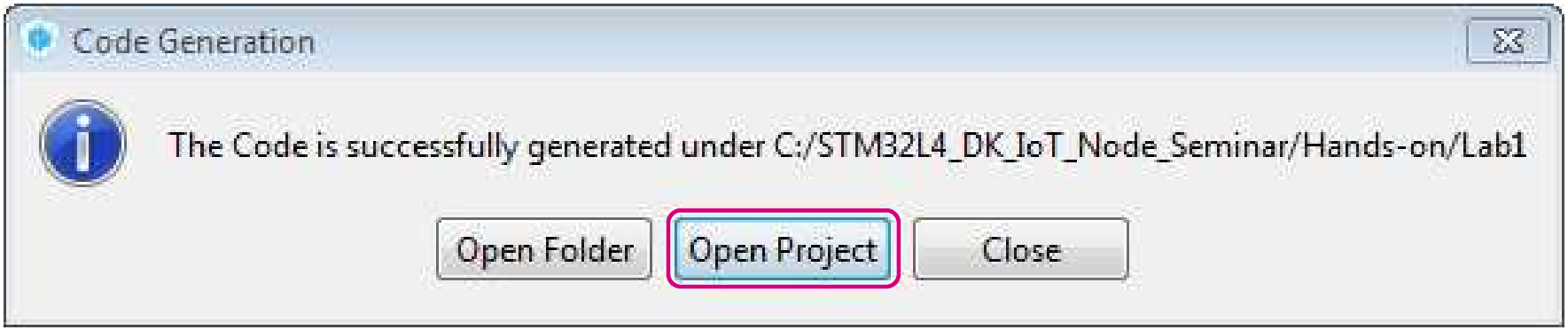


Generate the Project



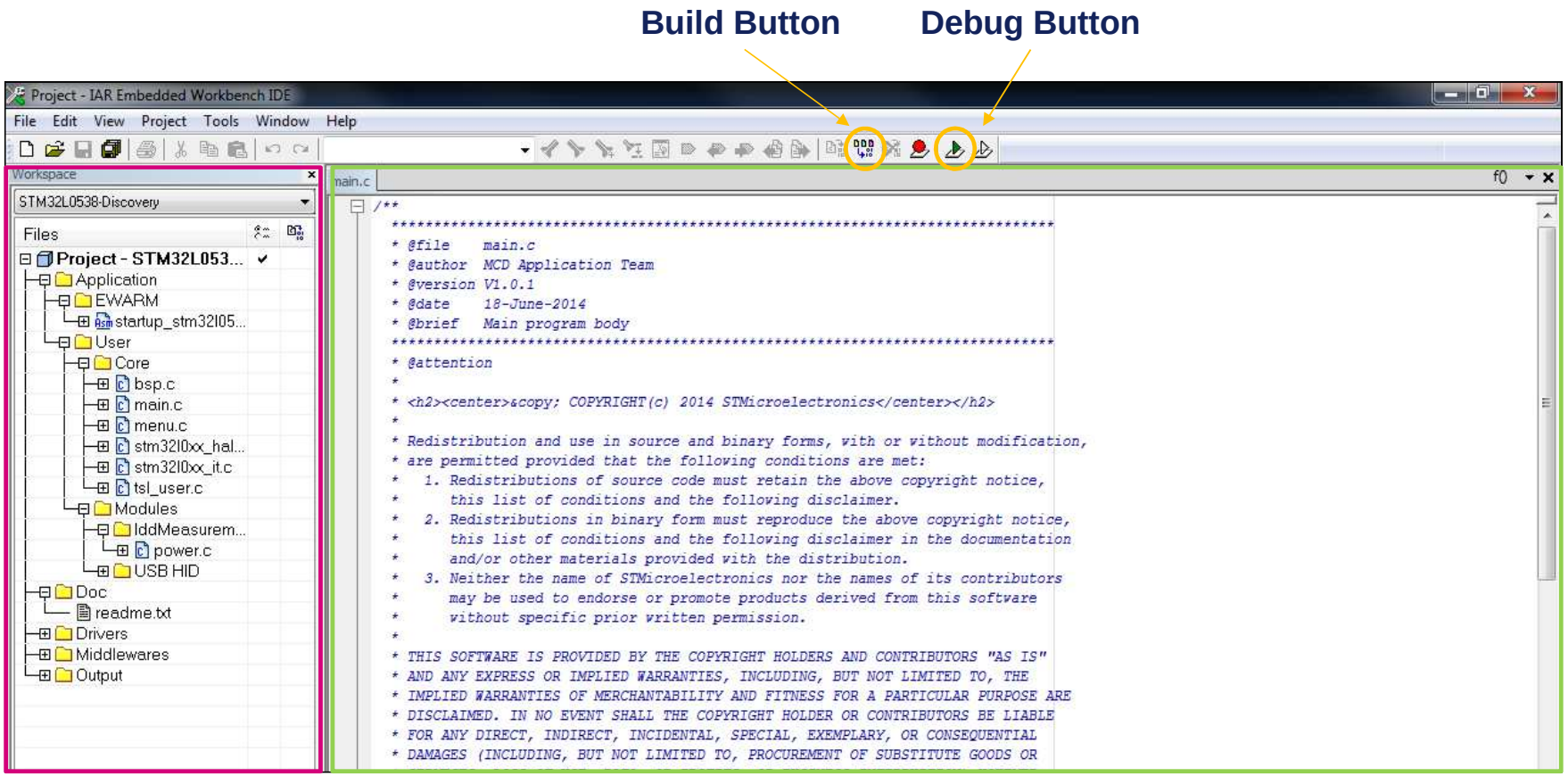
- Click **Generate Code** (Ctrl + Shift + G)

Open the Project



- Click **Open Project**

Inside IAR EWARM



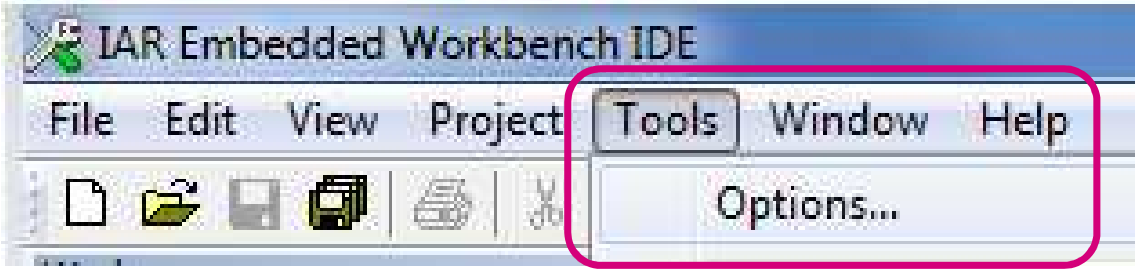
Project Window

Files Window

life.augmented

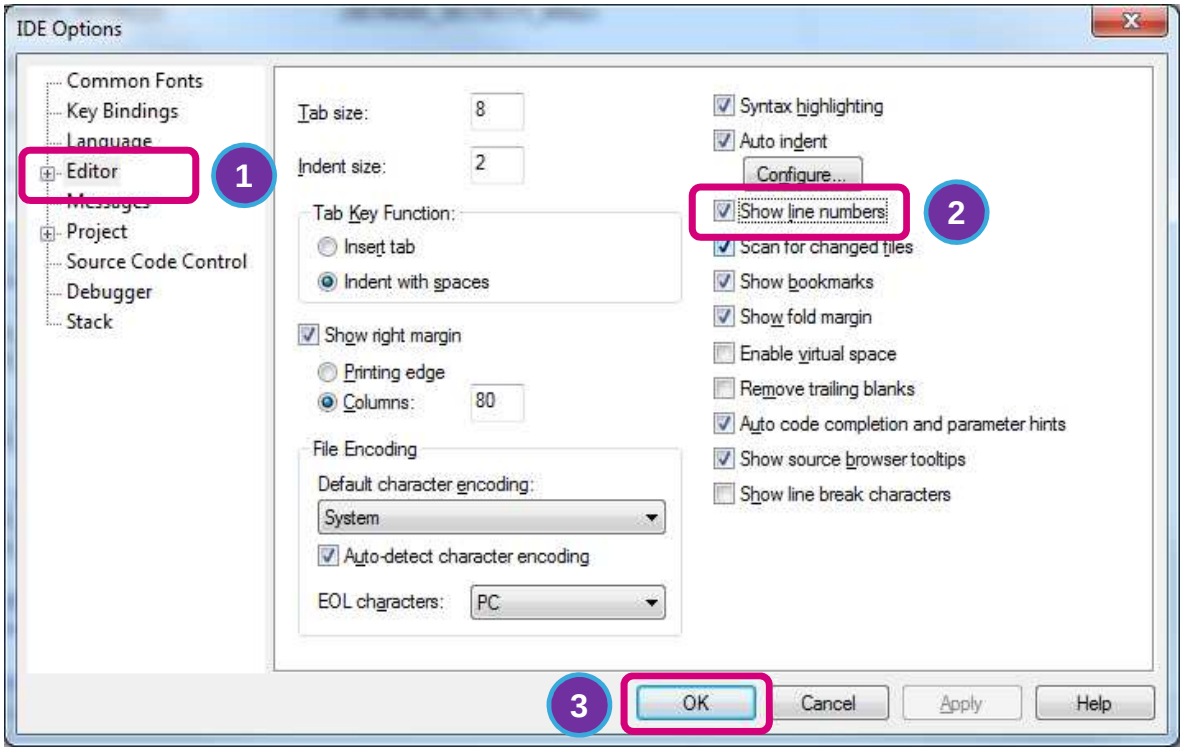
Configure IAR to Show Line Numbers

1. Go to Tools → Options



Configure IAR to Show Line Numbers

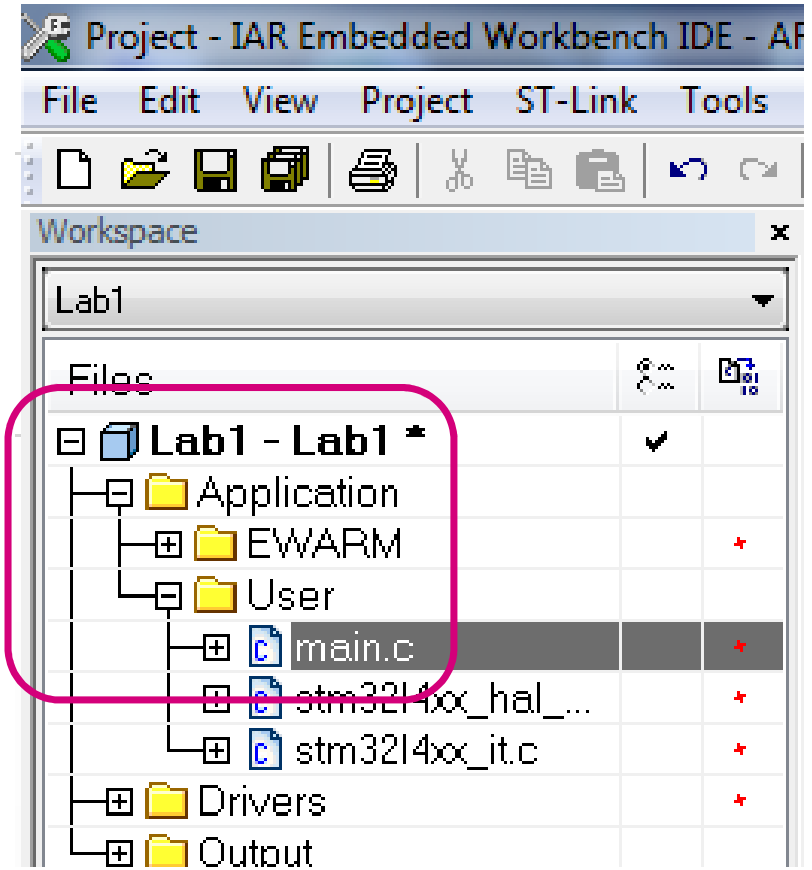
1. Select Editor
2. Check ‘Show Line Numbers’
3. Click OK



life.augmented

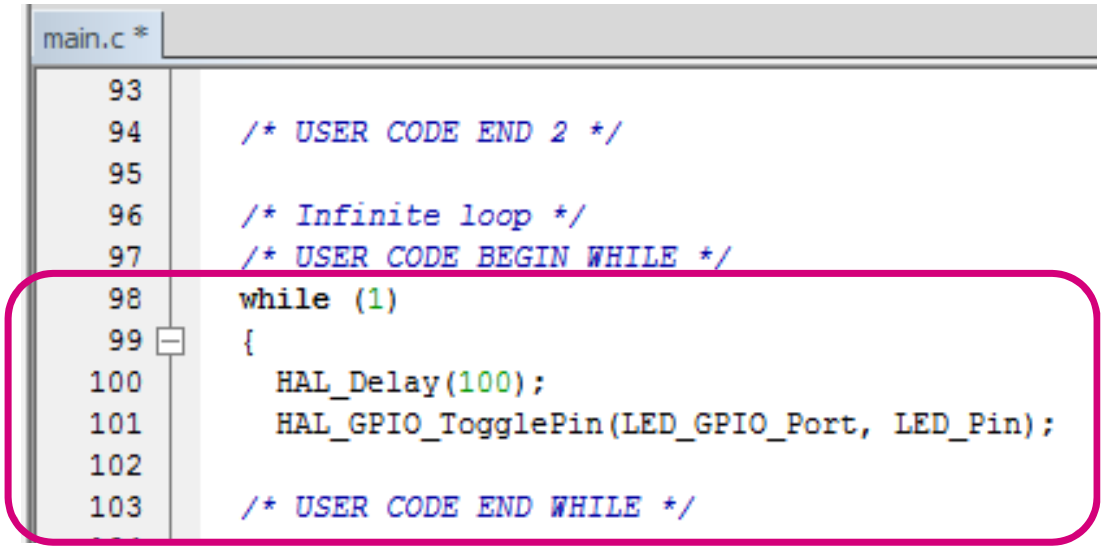
Edit main.c

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



life.augmented

Edit main.c

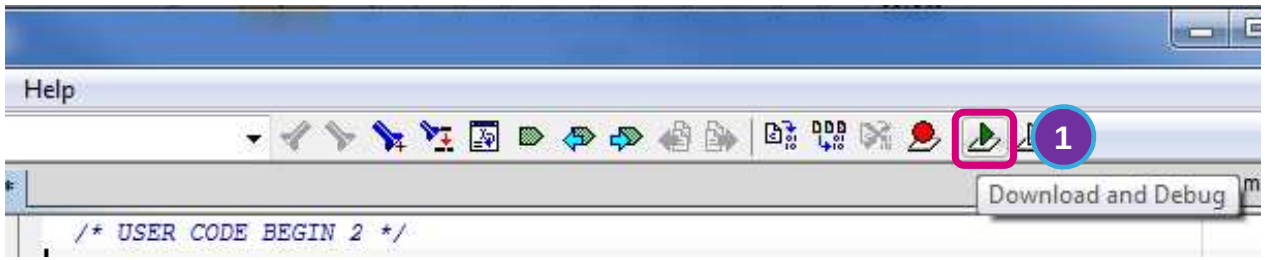


Add the following code inside the **while(1)** loop (line 98):

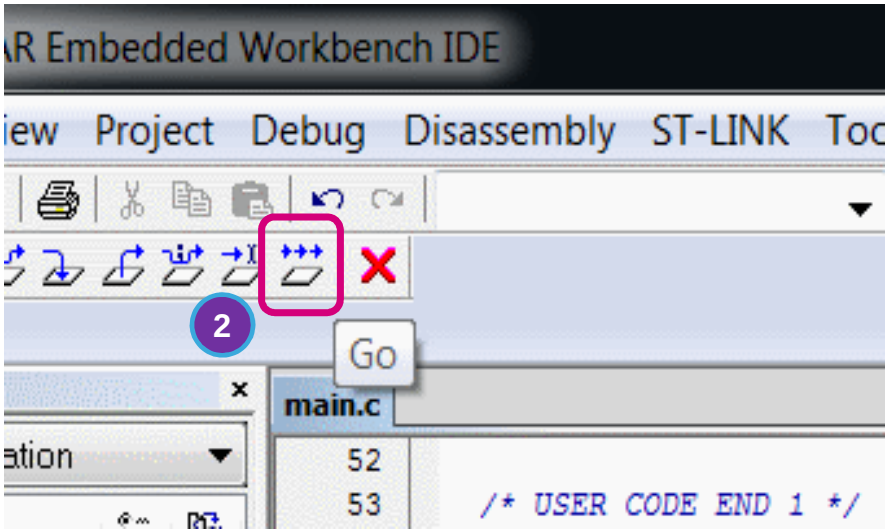
```
HAL_Delay(100);
HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
```

Load and Run

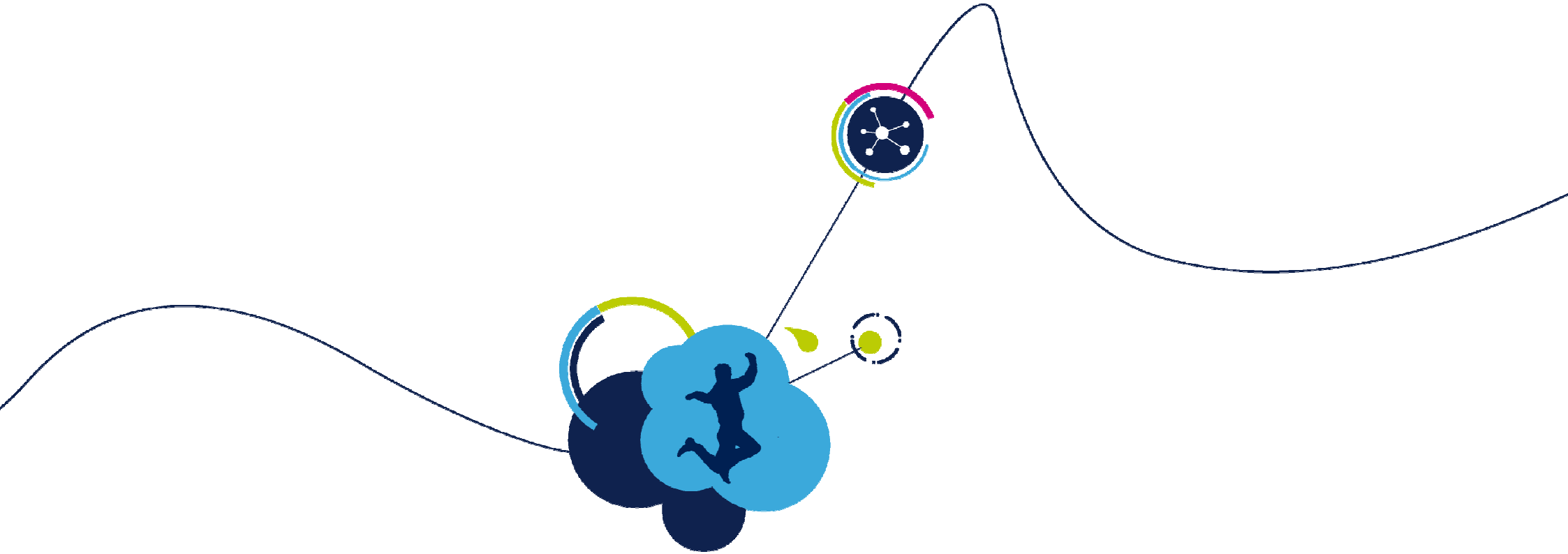
1. Click the GREEN ARROW to Build the Project, **Download** and start the **debugger**. (Ctrl + D)



2. Click the triple-arrow **GO** button! (F5)



The board LED is now blinking!



USB Overview

life.augmented

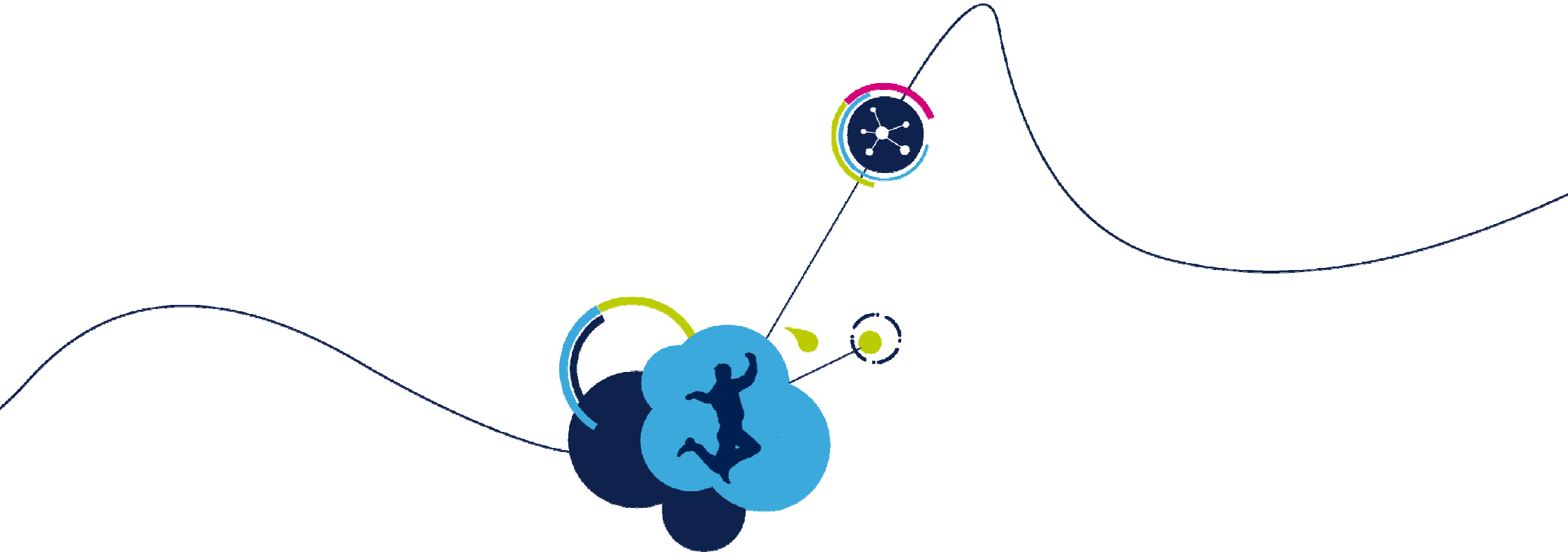
USB Overview

- The Universal Serial Bus is a plug and play serial interface designed to easily connect peripherals to a host.
- The physical USB network is implemented as a star network with one host (master) and up to 127 devices (slaves).
- USB uses two wires for power and two wires for data. It supports multiple data transfer speeds:
  - Low-Speed (1.5Mbits/s)
  - Full-Speed (12Mbits/s)
  - Hi-Speed (480Mbits/s)
  - SuperSpeed(up to 10Gbits/s).
- The functionality of USB devices is defined by class codes, communicated by the slave to the USB host to load a suitable driver.

life.augmented

USB CDC

- The **C**ommunication **D**evice **C**lass supports a wide range of devices that can perform telecommunication and networking functions.
- It is mostly used as a Virtual COM Port and for USB to Wi-Fi or USB to Ethernet dongles.
- The USB Virtual COM Port is an upgrade from the old RS232 serial port. It is a software interface that enables applications to access a USB device as if it were a built-in serial port.



Optional Lab : USB Device Using a Virtual COM Port

life.augmented

Goal

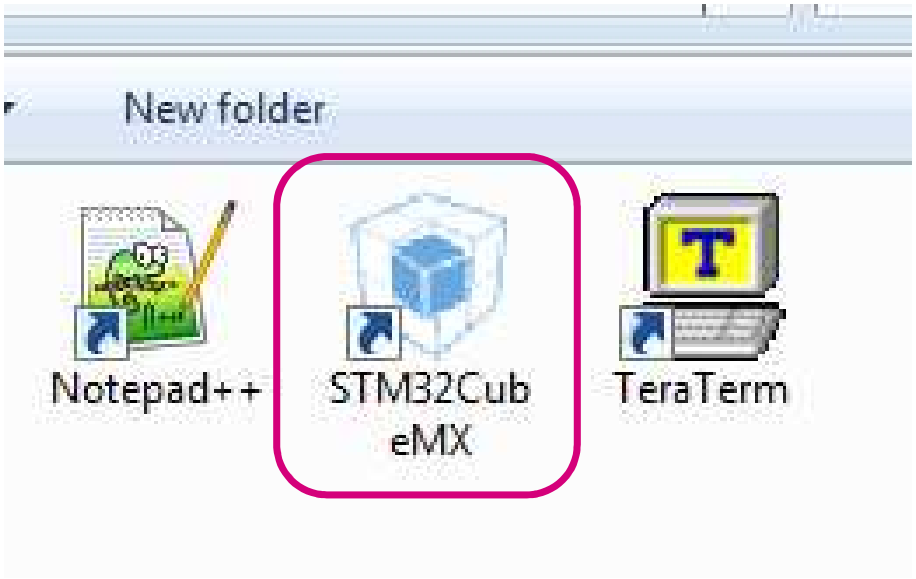
- In this example we are going to demonstrate how to:
  - Use STM32CubeMX to configure USB
  - Setup the STM32L475 USB_OTG_FS block as a CDC Virtual COM Port device.
  - Use serial terminal software to exchange data between STM32 and a PC over USB.

- Close the previous IAR and STM32CubeMX project.

Create New STM32CubeMX Project

- Open the **STM32L4 DK IoT Node Seminar** folder on your Desktop.

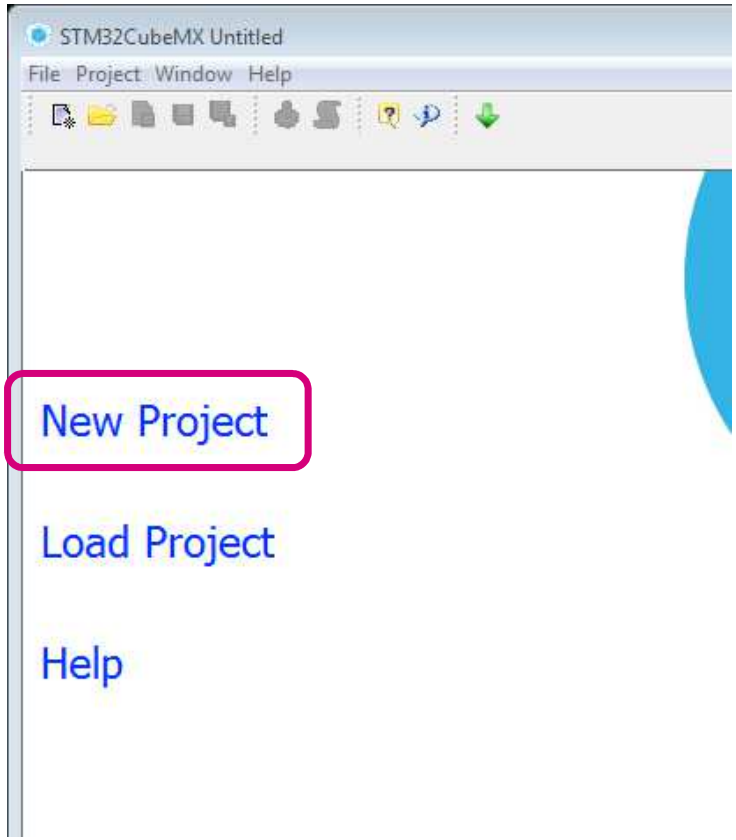
Create New STM32CubeMX Project



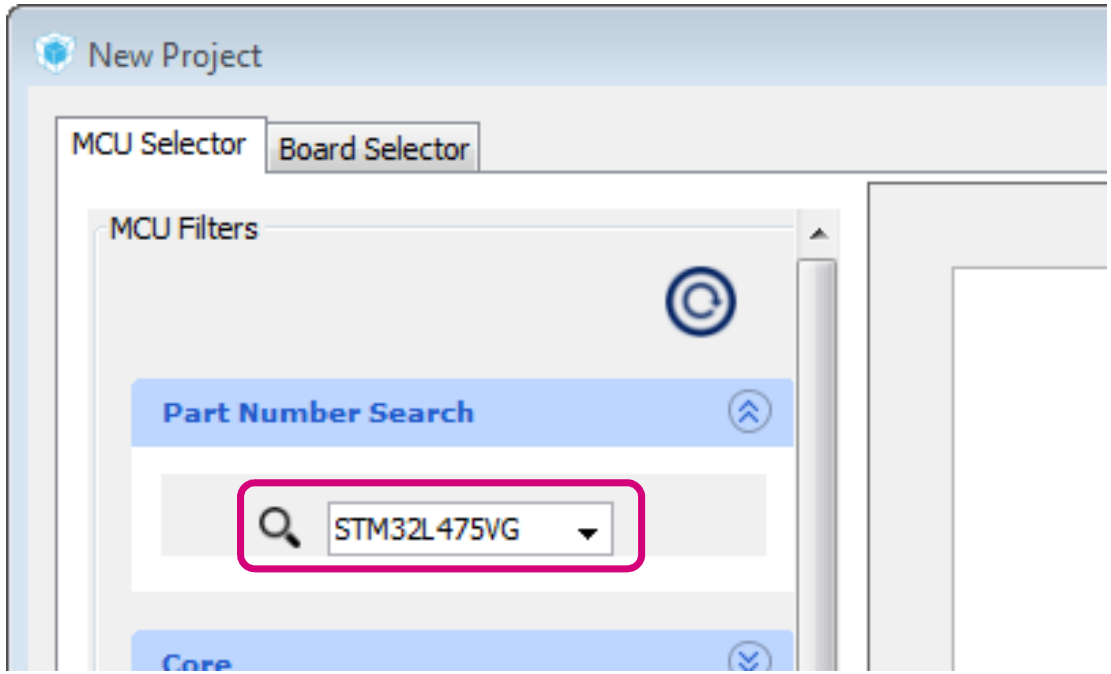
- Double-click on the **STM32CubeMX** shortcut

Create New STM32CubeMX Project

- Click on **New Project**

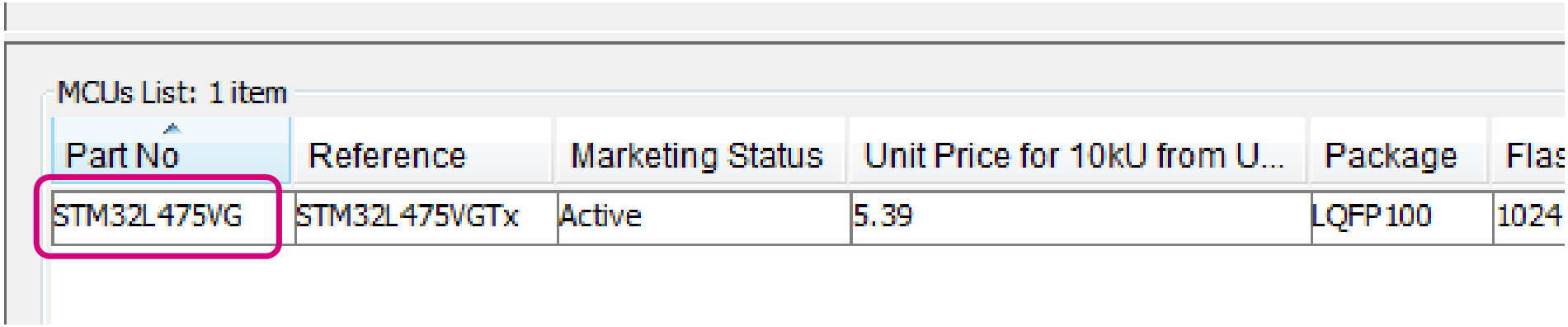


Select the Microcontroller



- Click in the **Part Number Search** box and type **STM32L475VG**

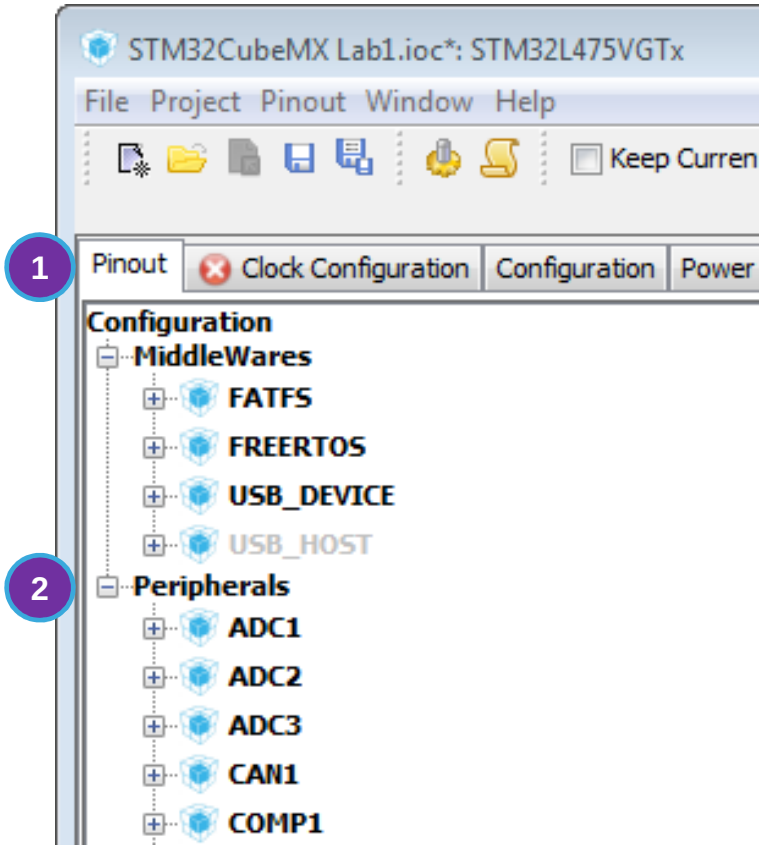
Select the Microcontroller



- Double-click on **STM32L475VG**

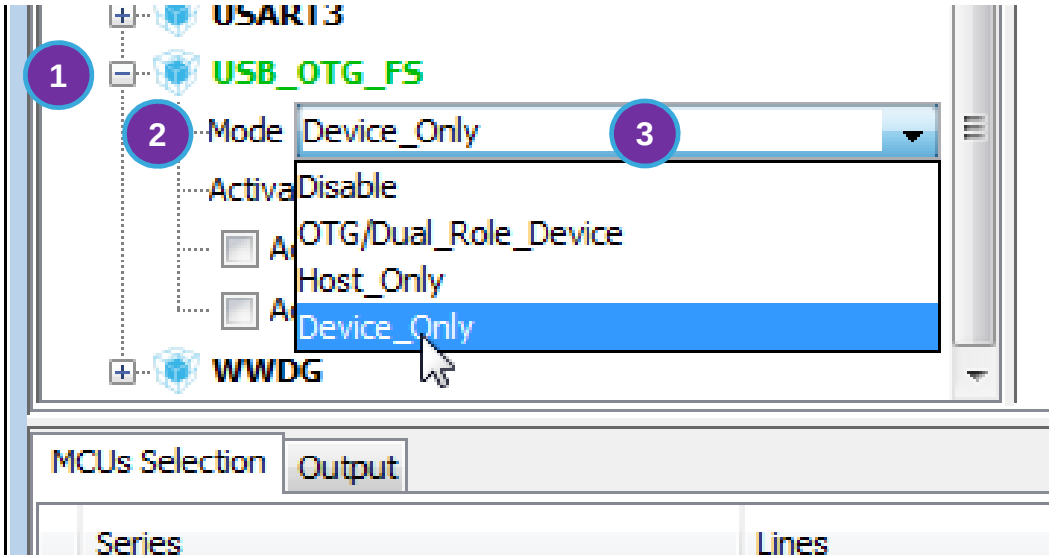
Enable USB Device Mode

1. Select the **Pinout** tab
2. Expand **Peripherals**



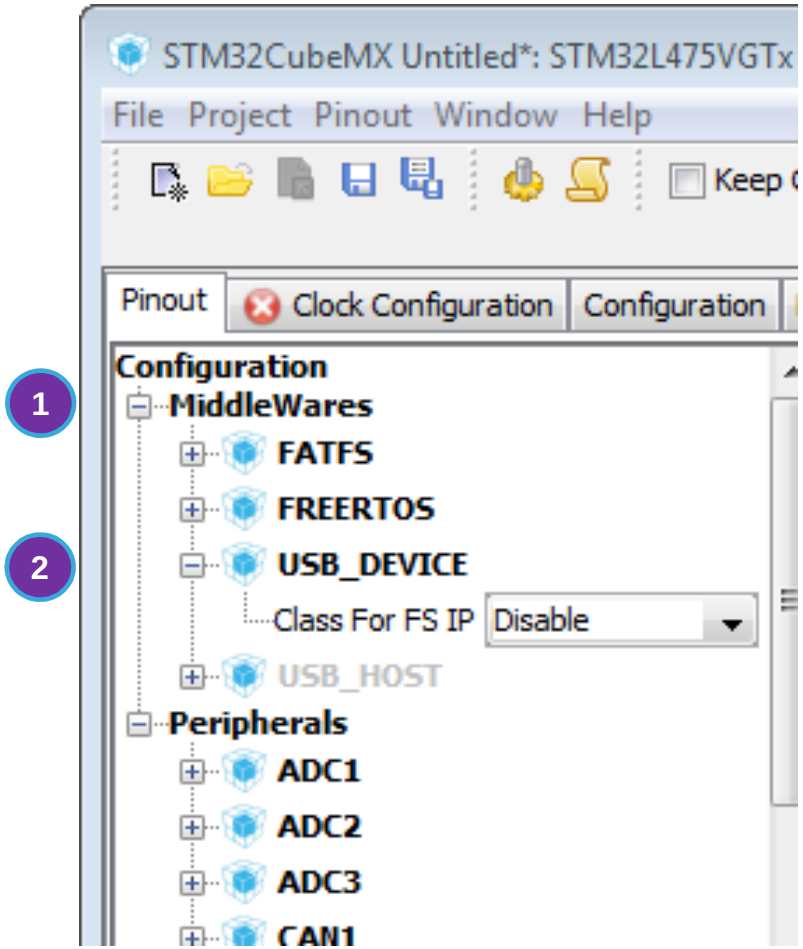
Enable USB Device Mode

1. Scroll down and select **USB_OTG_FS**
2. Select **Mode**
3. Select **Device_Only**

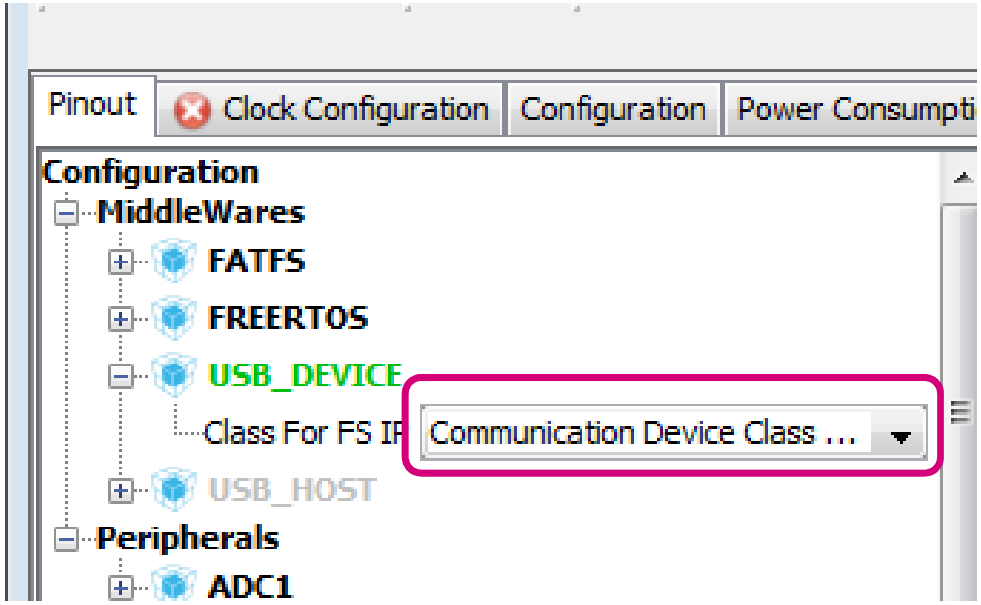


Enable USB CDC

1. Scroll up and expand **MiddleWares**
2. Expand **USB_DEVICE**



Enable USB CDC



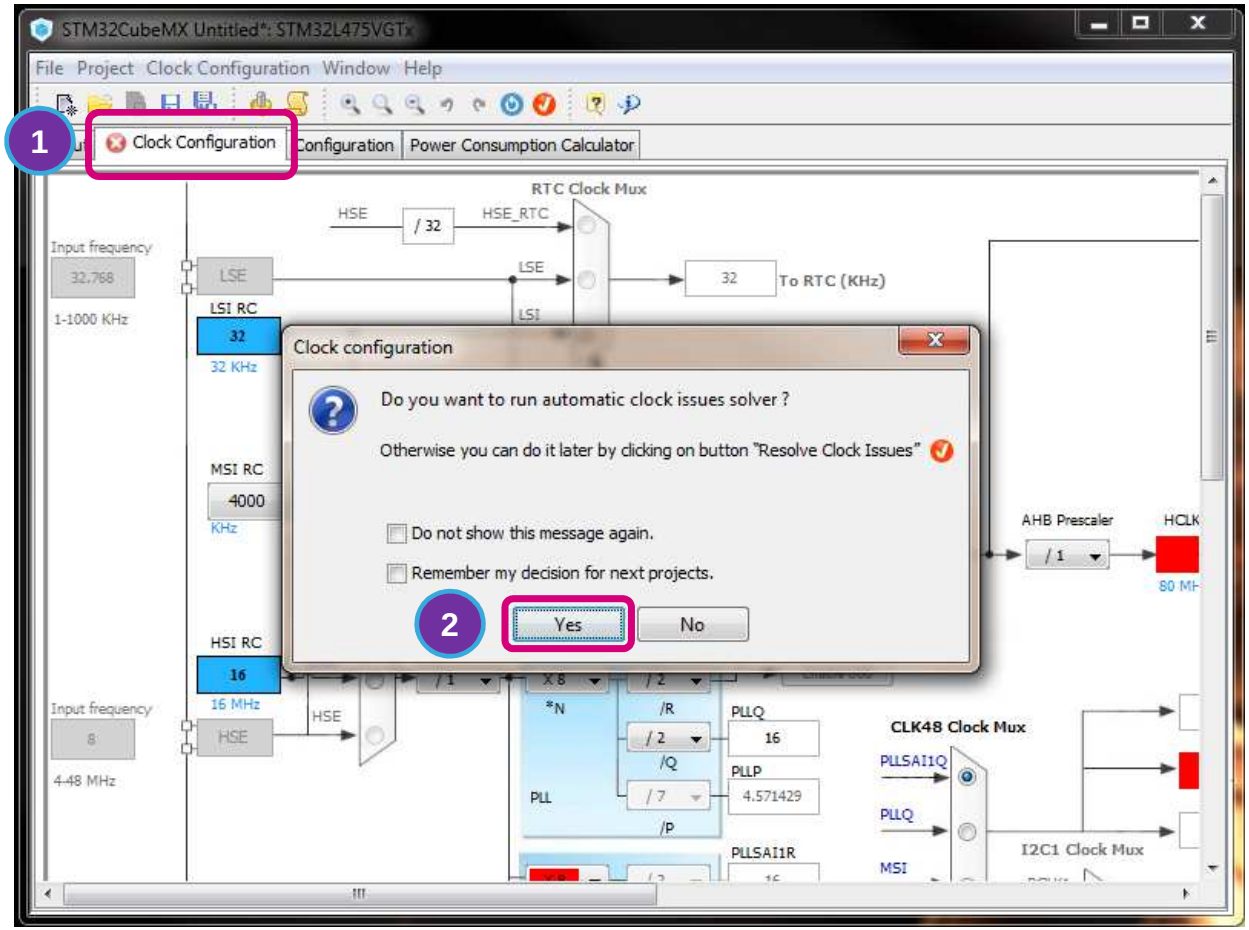
- Select **`Communication Device Class`**

life.augmented

Clock Configuration

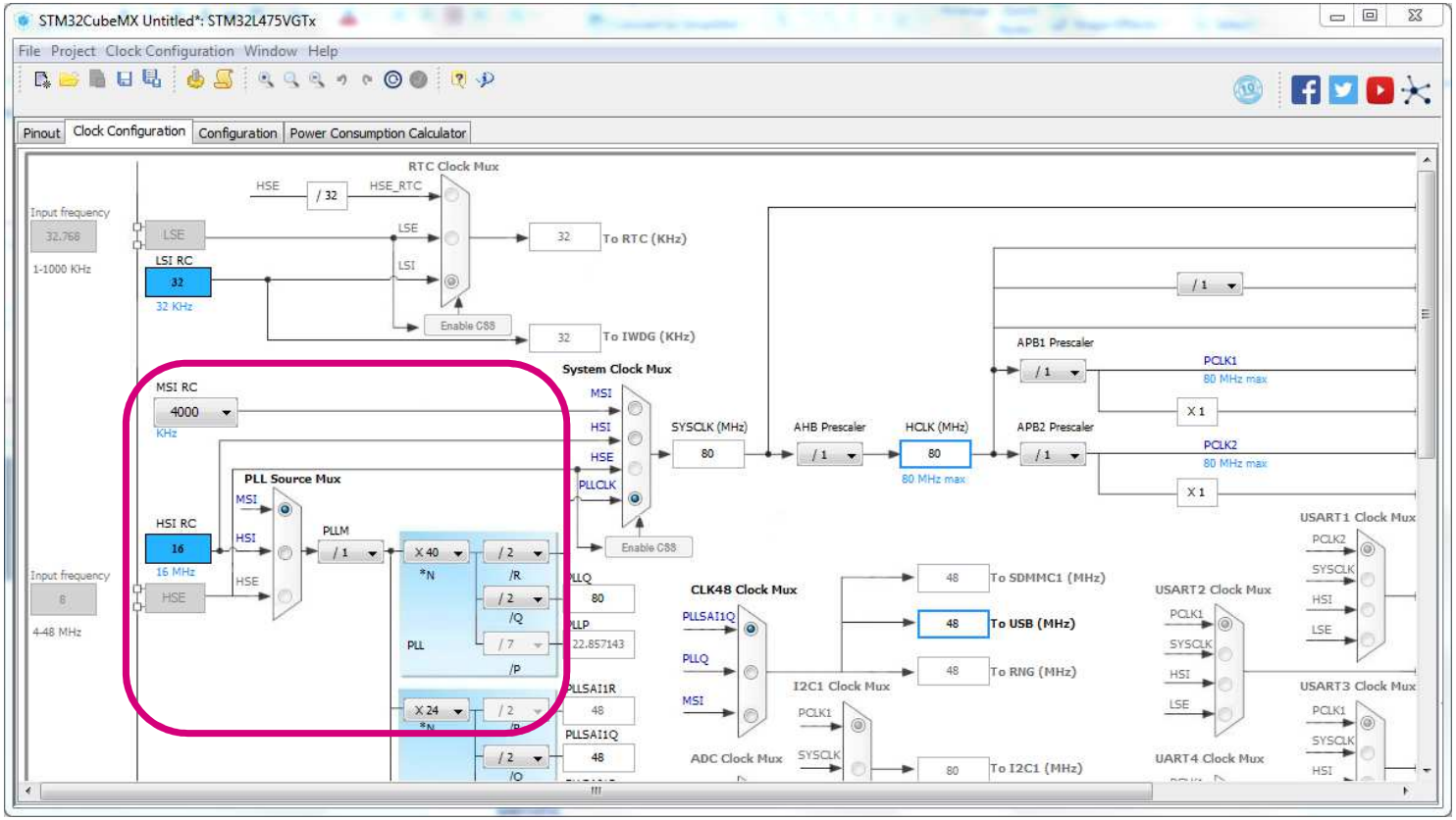
1. Select the **Clock Configuration** tab

2. Click **Yes** on the Clock Configuration Message.



life.augmented

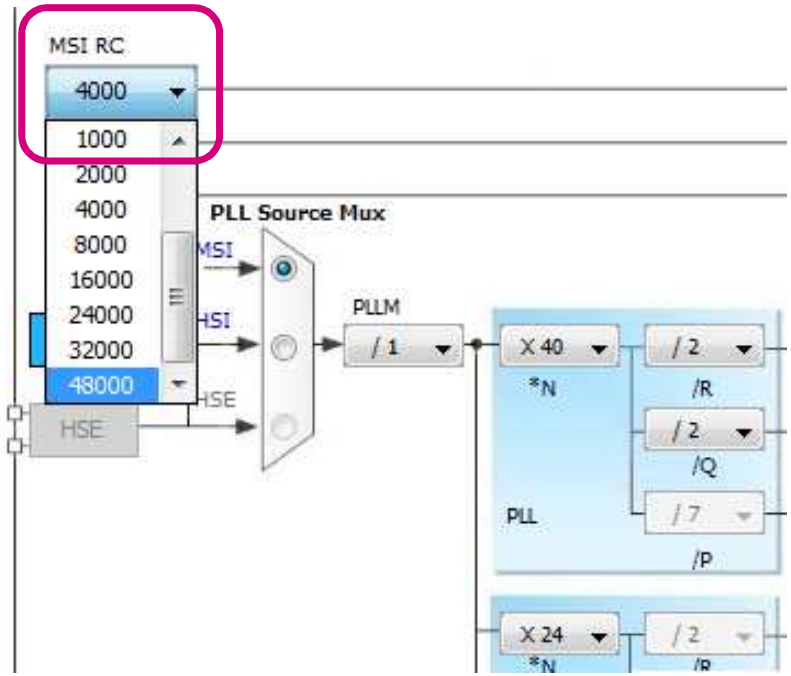
Clock Configuration



life.augmented

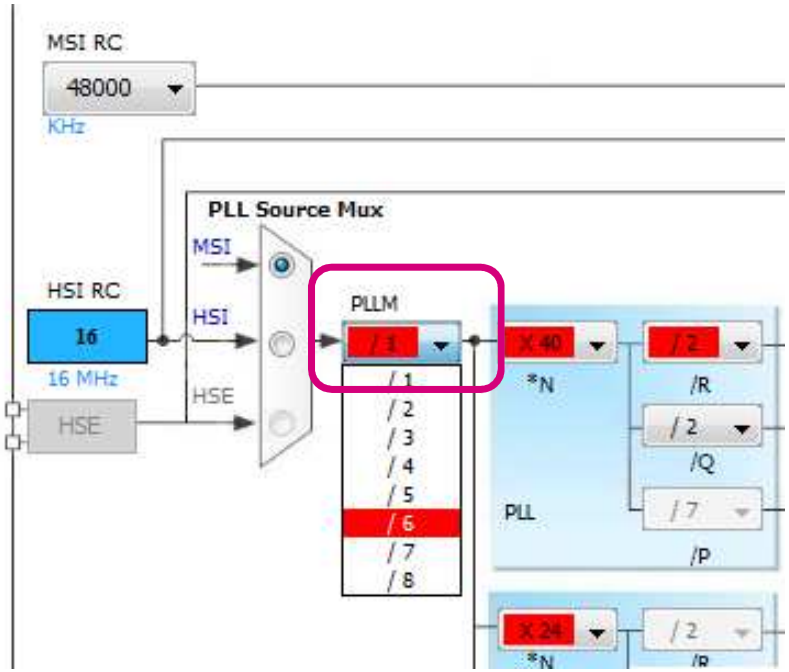
Clock Configuration

- Select **48000** in the **MSI RC** dropdown menu
- Several other boxes will turn **RED** to indicate conflicts.



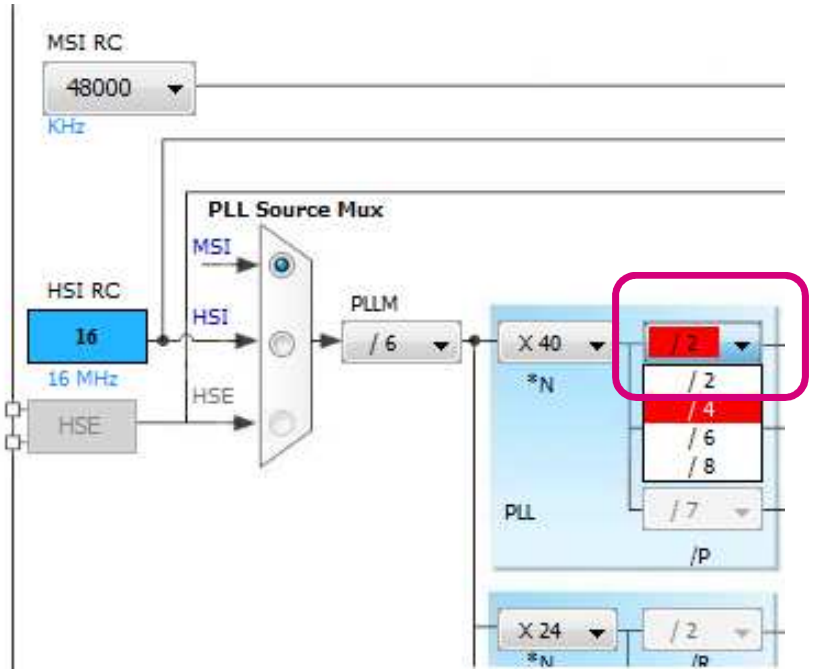
Clock Configuration

- Select **/6** in the **PLLM** dropdown menu



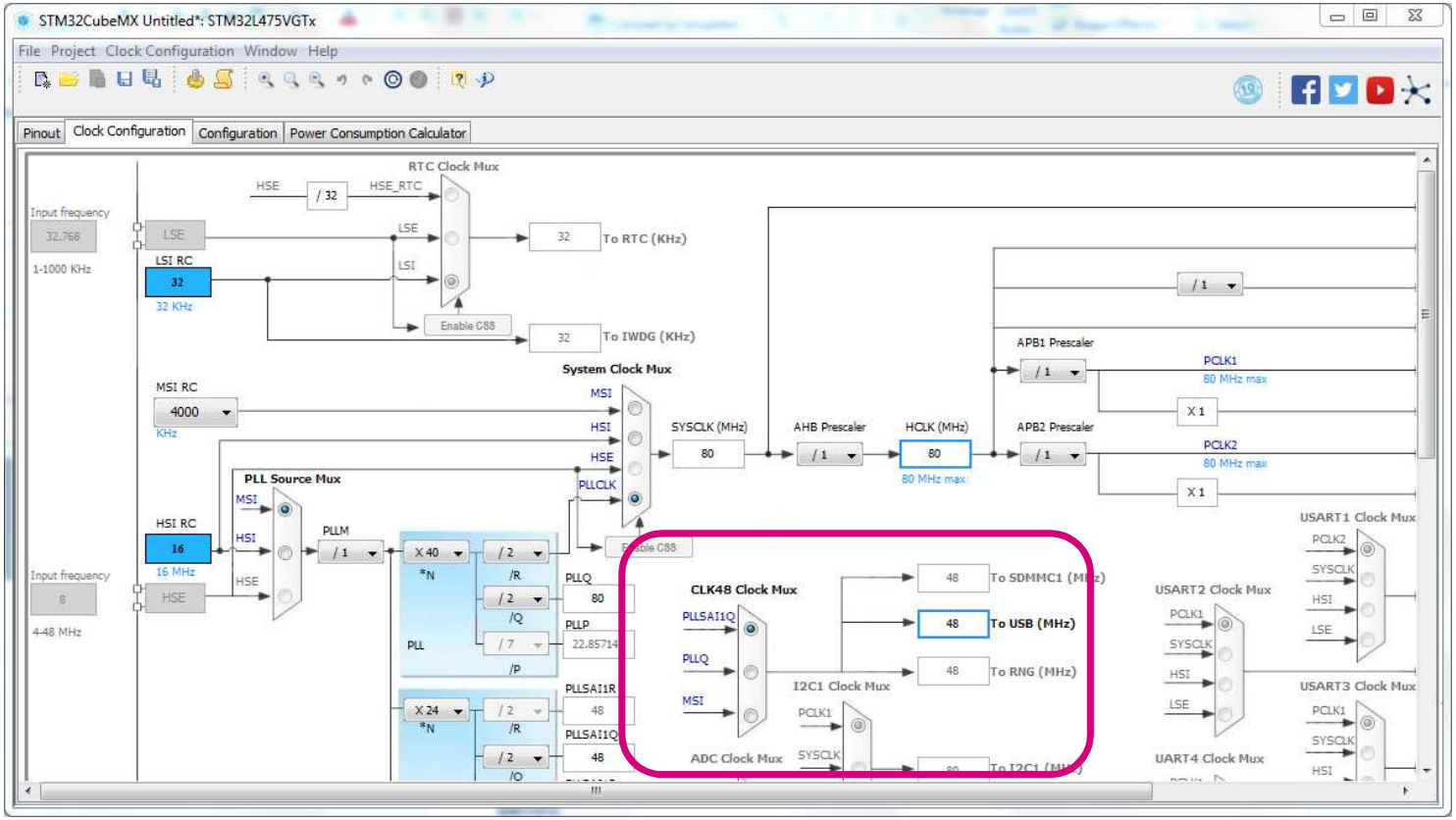
Clock Configuration

- Select */4* in the */R* dropdown menu



life.augmented

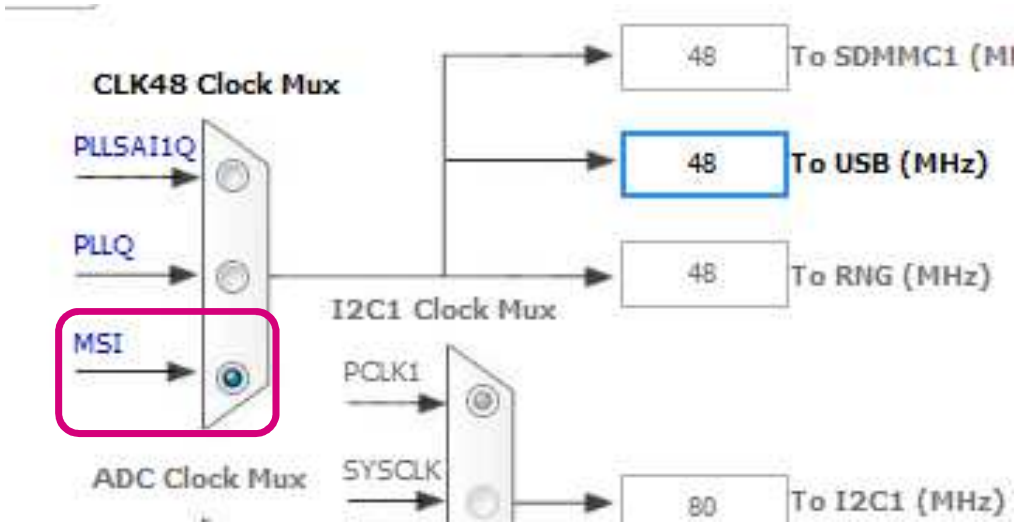
Clock Configuration



life.augmented

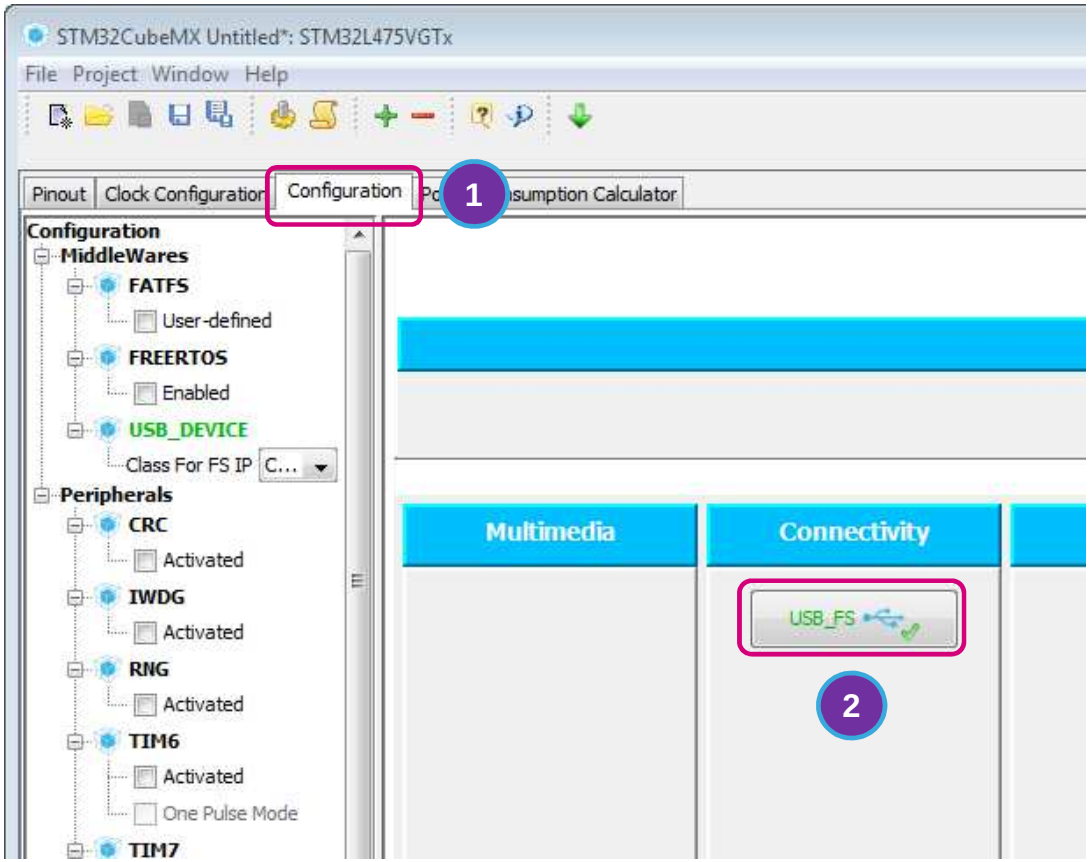
Clock Configuration

- Select **MSI** as the input for the **CLK48 Clock Mux**



USB Configuration

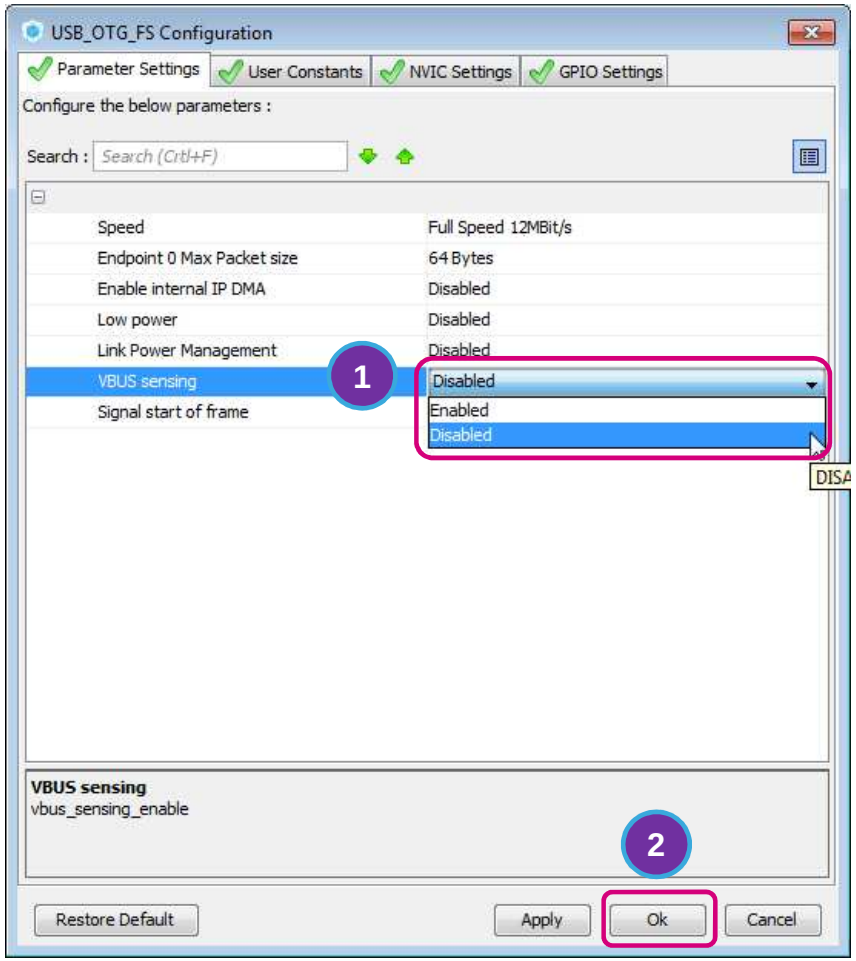
1. Click on the **Configuration** tab
2. Then click on **USB_FS** button



life.augmented

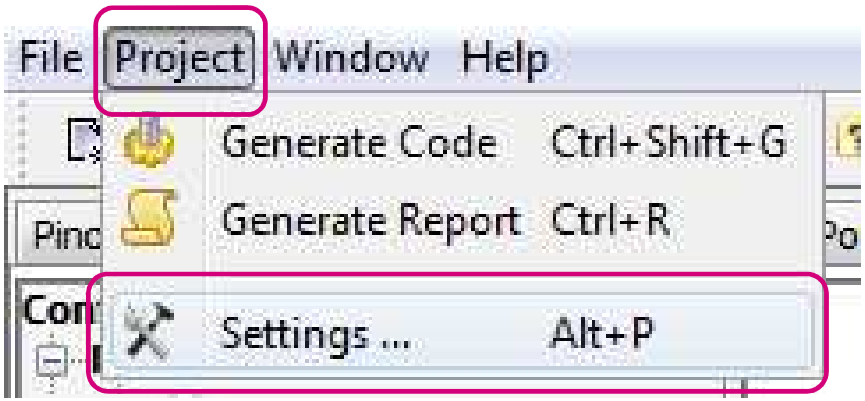
USB Configuration

1. Set the VBUS sensing to **Disabled**
2. Click OK.



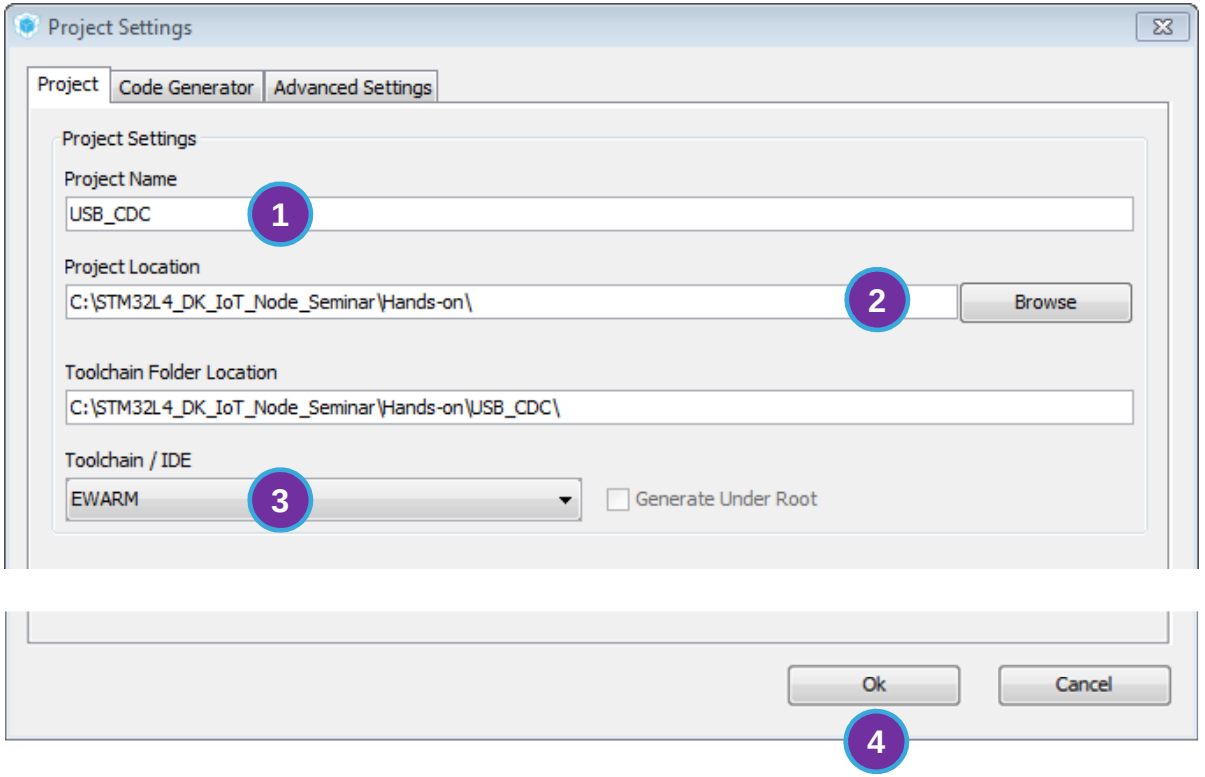
Project Settings

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

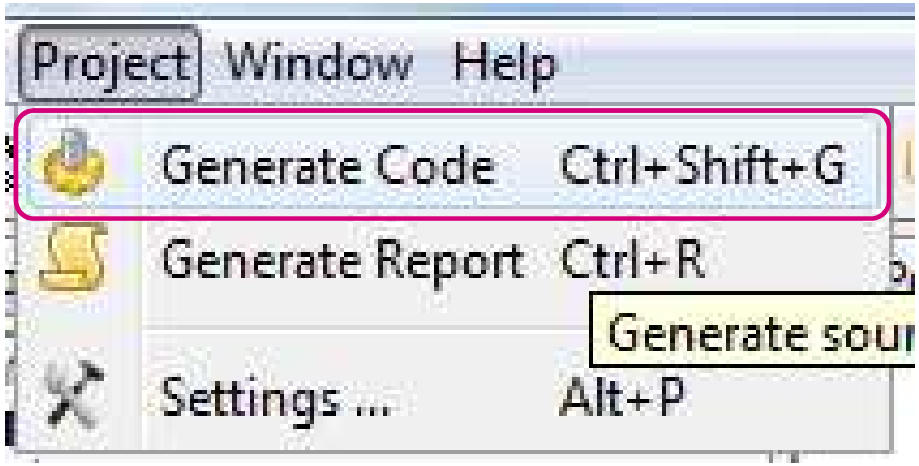


Project Settings

1. Set the project name to **USB_CDC**
2. Set the project location: **C:\STM32L4_DK_IoT_Node_Seminar\Hands-on**
3. Set the IDE Toolchain to **EWARM**.
4. Click **OK**

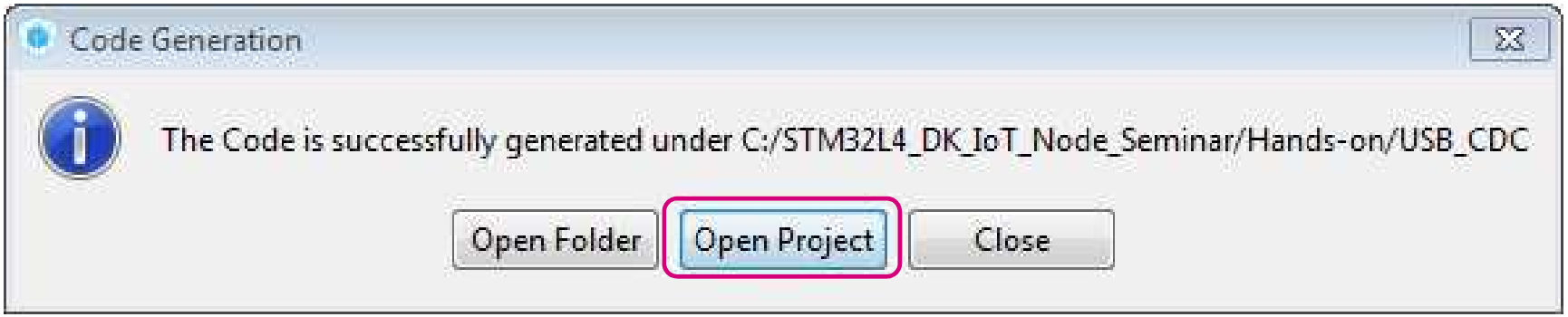


Generate the Project



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

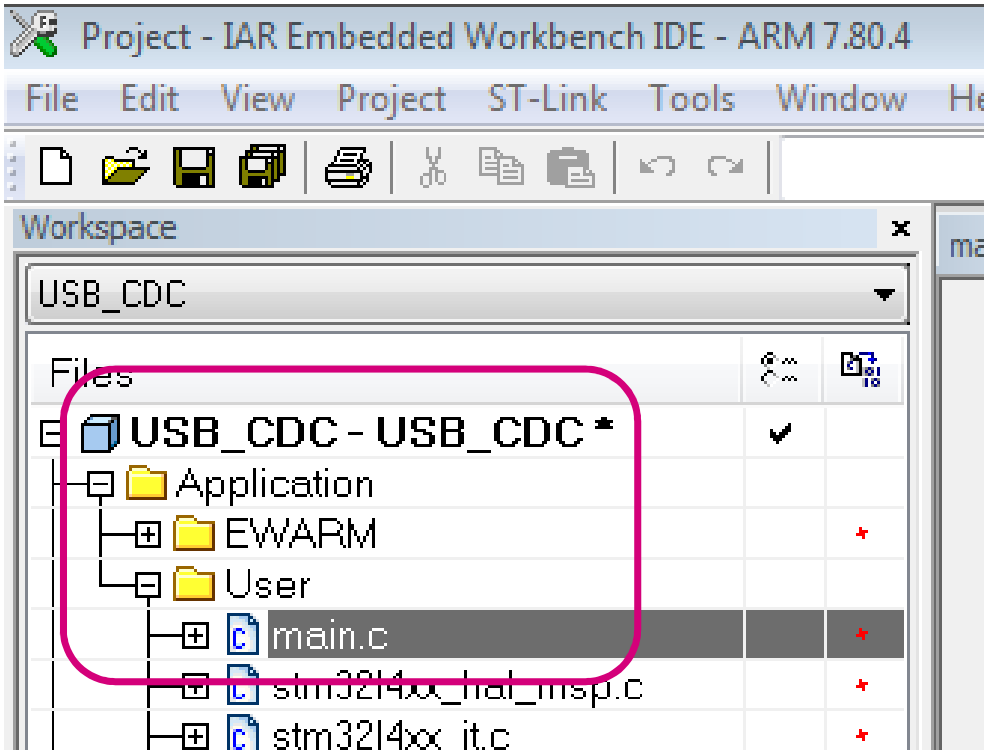
Open the Project



- Click **Open Project**

Edit main.c

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



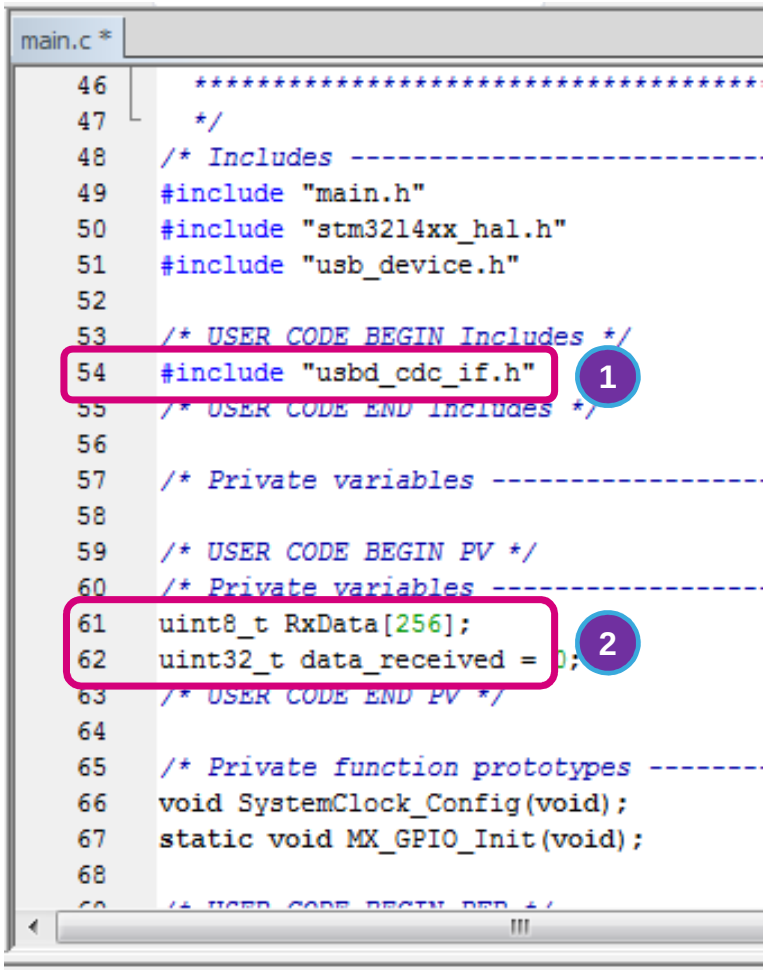
Edit main.c

1. Add the USB device CDC header file on line 54:

```
#include “usbd_cdc_if.h”
```

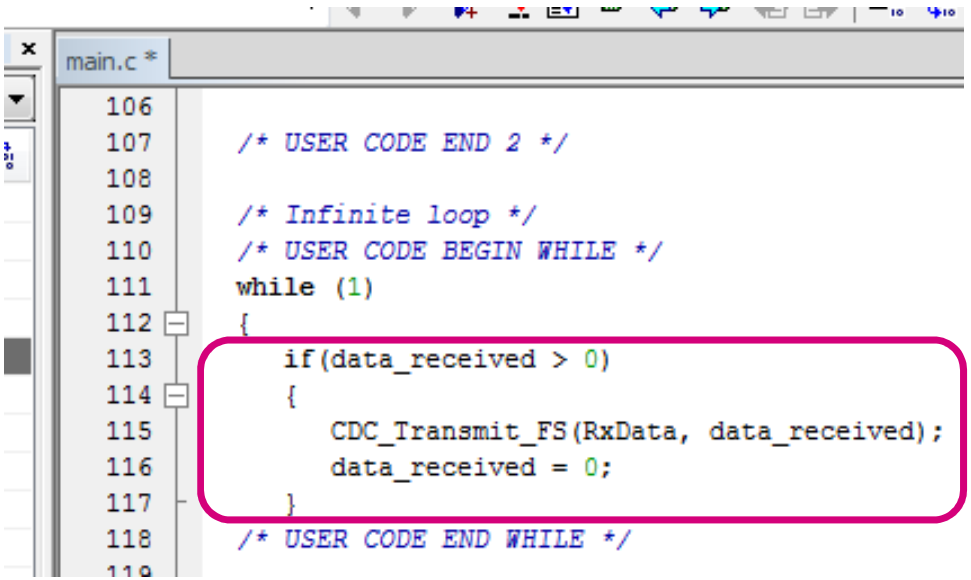
2. Add the following variables on lines 61 & 62:

```
uint8_t RxData[256];
uint32_t data_received = 0;
```



life.augmented

Edit main.c



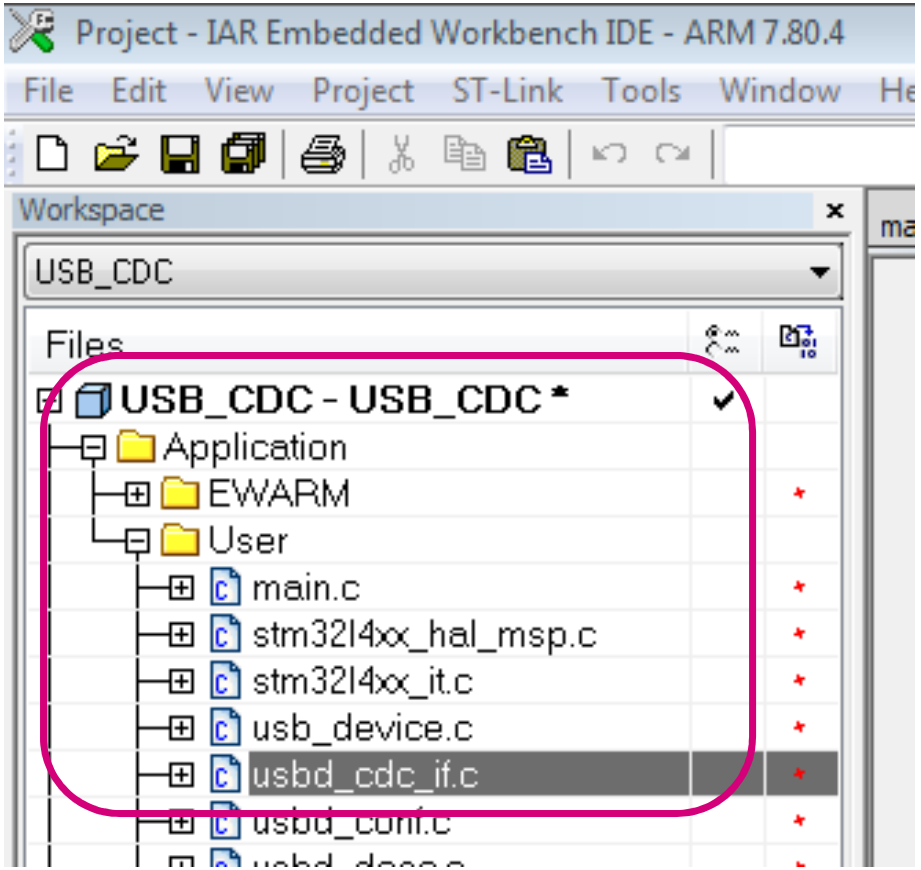
- Add the following code inside the while(1) loop, lines 113-117.

```
if(data_received > 0)
 {
    CDC_Transmit_FS(RxData, data_received);
    data_received = 0;
 }
```

life.augmented

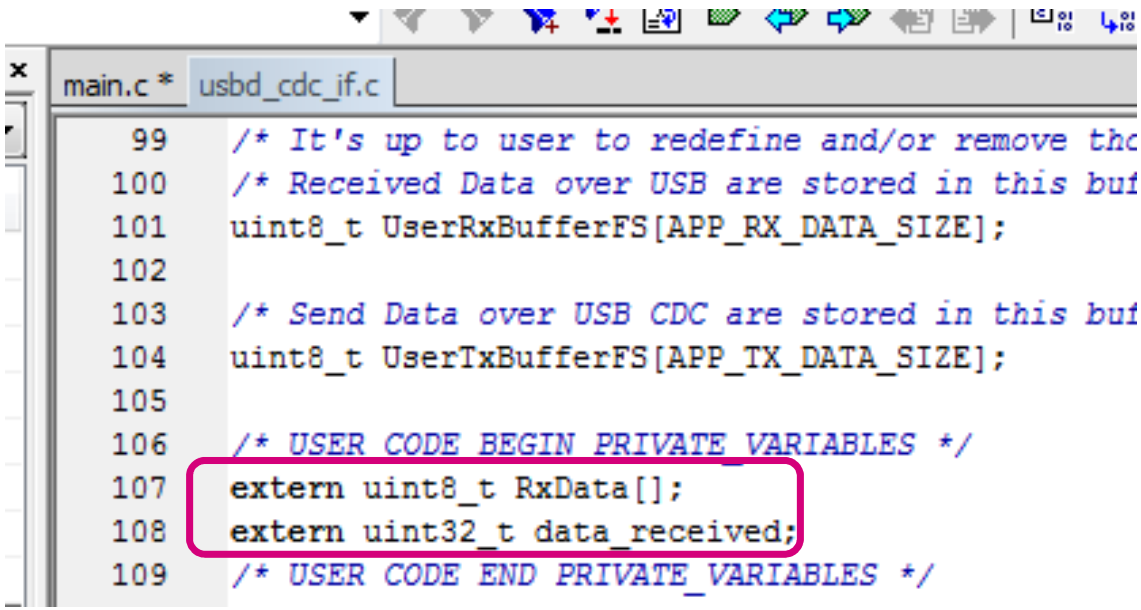
Edit usbd_cdc_if.c

- Expand the file tree:
  1. Expand **USB_CDC**
  2. Expand **Application**
  3. Expand **User**
  4. Double-click **usbd_cdc_if.c**



life.augmented

Edit usbd_cdc_if.c



- Add the following variables to lines **107** and **108**:

```
extern uint8_t RxData[];
extern uint32_t data_received;
```

Edit usbd_cdc_if.c

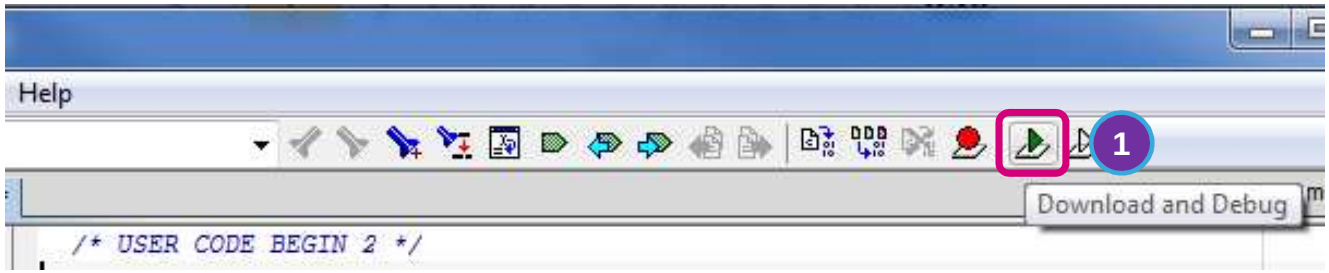

```
268  static int8_t CDC_Receive_FS (uint8_t* Buf, uint32_t *Len)
269  {
270    /* USER CODE BEGIN 6 */
271    USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
272    USBD_CDC_ReceivePacket(&hUsbDeviceFS);
273    data_received = *Len;
274    memcpy(RxData, Buf, data_received);
275
276    return (USBD_OK);
277    /* USER CODE END 6 */
```

- Add the following code to lines **273** & **274:**

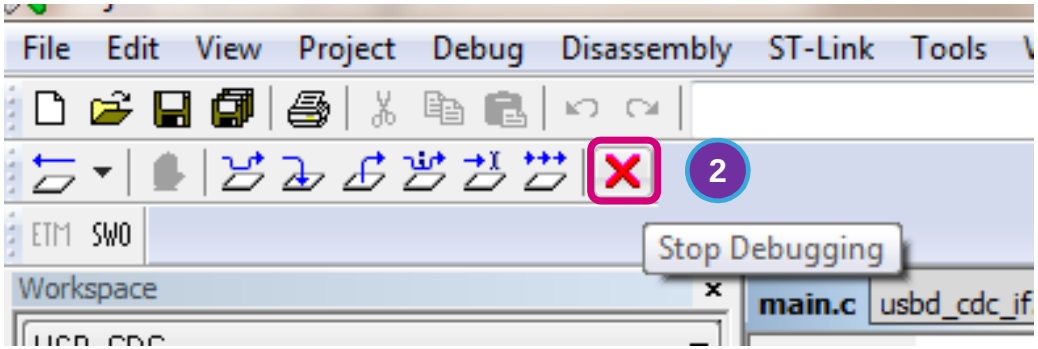
```
data_received = *Len;
memcpy(RxData, Buf, data_received);
```

Build and Load

1. Click the GREEN ARROW to Build the Project, **Download** and start the **Debugger**. (Ctrl + D)



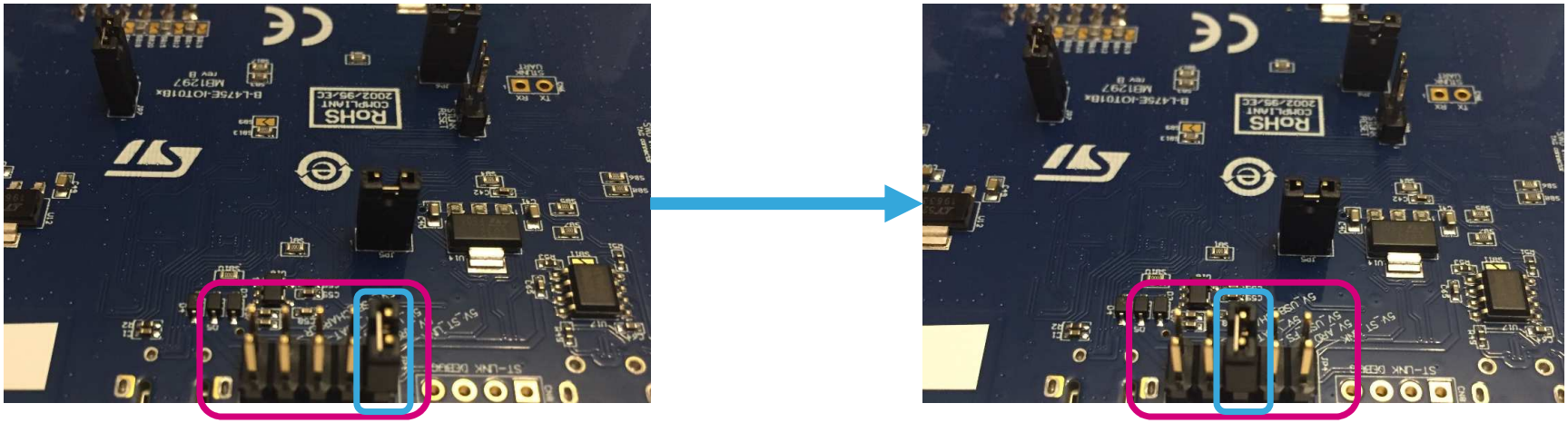
2. Click the **X** button (Ctrt + Shift + D) to **Stop Debugging**.



Move the DK IoT Board Power Jumper

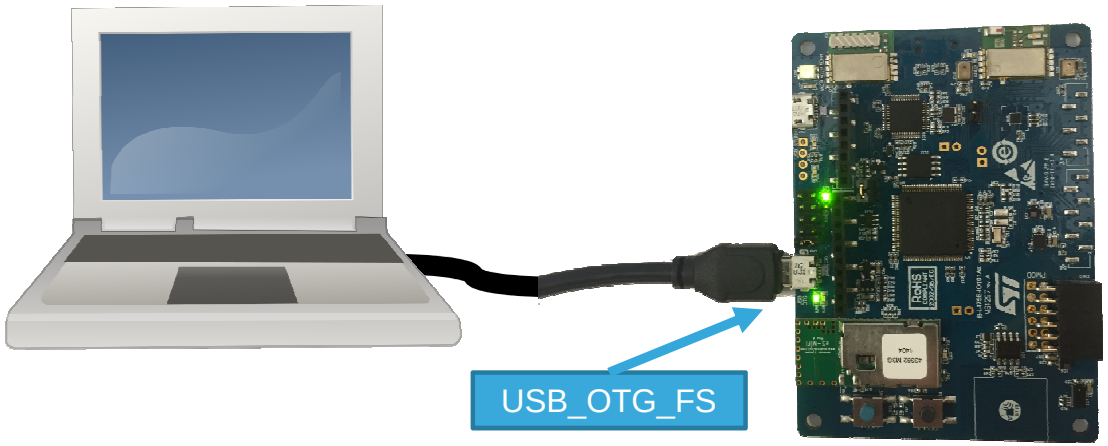
1. Disconnect the USB cable from the on-board ST-Link.

2. Flip your DK IoT board over and move the jumper to the middle position.



Connect the board to the PC

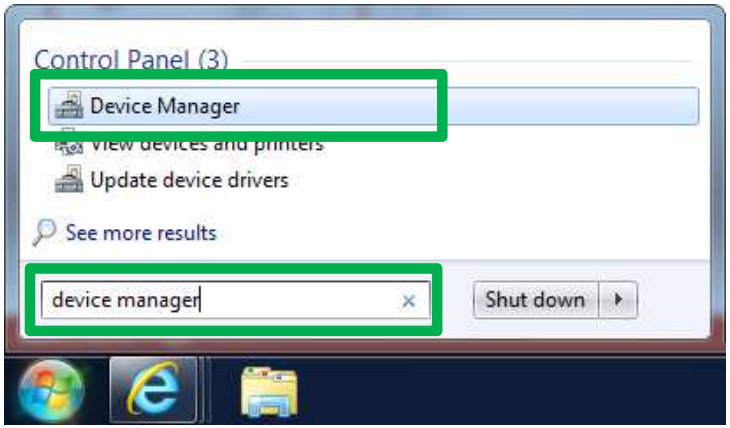
1. Connect the USB cable between the STM32L475 USB_OTG_FS and the PC.
2. Windows should detect the device and automatically install the driver.
3. If the device is NOT detected, you will need to install the driver manually.



life.augmented

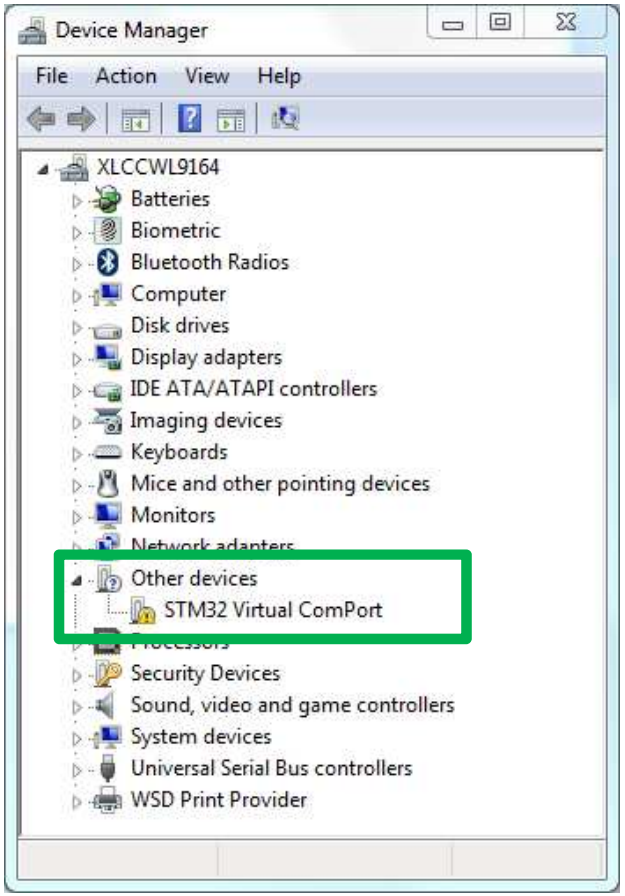
Manual Driver Installation

- Open the **START Menu** and type '**device manager**'
- Select "**Device Manager**" in the search results that appear



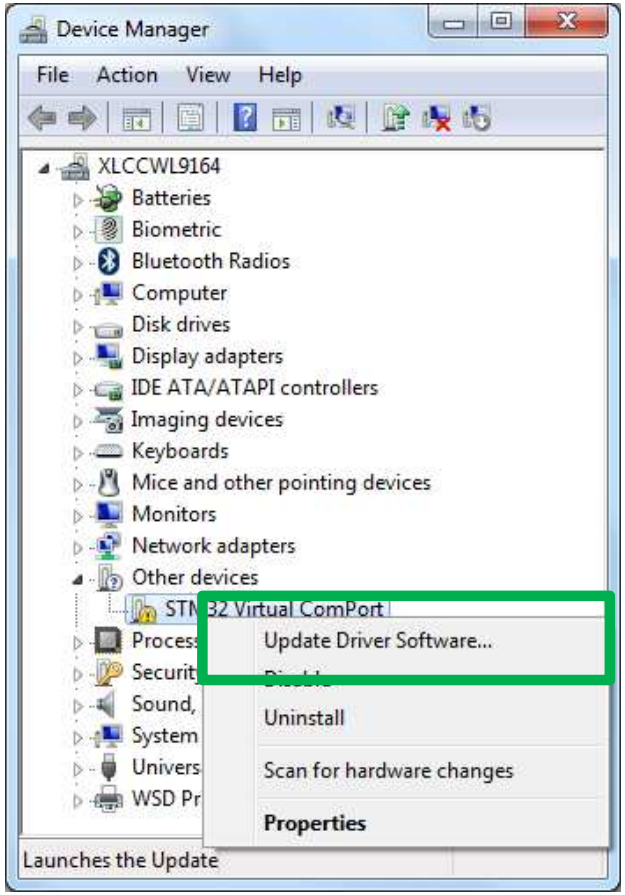
Manual Driver Installation

- Right-click on the **STM32 Virtual ComPort** device under **Other devices**



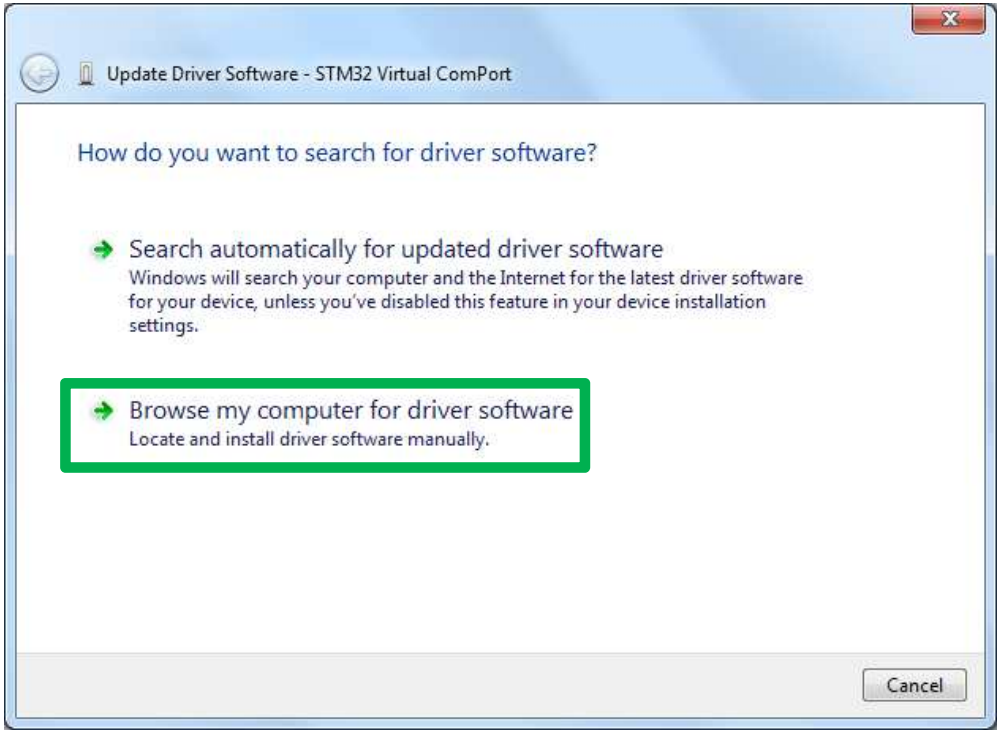
Manual Driver Installation

- Click on **Update Driver Software**



life.augmented

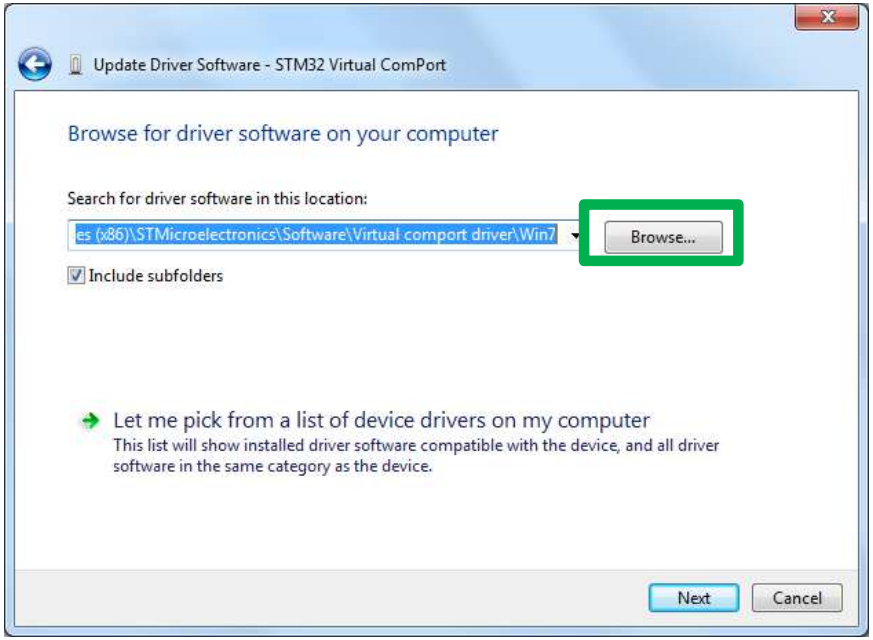
Manual Driver Installation



- Click on **Browse my computer for driver software**

life.augmented

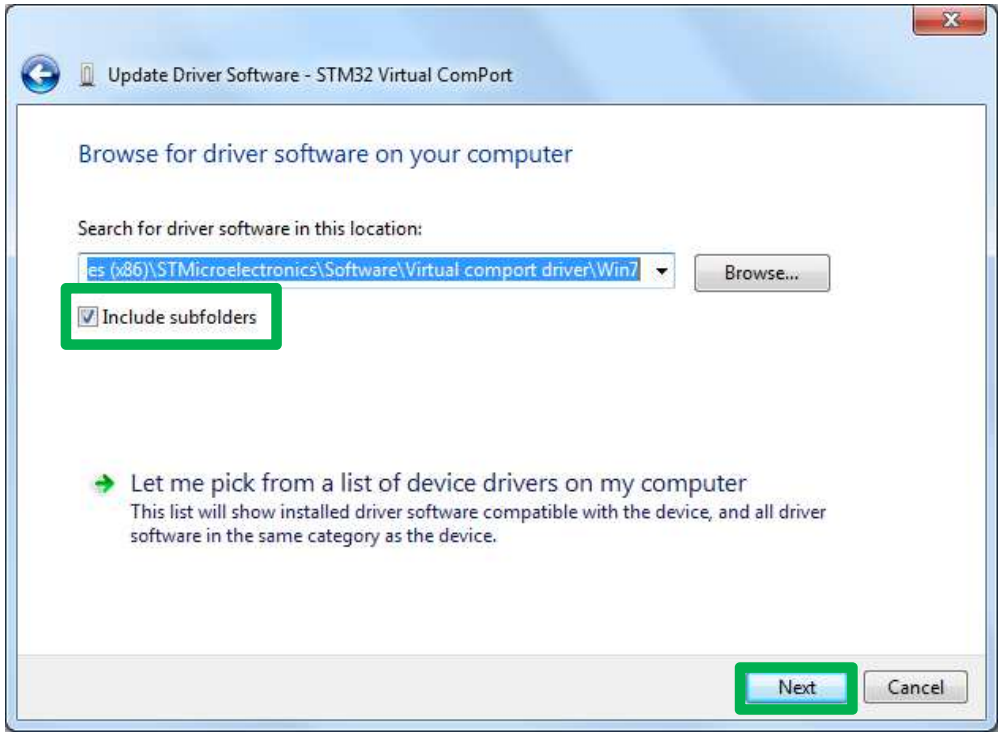
Manual Driver Installation



- Click **Browse**, then navigate to:

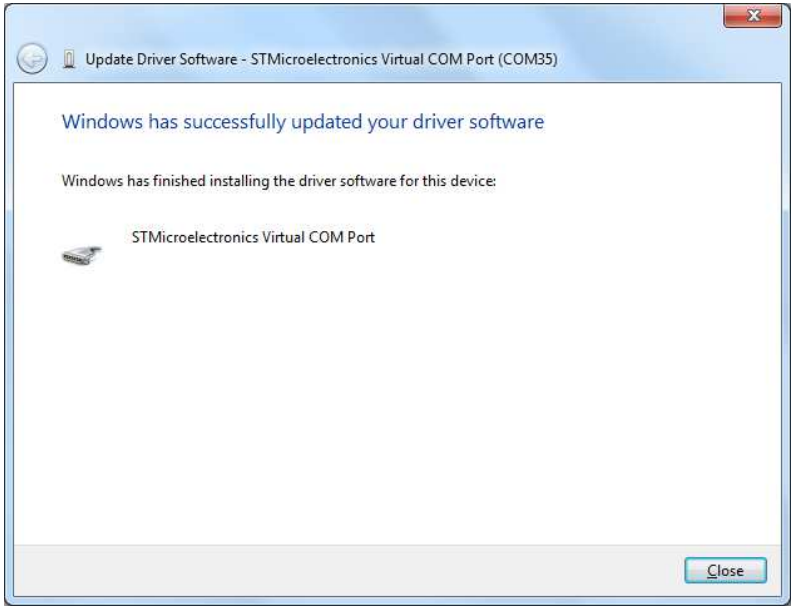
```
C:\Program Files (x86)\STMicroelectronics\Software\Virtual
comport driver
```

Manual Driver Installation



- Select Include subfolders, then click **Next**

Manual Driver Installation



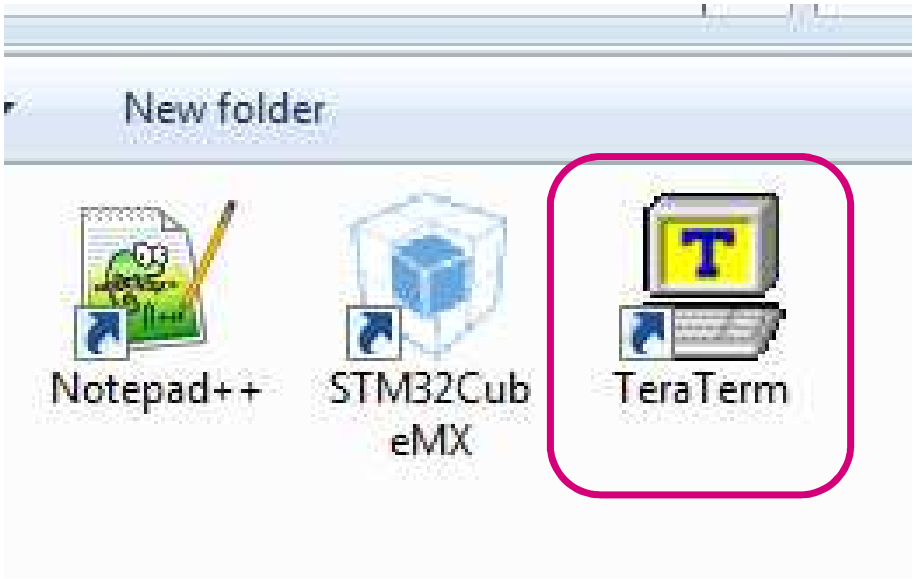
- Windows will install the driver
- Click ‘Close’

life.augmented

Open TeraTerm

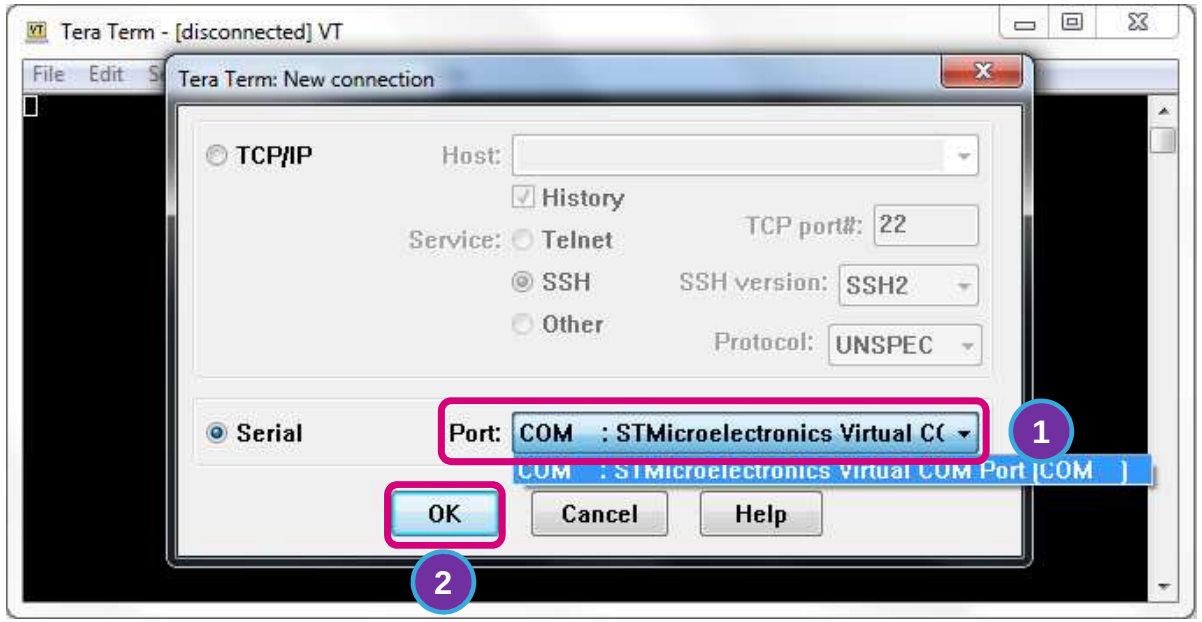
- Open the **STM32L4 DK IoT Node Seminar** folder on your Desktop.

Open TeraTerm



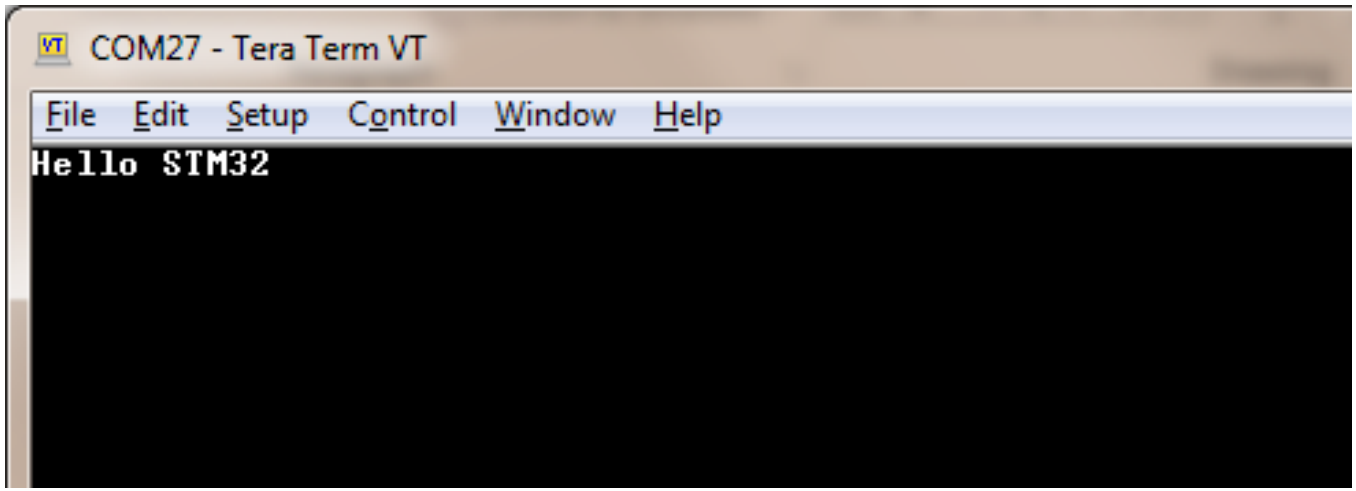
- Double-click on the **TeraTerm** shortcut

Configure TeraTerm



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

TeraTerm Echo Output



- Type “**Hello STM32**”
- The Tera Term window will display the characters that are echoed back

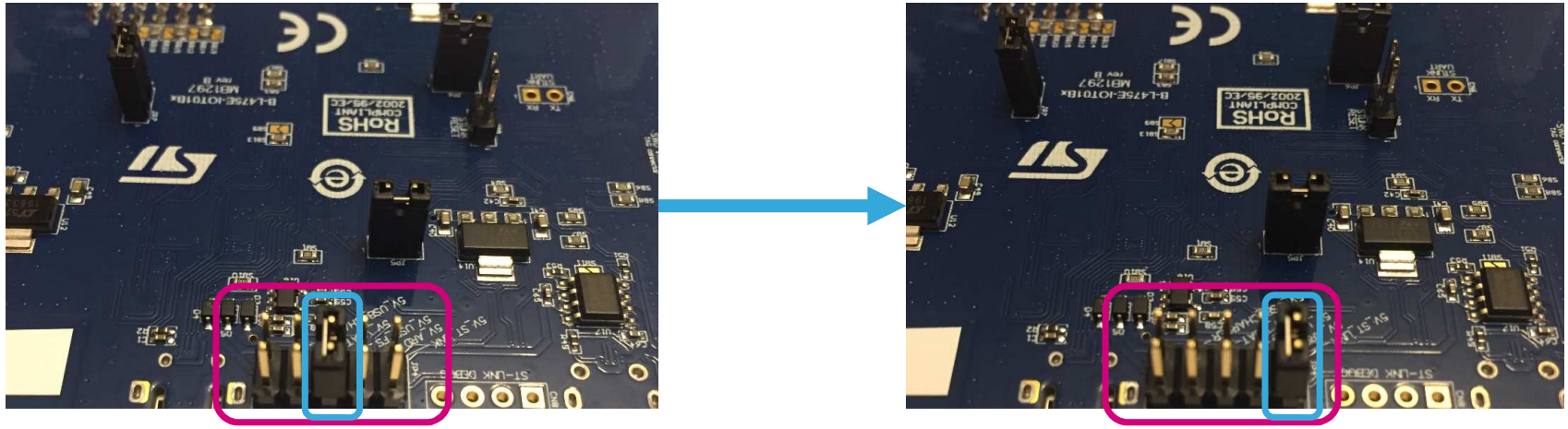
How this example can be used?

- You can use the example to:
  - Configure your application
  - Debug
  - Firmware Update
  - Bluetooth Low Energy / SubGHz to USB bridge

Move the DK IoT Board Power Jumper

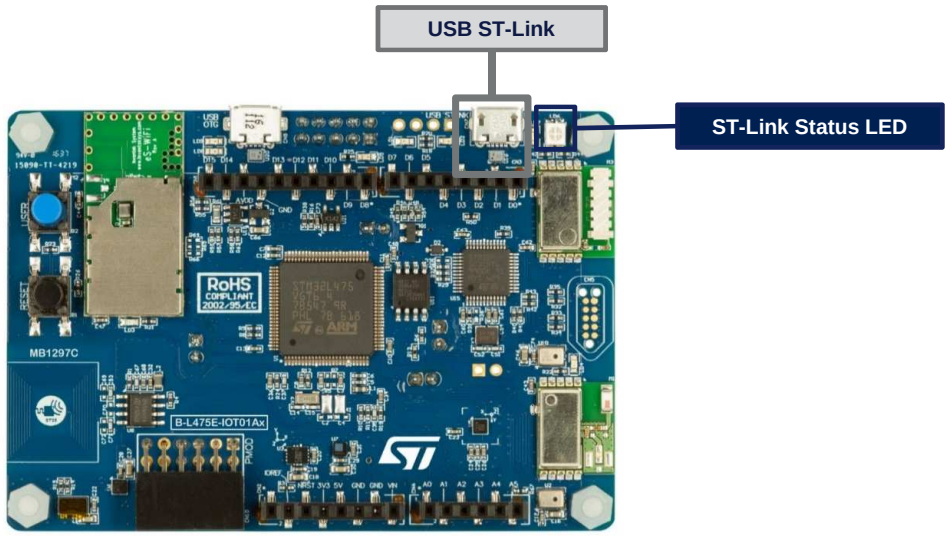
1. Disconnect the USB cable between the STM32L475 USB_OTG_FS and the PC.

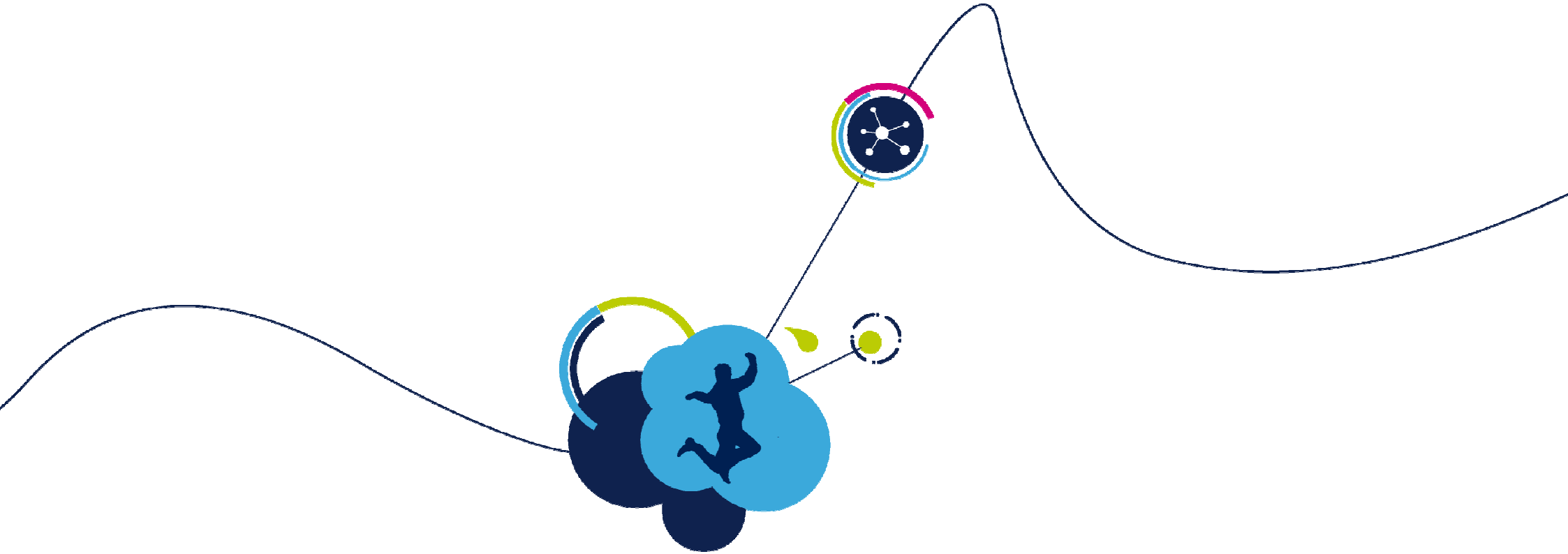
2. Flip your DK IoT board over and move the jumper to the 5V_ST_LINK position.



Connect the board to the PC

- 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.

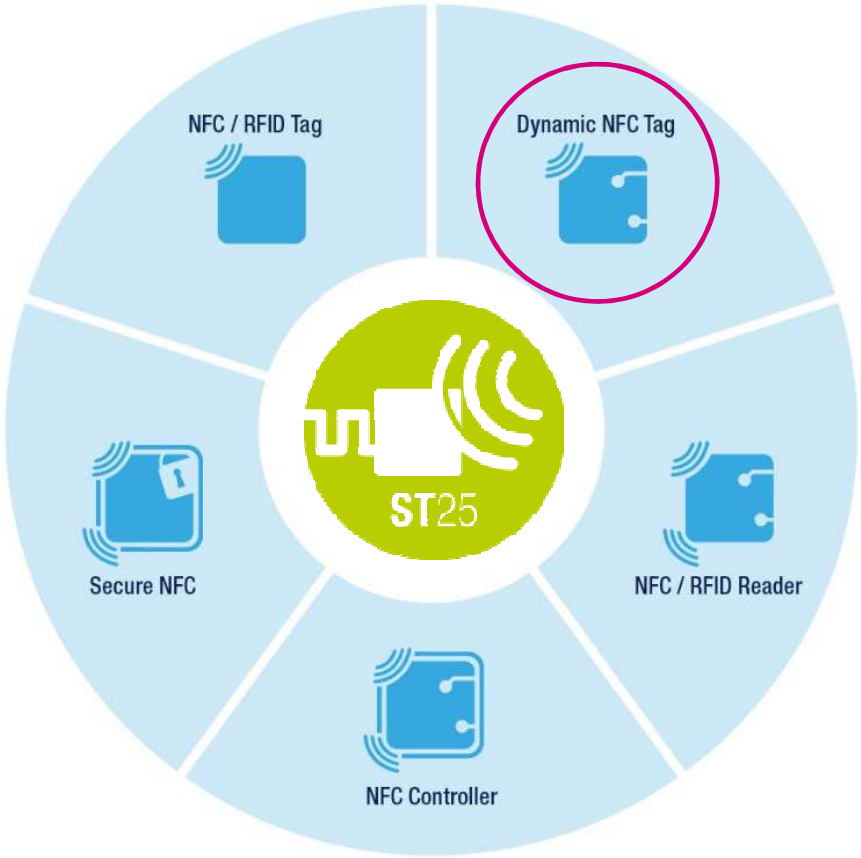




NFC Overview

life.augmented

ST NFC Positioning



Complete Portfolio

→ From Tags to High Performance Readers

Market Coverage

→ Covering all NFC application needs and leveraging a rich ecosystem

Acquisition

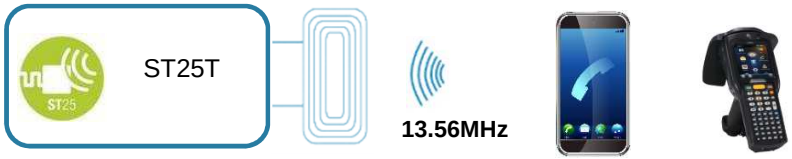
ST acquired ams AG NFC / RFID Readers Assets

life.augmented

ST25 Product Lines

Ticketing, Gaming, Medical, Brand protection, Access control, …

NFC Tags



Industrial, Consumer, Metering, Appliance, IoT…

NFC Dynamic Tags

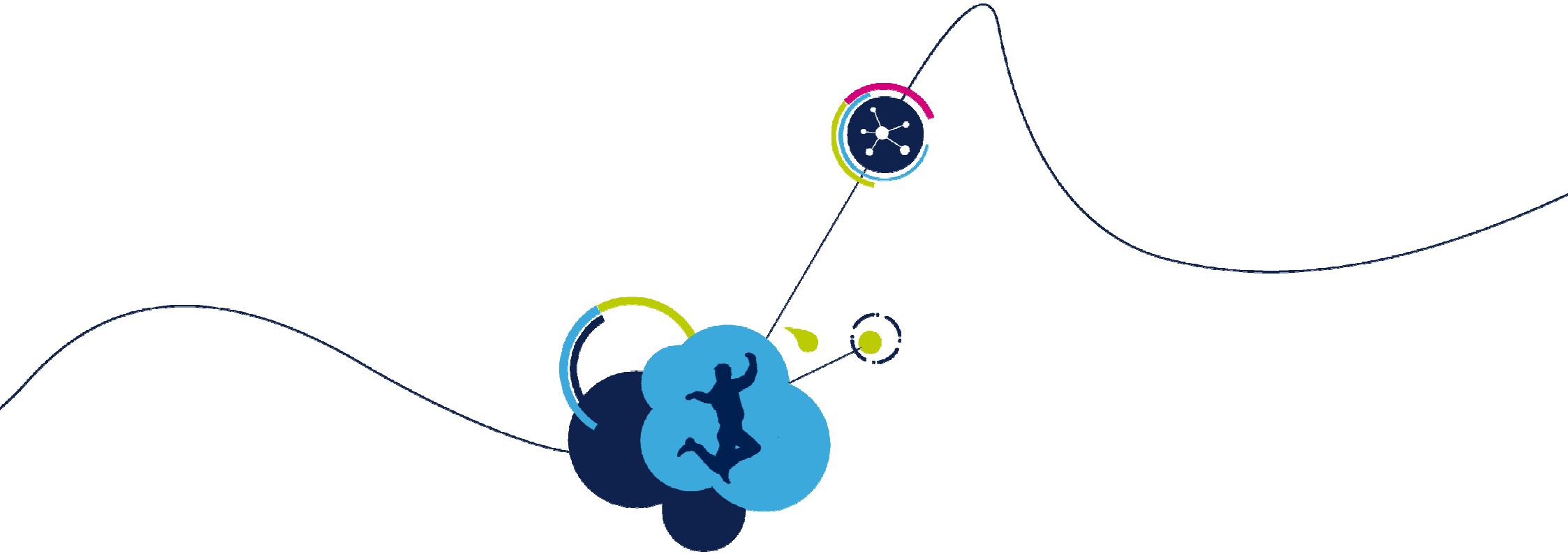
POS & mPOS Terminals

EMVCO Readers

Automotive, Accessory ID, Gaming, Medical, Brand protection, Access control, …

Readers

life.augmented



Sensors Overview

life.augmented

Motion MEMS : 6-axis IMU and Magnetic Sensors

6-axis
LSM6DSL
LSM6DSM

2.5x3x0.86 mm

Key advantages:
- Accuracy
- Power consumption
- Smart Functions

Main features

- Stability and ultra low noise to address demanding applications
- Best in class Low power consumption for extend battery life. I_{COMBO} = 0.65mA in H. Perf. mode
- Advanced digital features for Sensor HUB, pedometer and significant motion detection functionalities.
- 4kB Embedded FIFO for smart data storage and power saving.
- LSM6DSM: Dedicated auxiliary SPI for OIS applications and Android M compliant

Magnetometer
LIS3MDL

2x2x1 mm

Key advantages:
- High Resolution
- Low Power

Main features

- Ultra-compact high-performance 3D magnetometer
- Wide magnetic sensor dynamic range
- Low magnetic offset
- Embedded Self Test and Temperature compensated

life.augmented

Environmental MEMS : Pressure & Humidity Sensor

Pressure
LPS22HB

2x2x0.76 mm

Key advantages:
- High sensitive
- Ultra Low Power
- Embedded temp comp
- Small and thinner

Main features

- Highest Accuracy (pressure): up to 8cm in vertical resolution
- Unprecedented ascent and descent speed estimation
- Ultra Low Power consumption 3uA LP – 15uA HR
- Patented Package Technology enabling Water and Dust resistance
- Embedded FIFO for Pressure and Temperature data

Humidity
HTS221

2x2x0.9 mm

Key advantages:
- Best tradeoff between performance and current consumption

Main features

- Humidity accuracy ±3.5% rH, 20% to 80%rH
- -40°C to 120°C operating temperature range
- Temperature accuracy ±0.5°C, 15 to 40°C
- Flexible ODR from 1Hz to 12.5Hz
- Low power consumption 2uA @ 1Hz ODR
- Extended supply voltage (1.7 to 3.6 V)

life.augmented

Microphone MEMS

Microphone
MP34DT01
MP34DT05

3x4x1 mm

Key advantages:
- High SNR
- High AOP
- Flat Frequency Response
- Sensitivity matching

Main features

- Omni-directional PDM output, industry standard footprint (3x4mm)
- High SNR (64dB) allowing improved audio fidelity experience
- High AoP (122.5dBSPL) among TOP port MEMS microphones
- Ground ring around the port hole to improve ESD robustness

Software

Acoustic Open.Audio Free Libraries, to:
- Estimate the arrival of audio signal
- Create a virtual directional microphone/Beam Forming
- Acoustic Echo Cancelation

Open.Audio

life.augmented

ST is WWide
#1 ToF
supplier

Over 3 years of proven track record in manufacturing
>100M products shipped

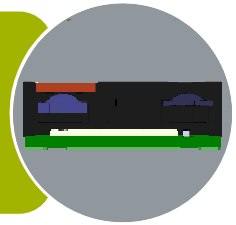
300% AAGR

Time of Flight Sensor : VL53L0X

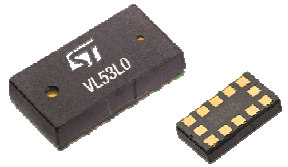
Single Photon Avalanche Diode
Ultra fast time resolution enabling Direct ToF processed in ST CMOS SPAD process

ST Proprietary Time-of-Flight IP
Best compromise of cost, complexity & power vs performance

Compact integrated system
Sensor, filters, optics, VCSEL and driver integrated
Fully calibrated system



Optimised and reliable supply chain
High volume & low cost

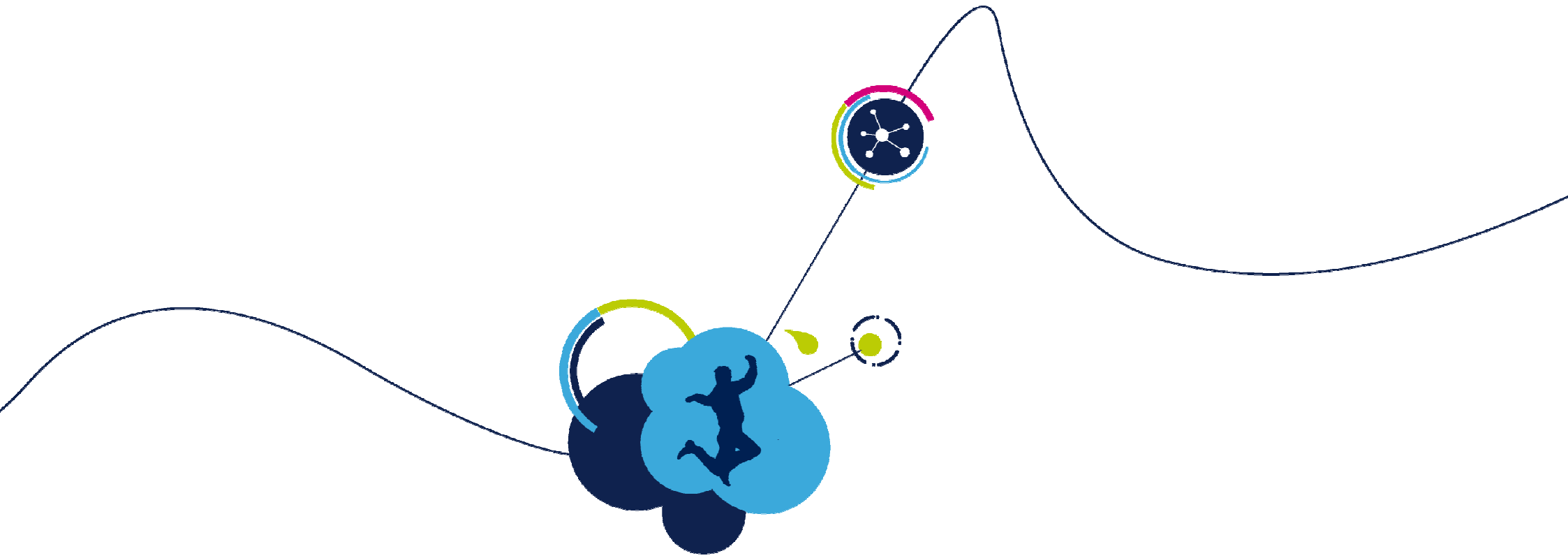


OLGA **4.4 x 2.4 x 1mm**

PRODUCT HIGLIGHTS

- **Fully integrated** (IR 940nm Vcsel emission, filters, SPAD receiving array, advanced µC)
- Range up to **200cm in less than 30ms**
- **High accuracy (typ. 5%)**
- **Low power** (stdby 5uA, active <20mW at 10Hz)
- FoV : 25°
- Laser Class1 device (eye safe)
- Excellent ambient light robustness (**940nm**)
- Works with cover glass
- Complete API package and **Android driver**

life.augmented



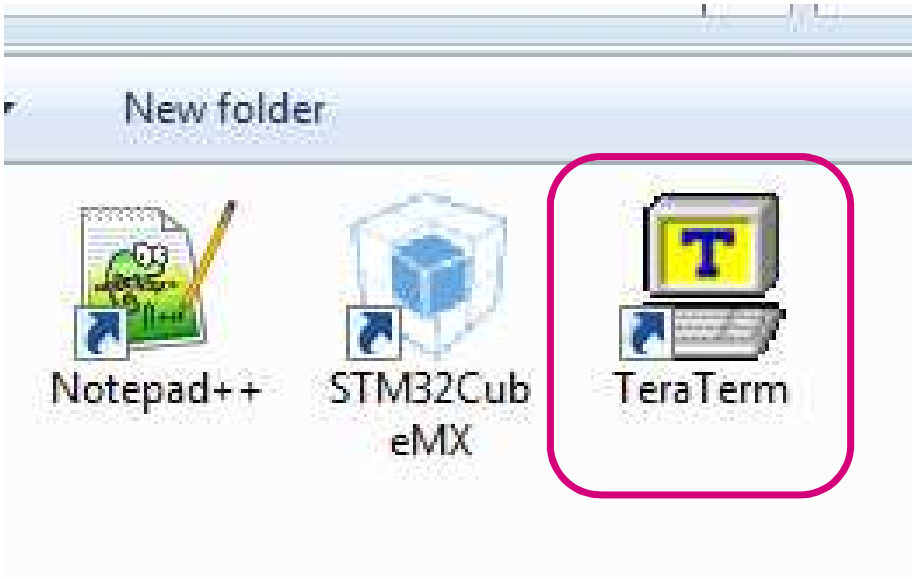
Lab 2 : Real Time Sensor Logging

life.augmented

Goal

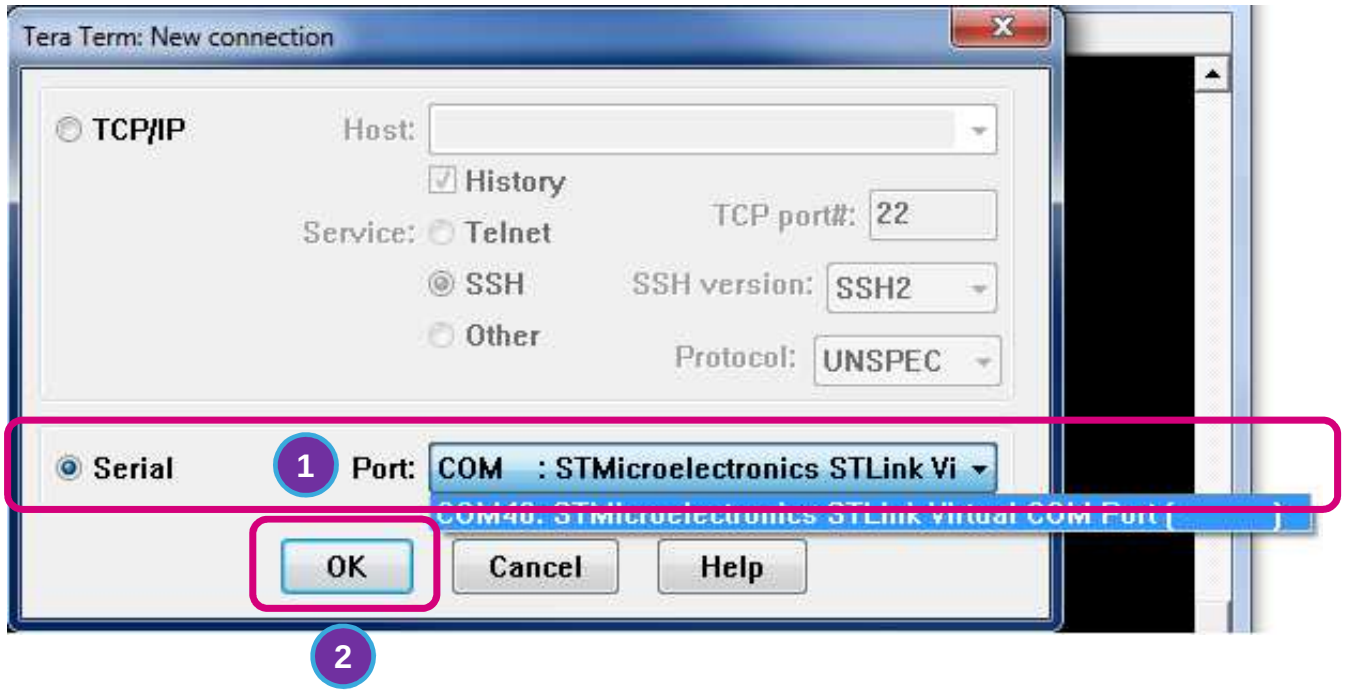
- In this example we are going to demonstrate how to:
  - Read the on-board MEMS sensor data using the sensor library
  - Use IAR
    - Breakpoints
    - Local Watch to view local variables
    - Live Watch to view global variables
  - Use Tera Term to exchange data between STM32 and a PC over USB
- Close the previous STM32CubeMX project and Tera Term

Open TeraTerm



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

Tera Term Configuration 1/5

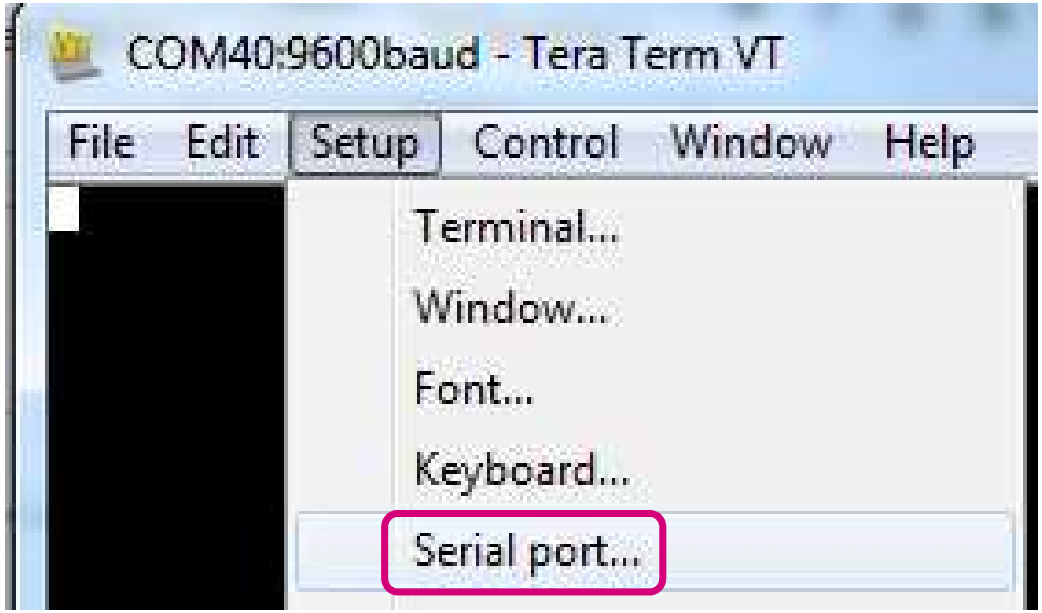


1. Select the **STMicroelectronics STLink Virtual COM Port**
2. Click **OK**

life.augmented

Tera Term Configuration 2/5

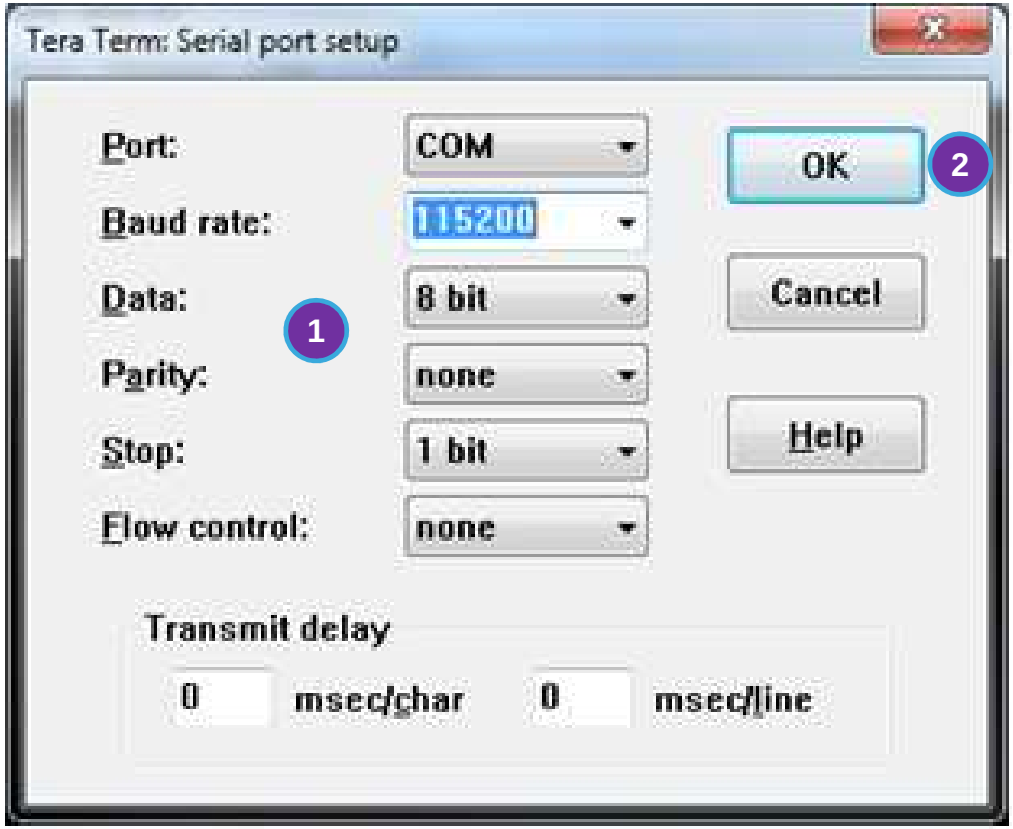
1. Click **Setup** -> **Serial port…**



Tera Term Configuration 3/5

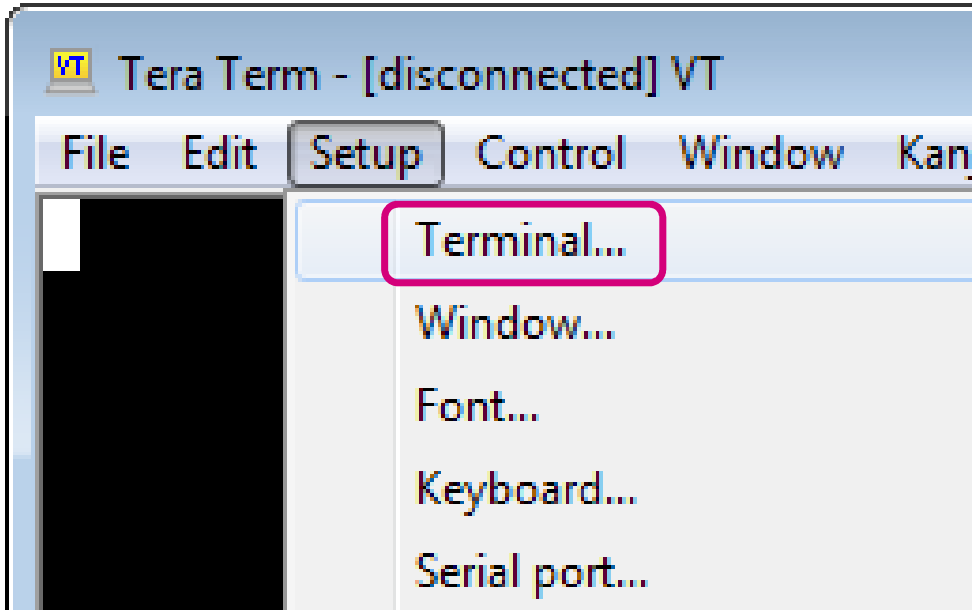
1. Set the following:
   - Baud rate : **115200**
   - Data : **8 bit**
   - Parity : **none**
   - Stop : **1 bit**
   - Flow control : **none**

2. Click **OK**



Tera Term Configuration 4/5

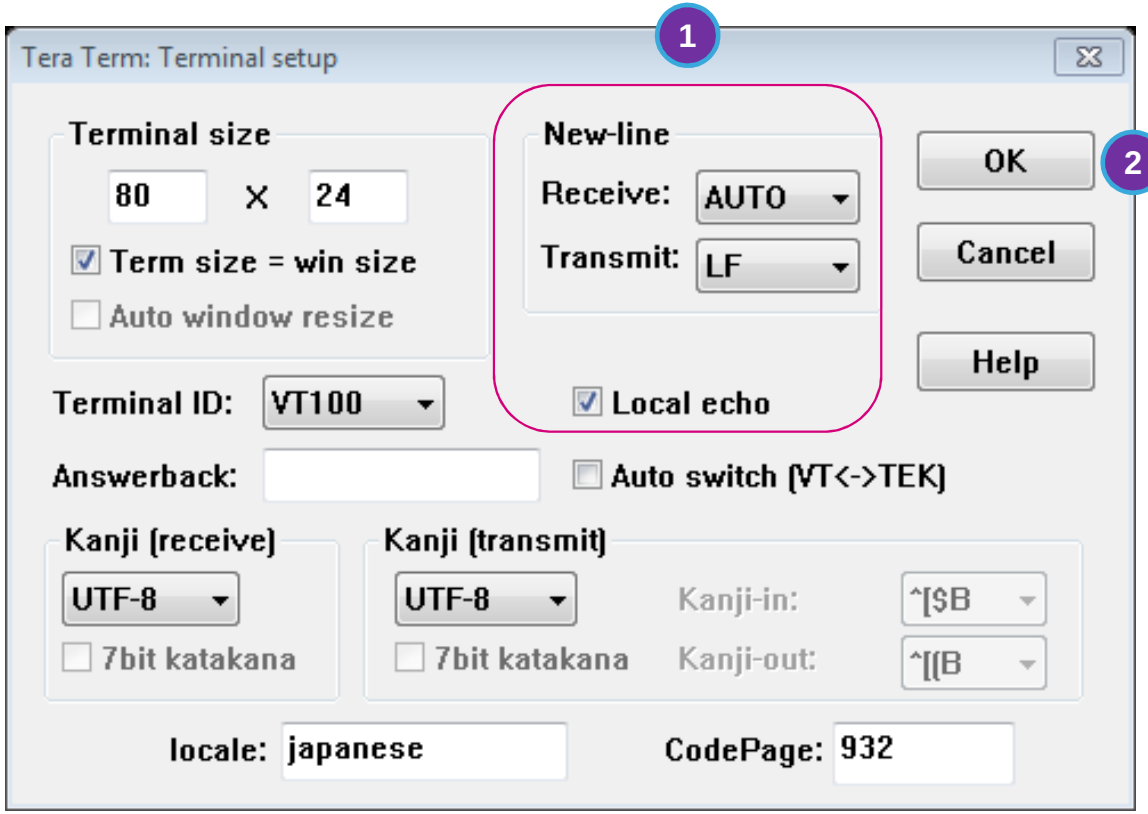
1. Click **Setup** -> **Terminal…**



Tera Term Configuration 5/5

1. Set the following:
   - New-Line Receive : **AUTO**
   - New-Line Transmit : **LF**
   - Enable **Local echo**

2. Click **OK**



life.augmented

Open the MEMS Project

- We are going to configure the **MEMS** project to view data coming from the on-board accelerometer in both raw and normalized formats.

1. Close the previous IAR project.
2. Double click on the **Project.eww** file located under:

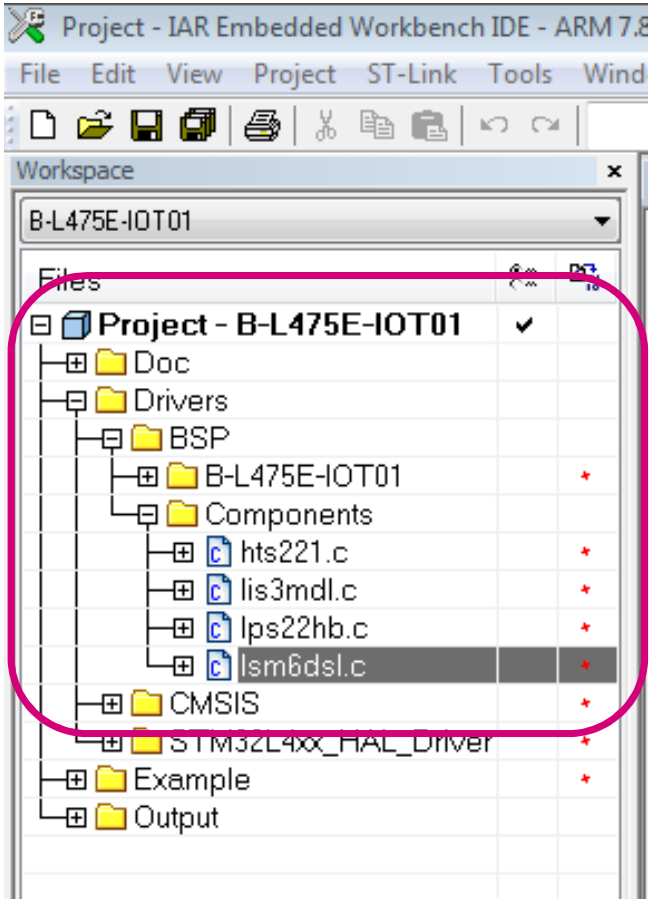
C:\STM32L4_DK_IoT_Node_Seminar\Hands-on\Labs\Projects\B-L475E-IOT01\Examples\BSP\EWARM

life.augmented

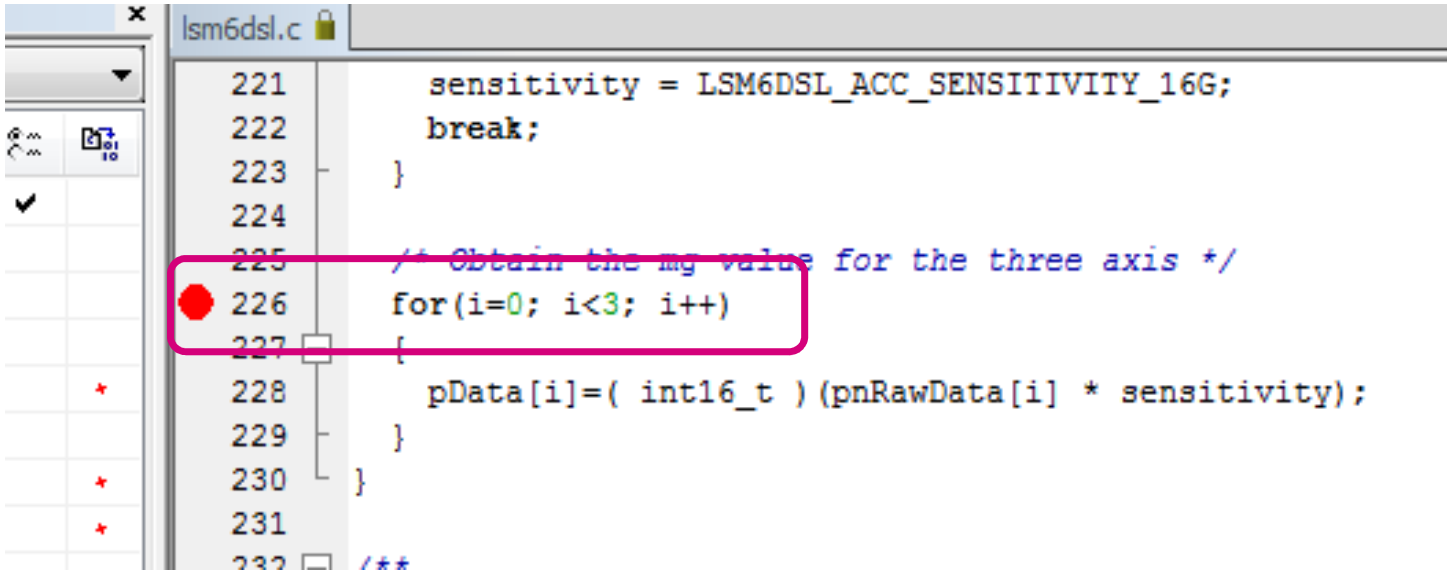
Setting a Breakpoint

Expand the file tree:

1. Expand **Project – B-L475E-IOT1**
2. Expand **Drivers**
3. Expand **BSP**
4. Expand **Components**
5. Double-click **lsm6dsl.c**



Setting a Breakpoint

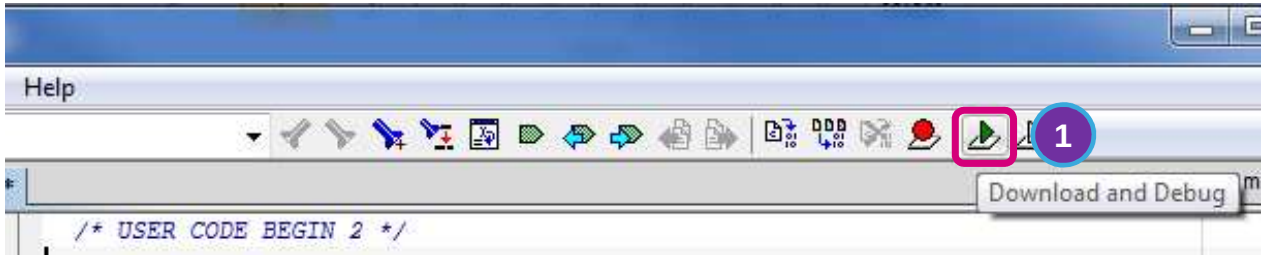


- Set a break point at line **226** by left-clicking on the left side of the line number.

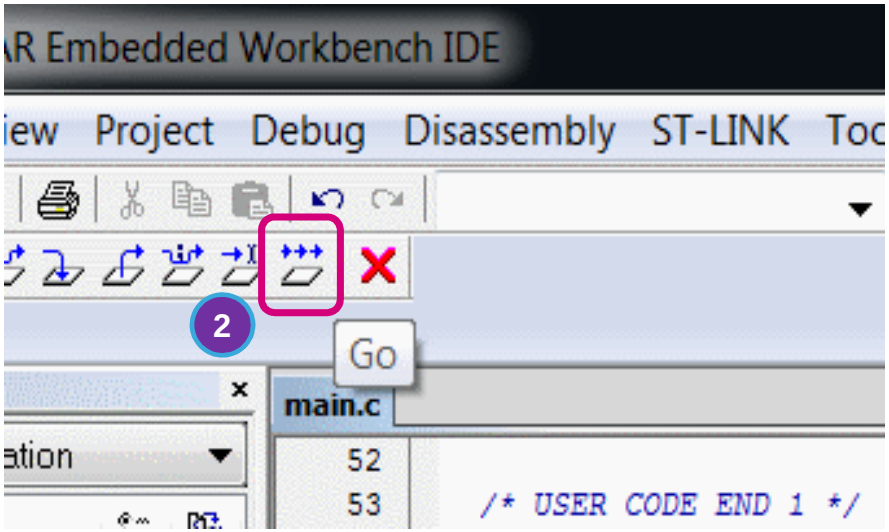
life.augmented

Load and Run

1. Click the GREEN ARROW to Build the Project, **Download** and start the **debugger**. (Ctrl + D)



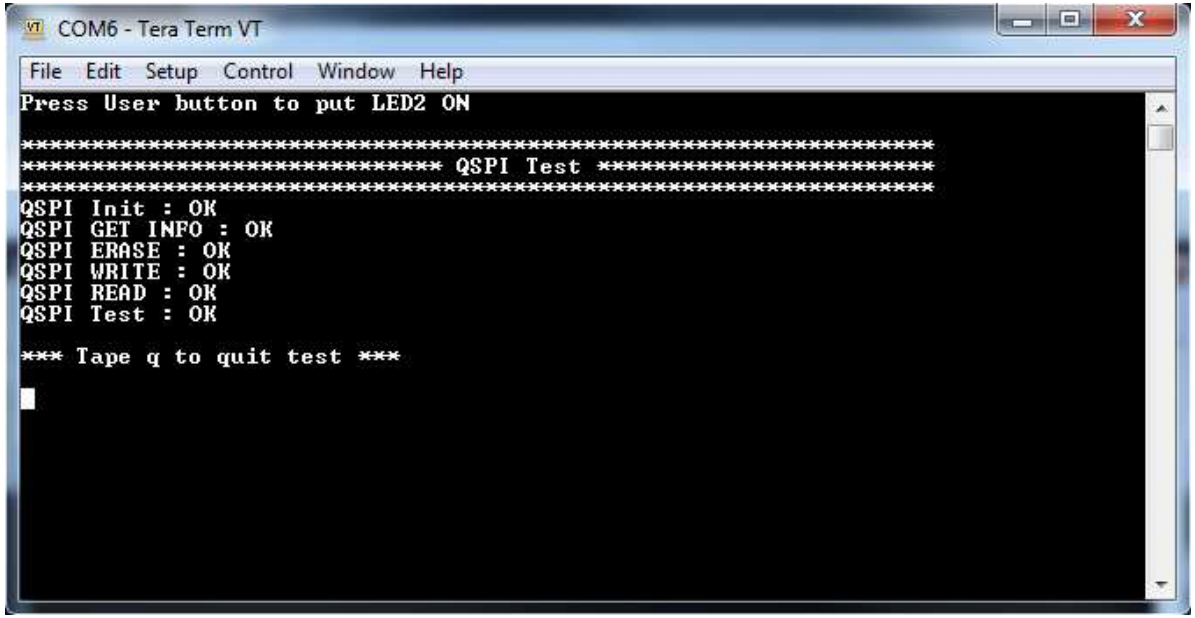
2. Click the triple-arrow **GO** button! (F5)



Debugging the Firmware : Hitting the Breakpoint

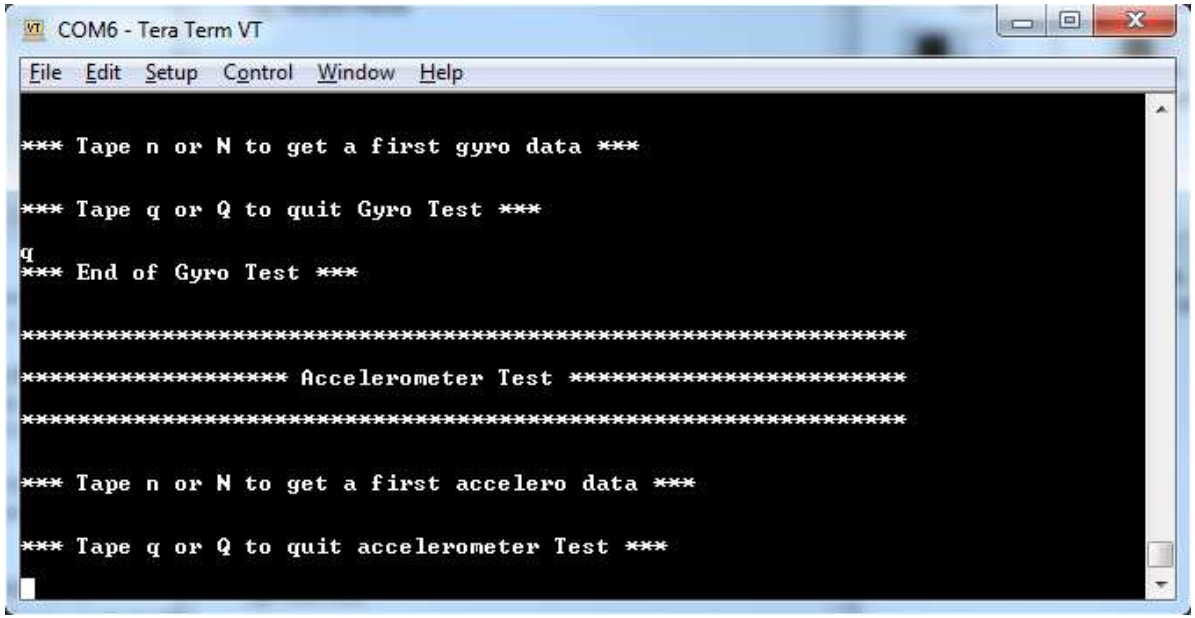
- In Tera Term, you will be asked to press the **BLUE** User button.
  - This will turn on **LED2**

Debugging the Firmware : Hitting the Breakpoint



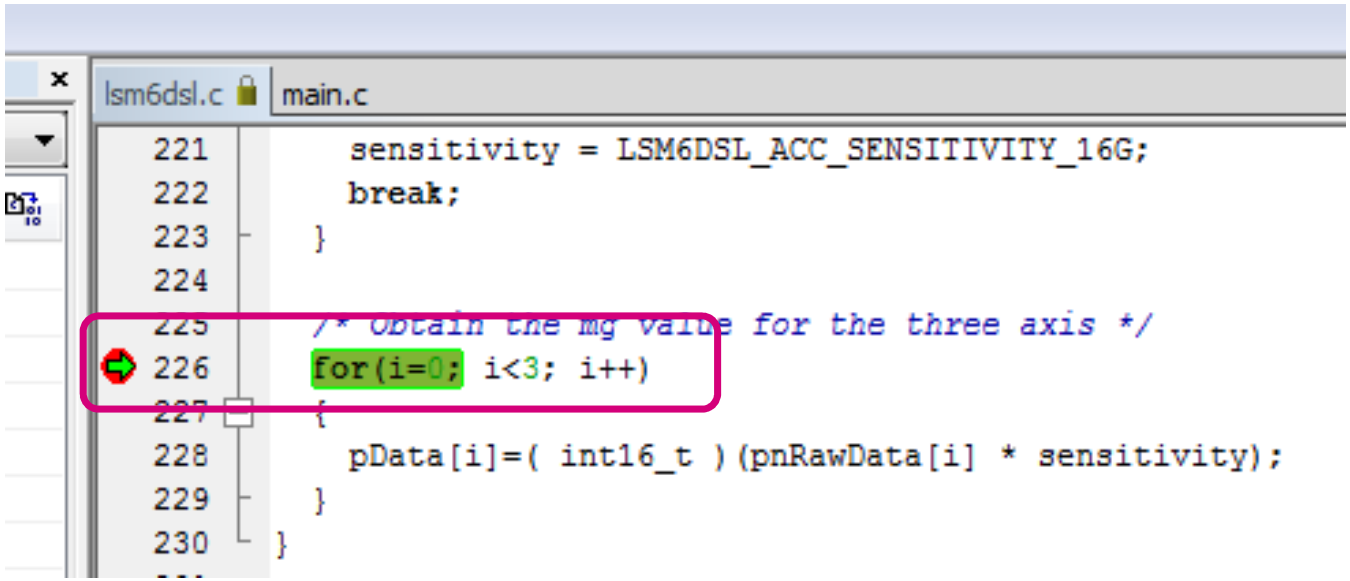
- Hit **Q** to quit each test option until you get to the **Accelerometer Test**

Debugging the Firmware : Hitting the Breakpoint



- At the **Accelerometer Test**, hit **N** to retrieve real time accelerometer data

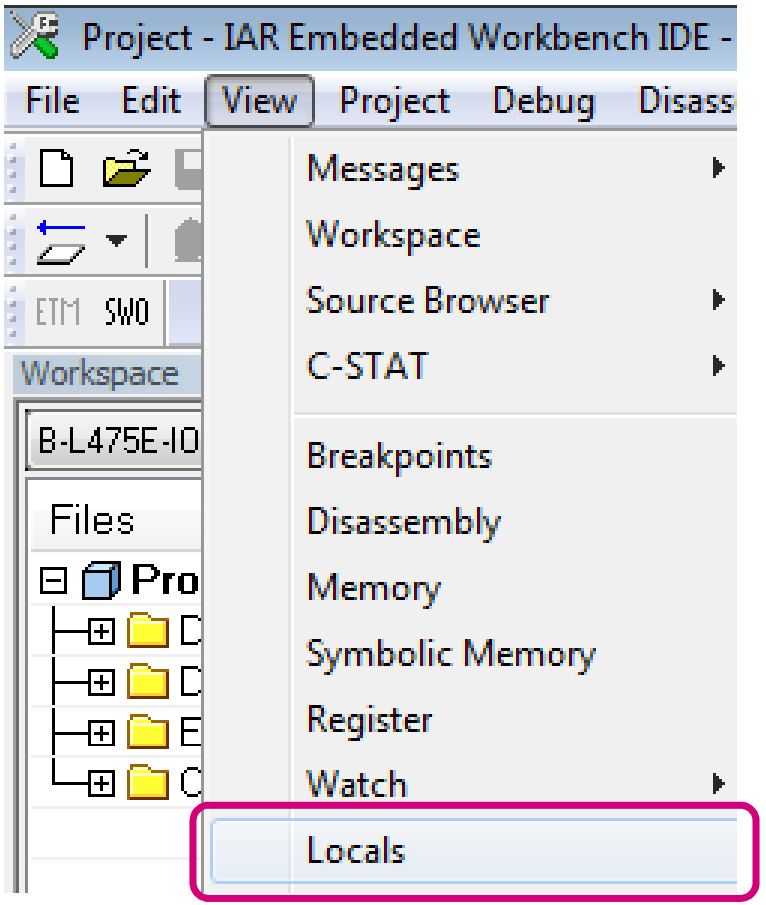
Debugging the Firmware : Hitting the Breakpoint



- In IAR, you will hit the previously set breakpoint at line **226**.

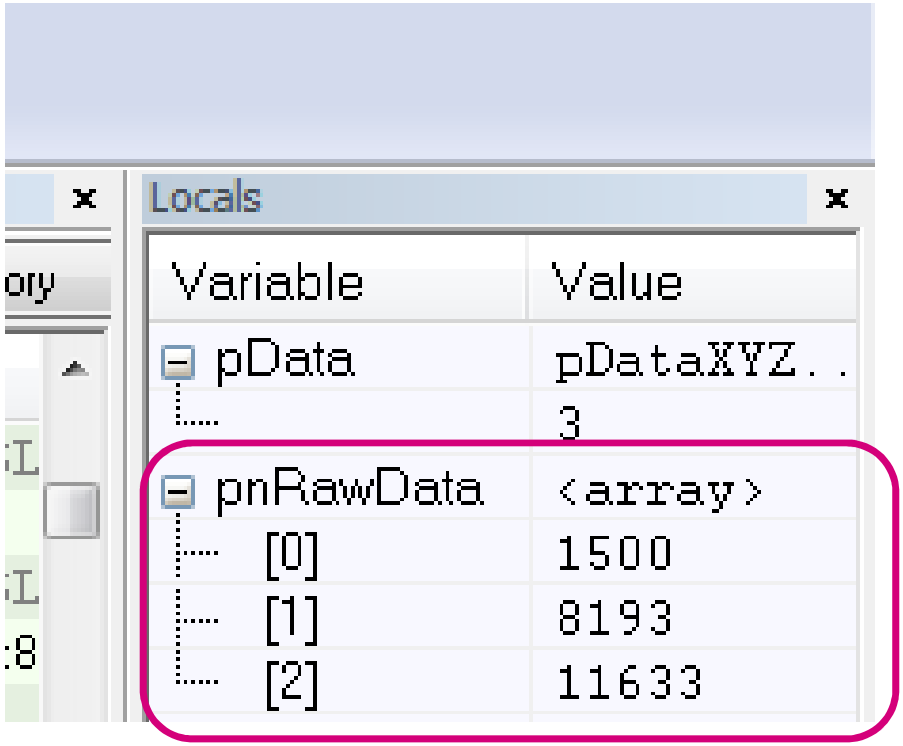
Debugging the Firmware : Viewing Local Variables

- In IAR, click on **View** -> **Locals**



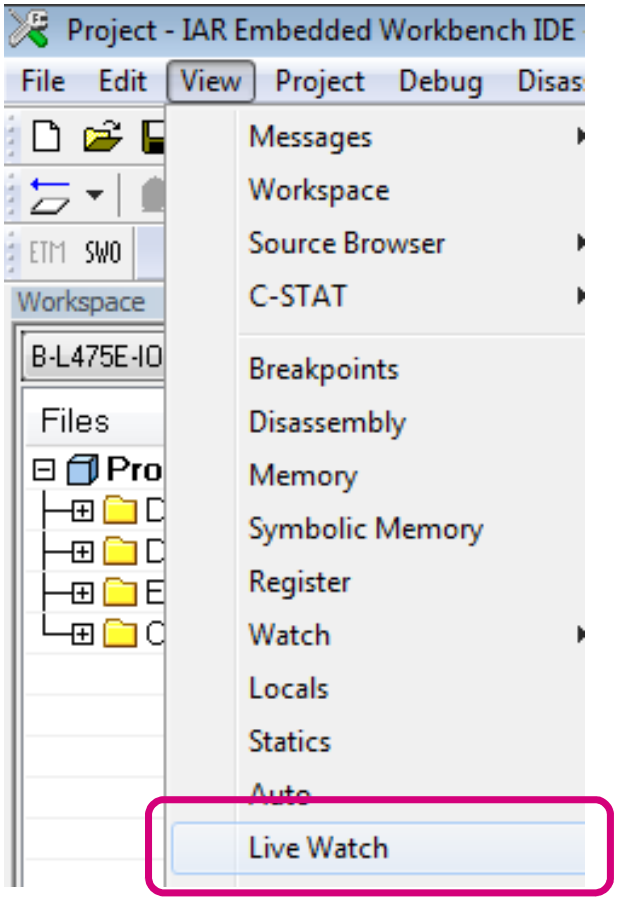
Debugging the Firmware : Viewing Local Variables

- A sub window will open on the right side of the IDE, with the title “Locals”.
- Expand the **`pnRawData`** variable array to view the raw accelerometer reading.



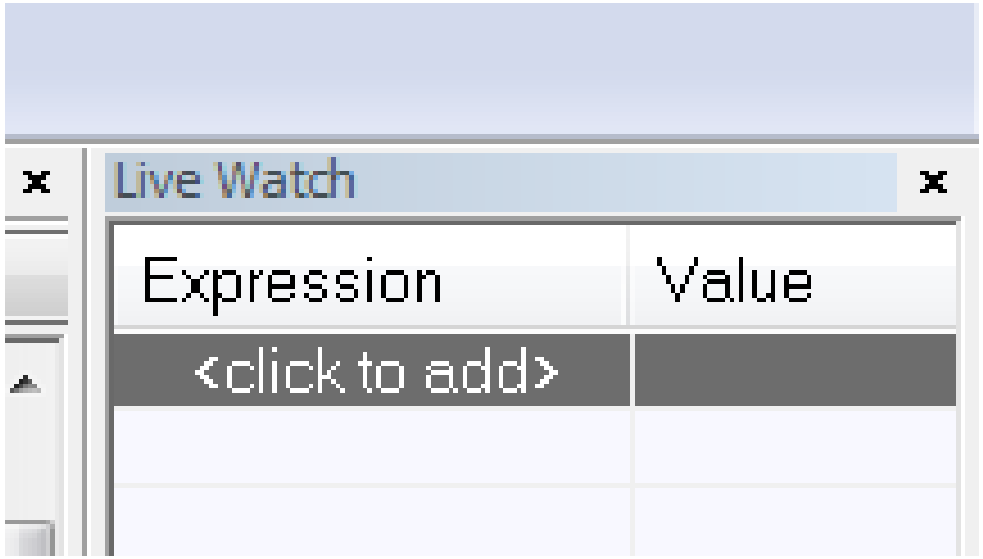
Debugging the Firmware : Viewing Global Variables

- In IAR, click on **View** -> **Live Watch**



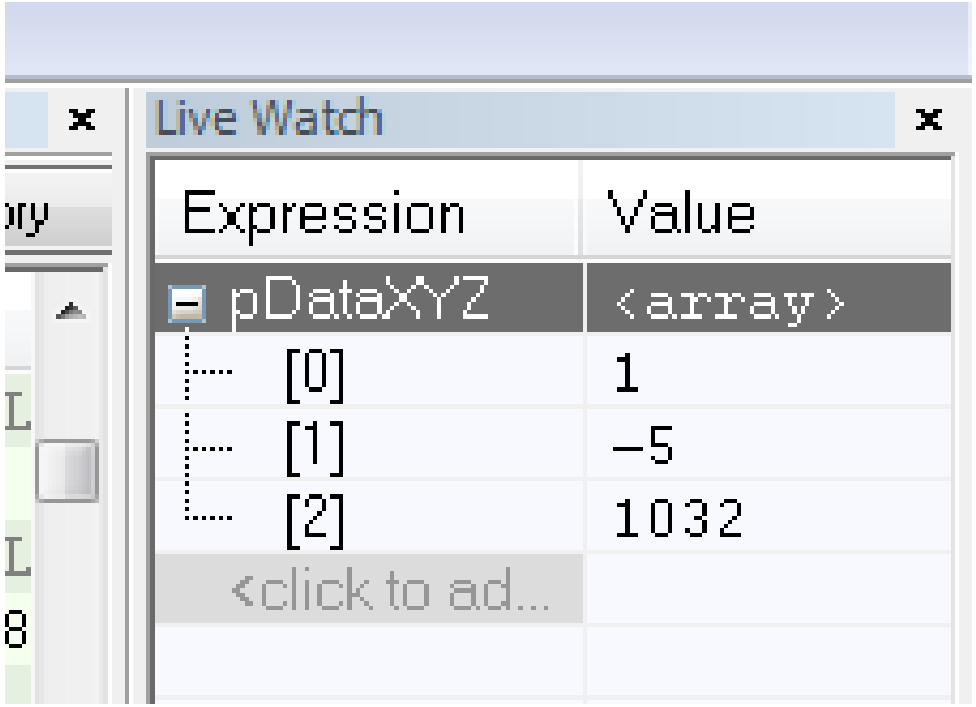
Debugging the Firmware : Viewing Global Variables

- A sub window will open on the right side of the IDE, with the title “Live Watch”
- Click on **<click to add>**



Debugging the Firmware : Viewing Global Variables

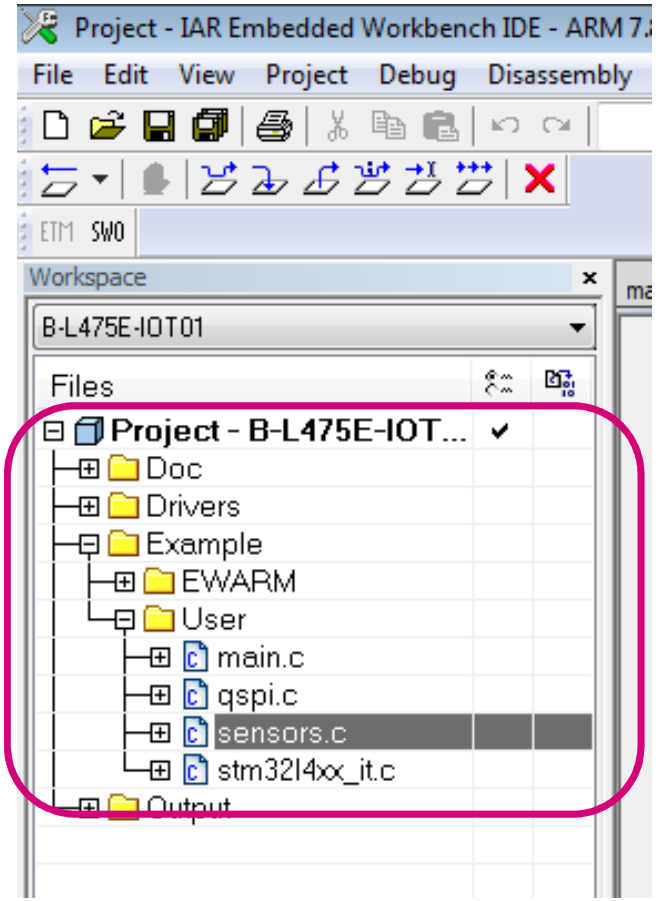
- Type **`pDataXYZ`** (case-sensitive)
- Expand **`pDataXYZ`** variable array to view the accelerometer reading.



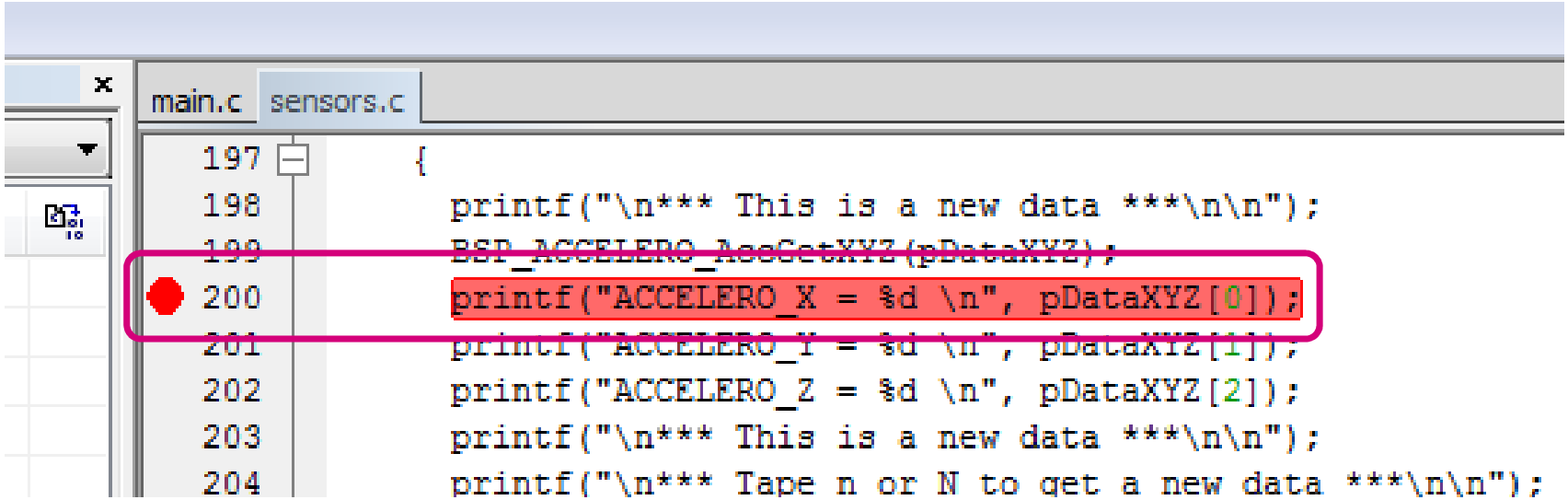
Debugging the Firmware : Viewing Global Variables

Expand the file tree:

1. Expand **Project – B-L475E-IOT1**
2. Expand **Example**
3. Expand **User**
4. Double-click **sensors.c**



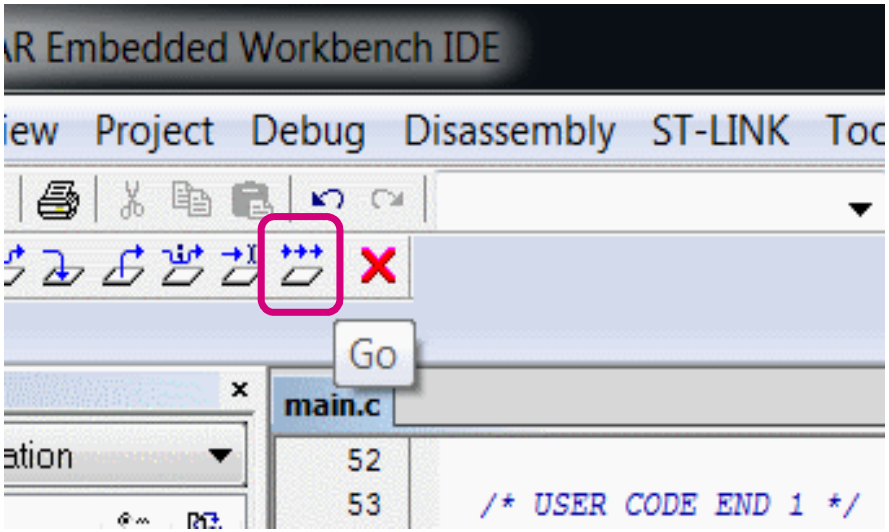
Debugging the Firmware : Viewing Global Variables



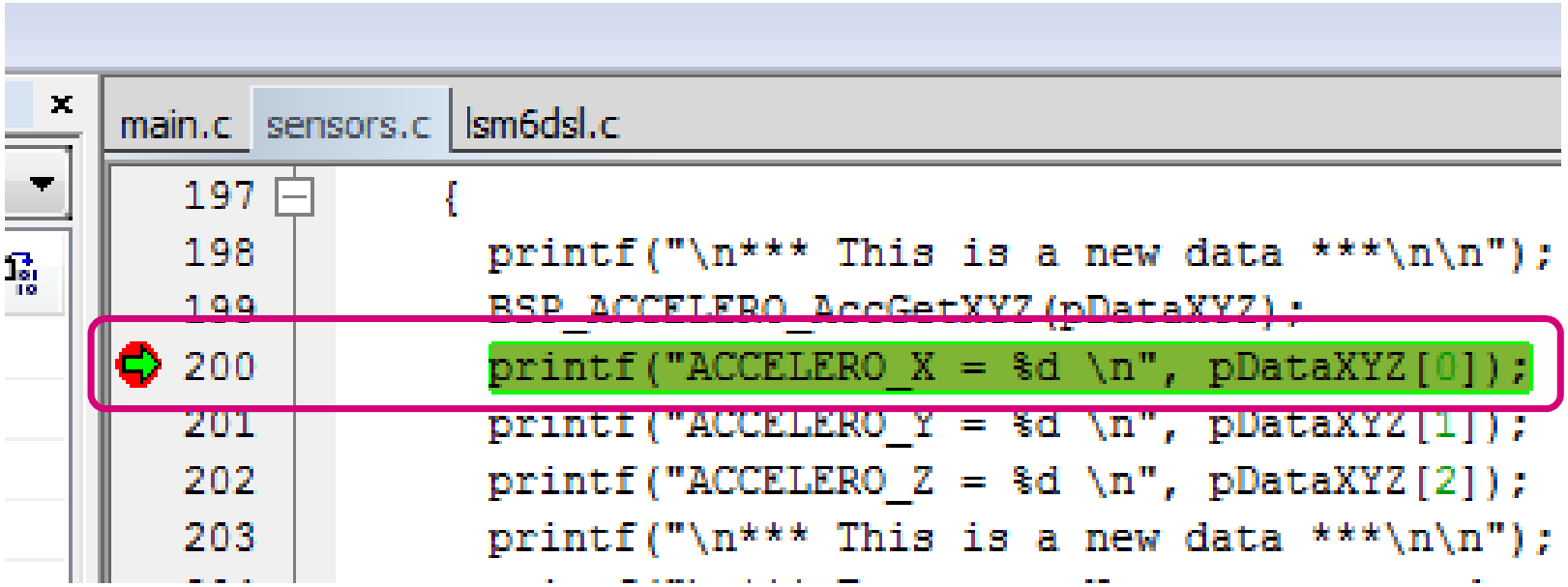
- Set a break point at line **200** by left-clicking on the left side of the line number.

Debugging the Firmware : Viewing Global Variables

- Click the triple-arrow **GO** button! (F5)



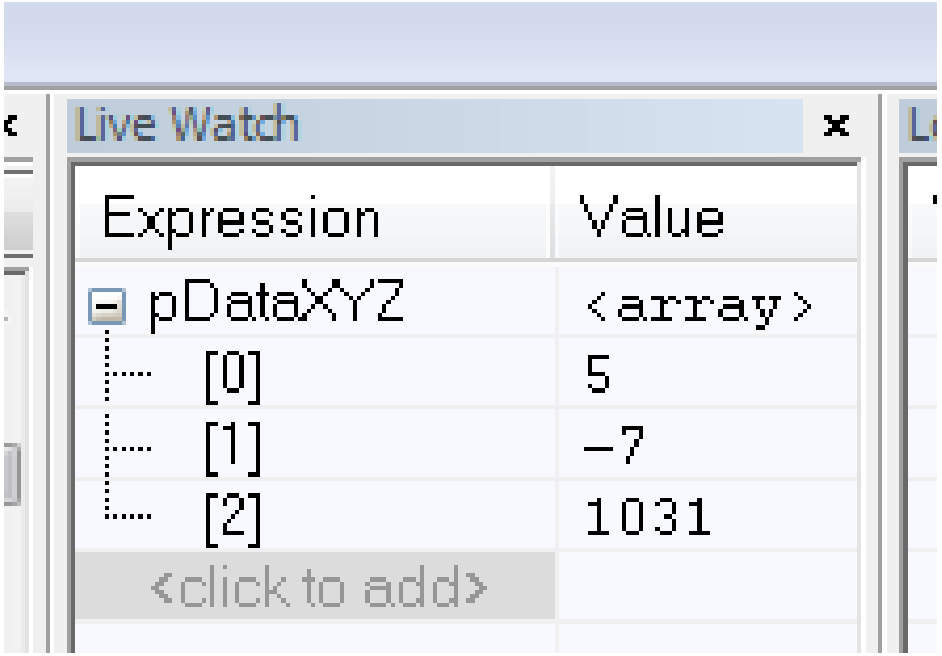
Debugging the Firmware : Viewing Global Variables

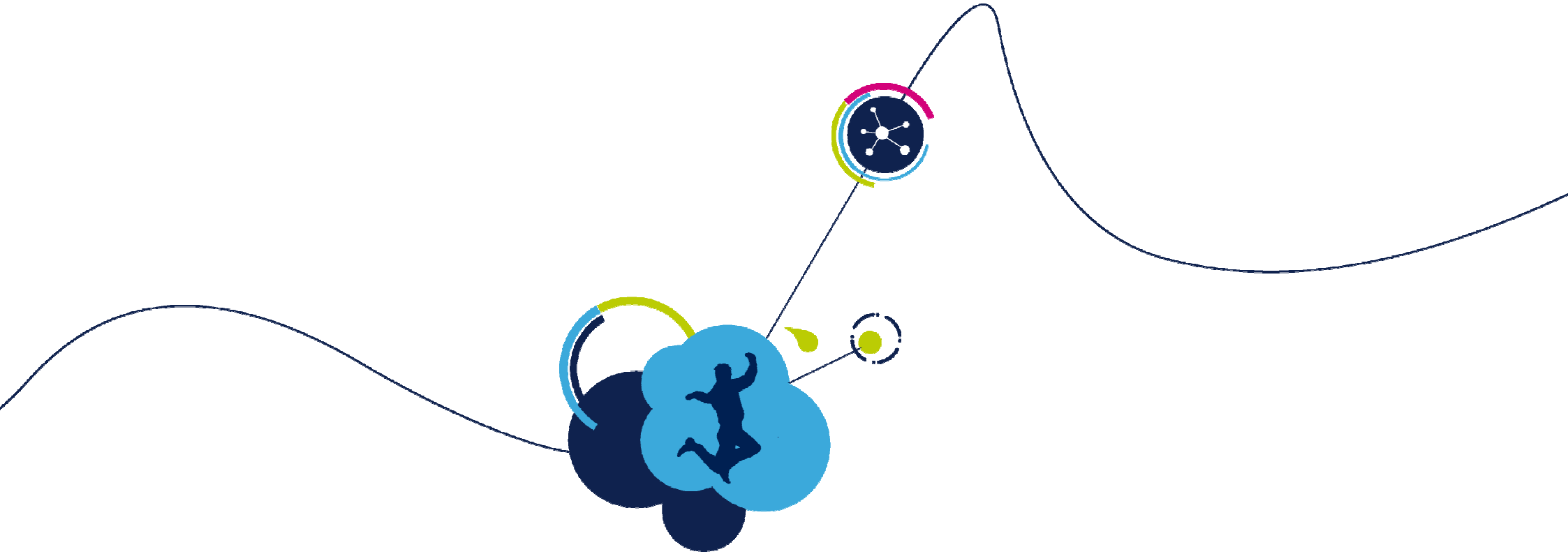


- In IAR, you will hit the previously set breakpoint at line **200**.

Debugging the Firmware : Viewing Global Variables

- In the Live Watch window, **pDataXYZ** will now have the normalized accelerometer data (in units of ‘g’).





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

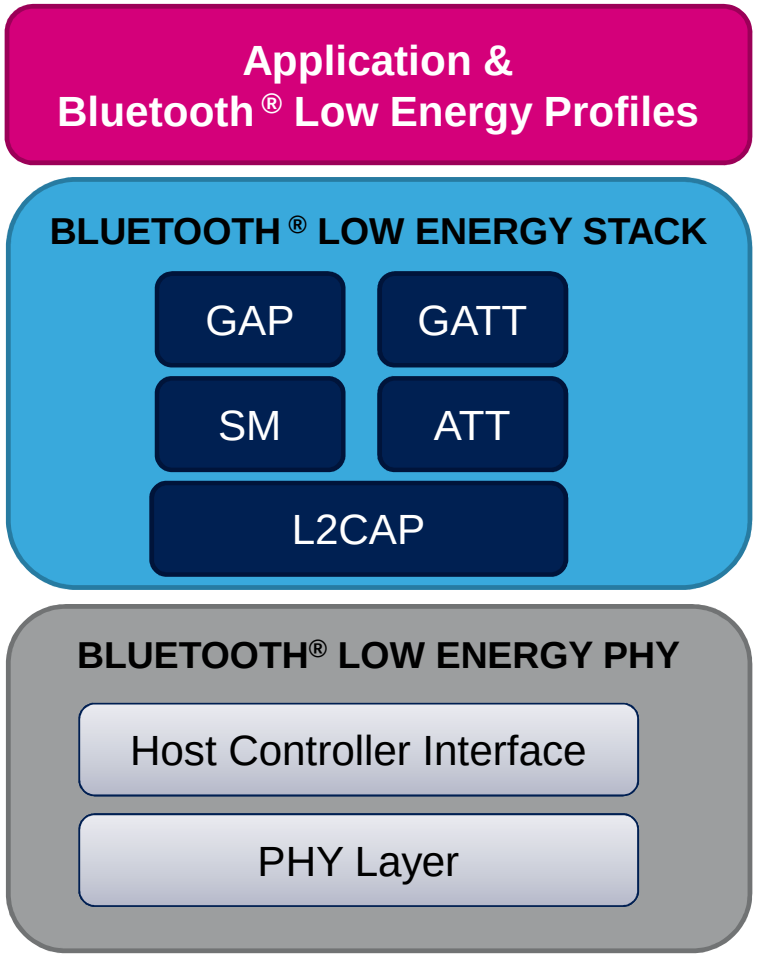
2017 **Back to one flavor**

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

life.augmented

ST BlueNRG-MS Solution

Integration

Flexibility

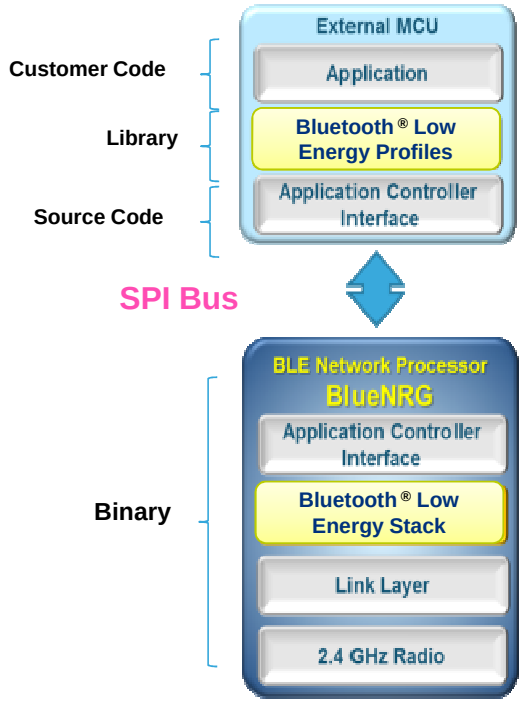
Low power

Best in Class

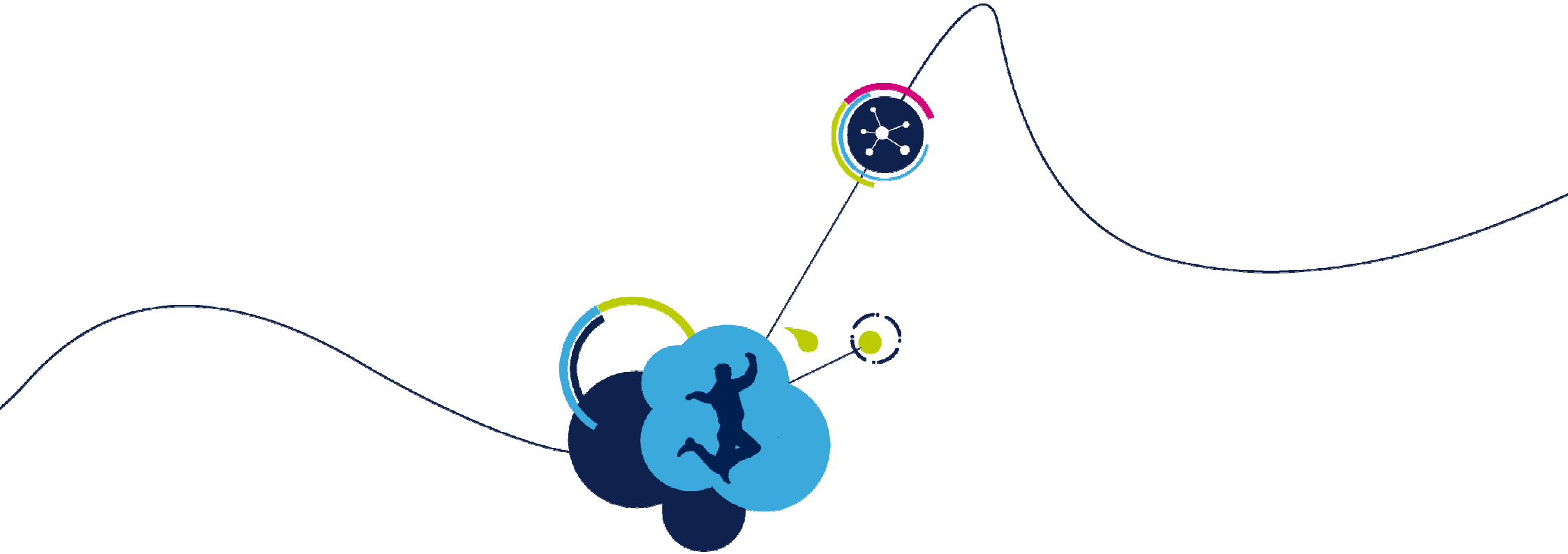
Small size

- **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



Lab 3 : Bluetooth® Low Energy Pairing

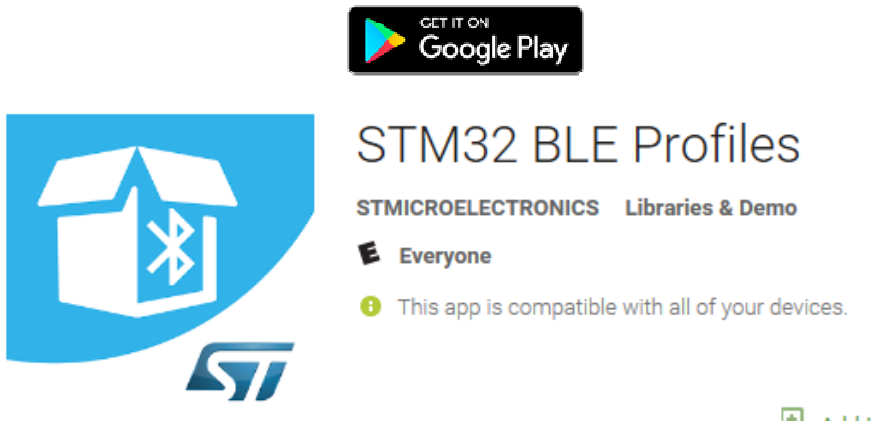
life.augmented

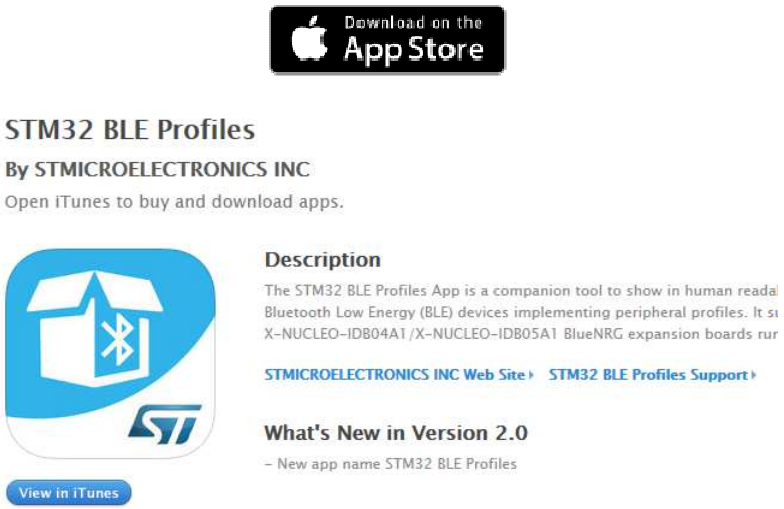
Lab Goal

- 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.





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.

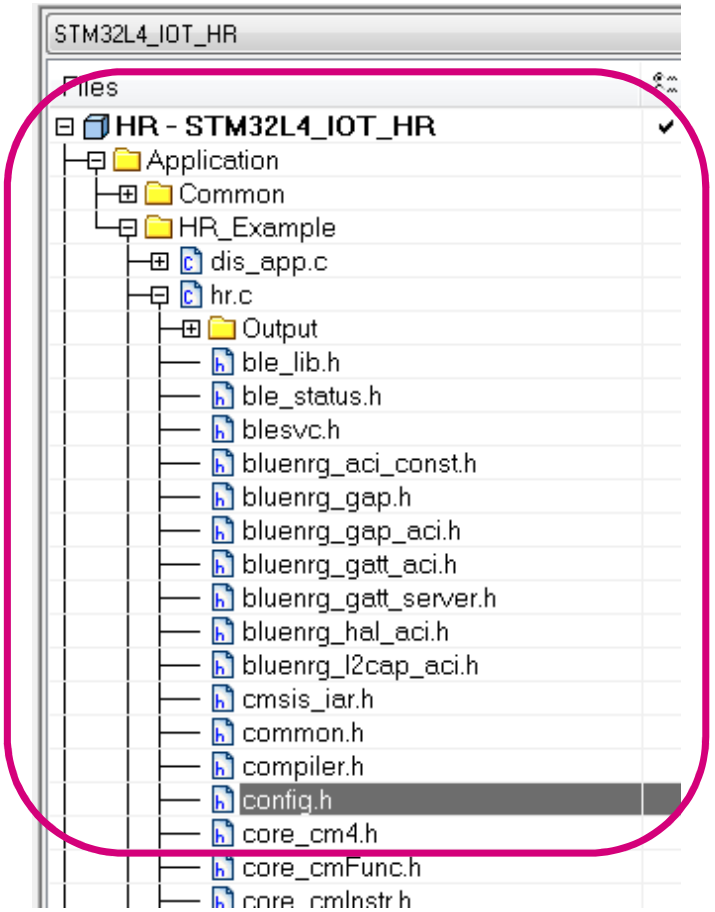
1. Close the previous IAR project.
2. Double click on the **`HR.eww`** file located here:

`C:\STM32L4_DK_IoT_Node_Seminar\Hands-on\Labs\Projects\B-L475E-IOT01\Applications\BLE\HeartRate\EWARM`

BlueNRG Module configuration (1/4)

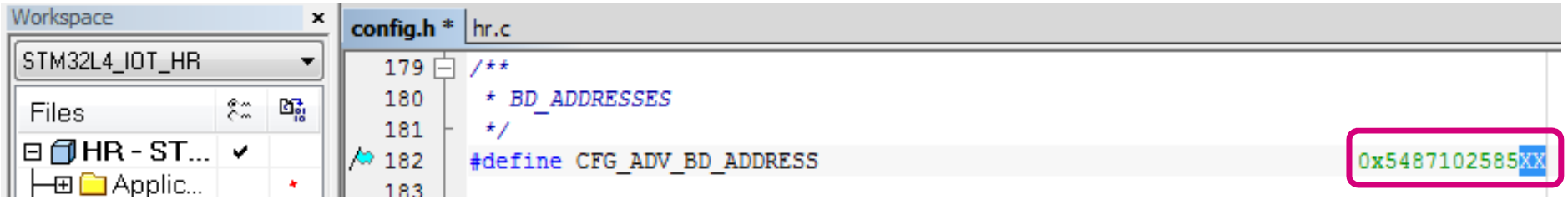
Expand the file tree:

1. Expand **HR – STM32L4_IOT_HR**
2. Expand **Application**
3. Expand **HR_Example**
4. Expand **hr.c**
5. Double-click **config.h**



life.augmented

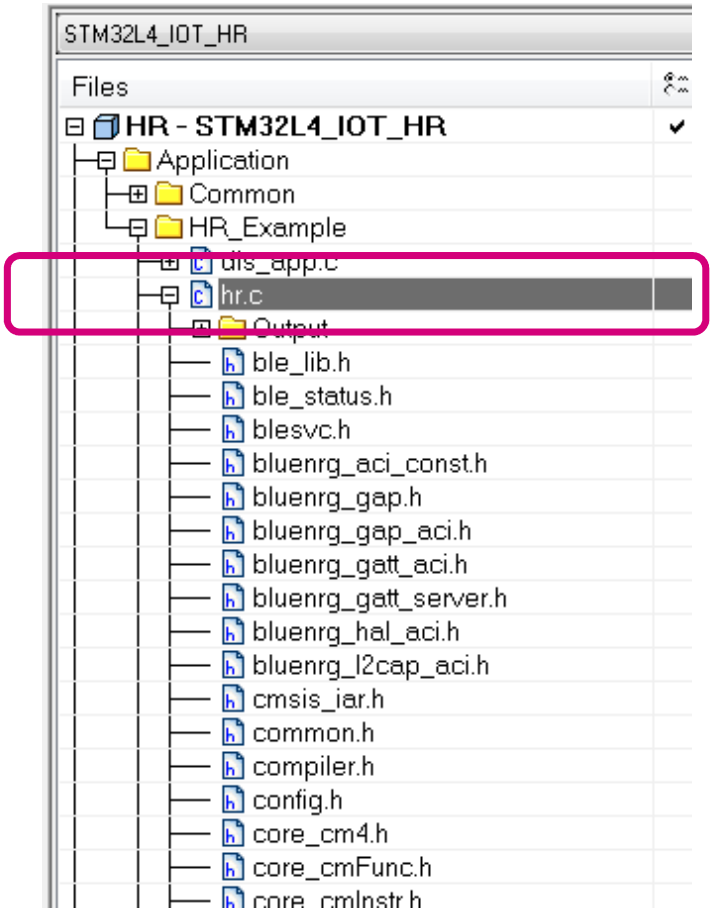
BlueNRG Module configuration (2/4)



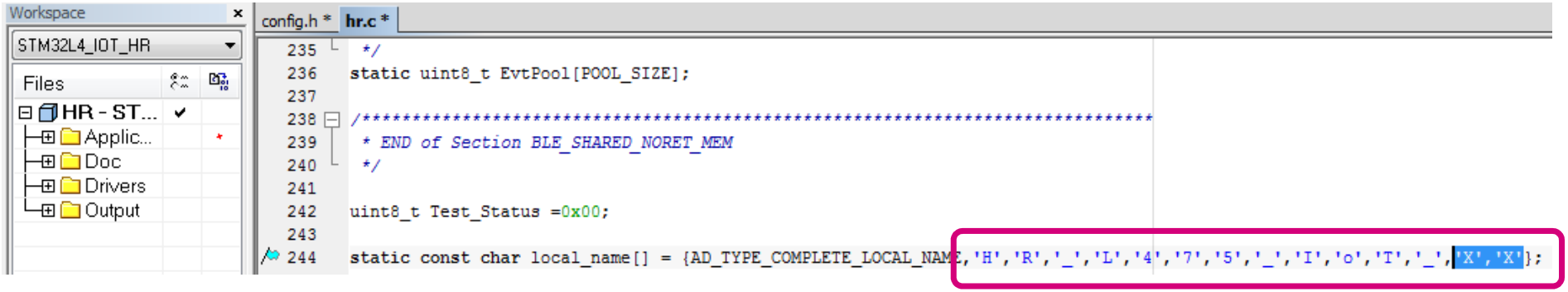
- Replace the '**XX**' in the **CFG_ADV_BD_ADDRESS** (line 182) with any desired number (2 characters) for customization, thus avoiding issues with 2 or more boards.

BlueNRG Module configuration (3/4)

1. Double-click **hr.c**



BlueNRG Module configuration (4/4)

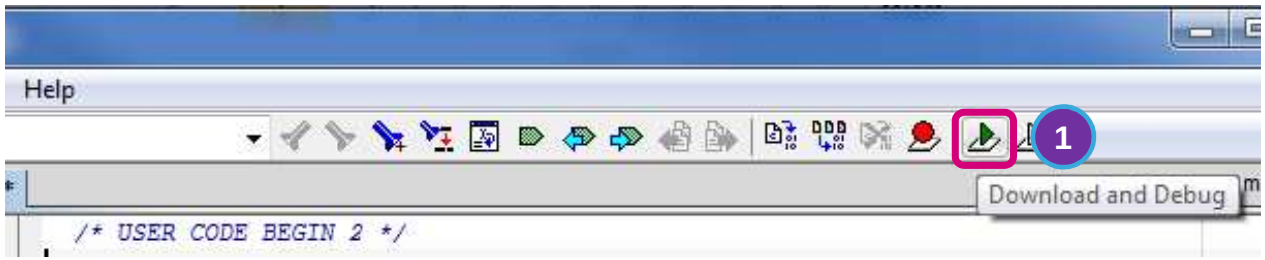


- Replace the `'X','X'` in the `local_name` (line 244) table with any number (2 characters).

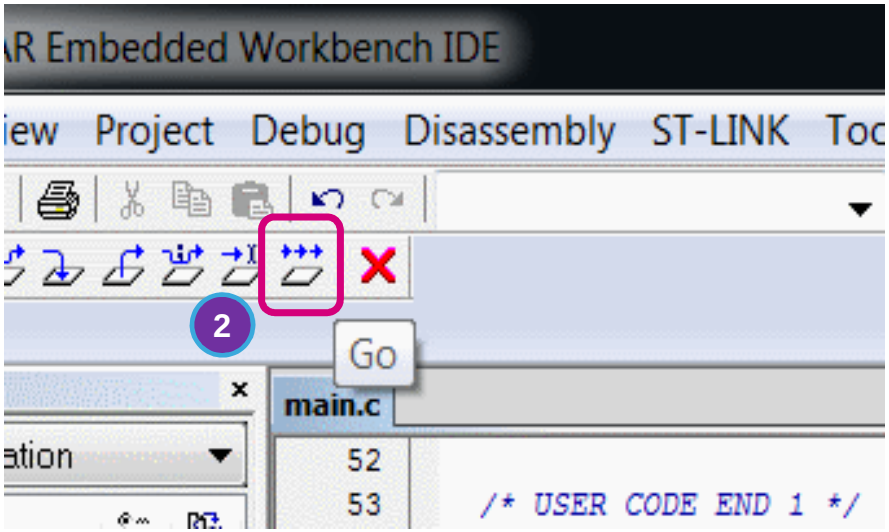
life.augmented

Load and Run

1. Click the GREEN ARROW to Build the Project, **Download** and start the **debugger**. (Ctrl + D)

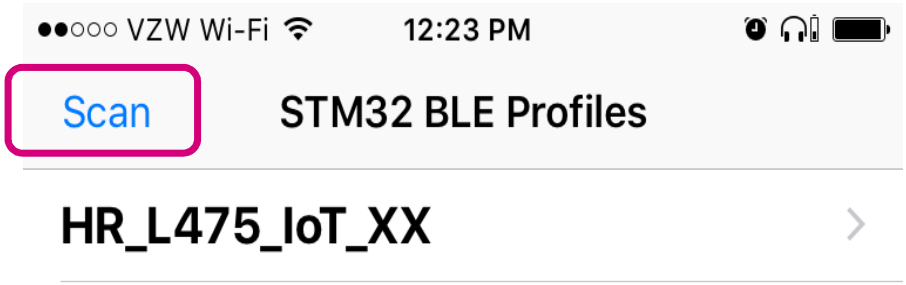


2. Click the triple-arrow **GO** button! (F5)

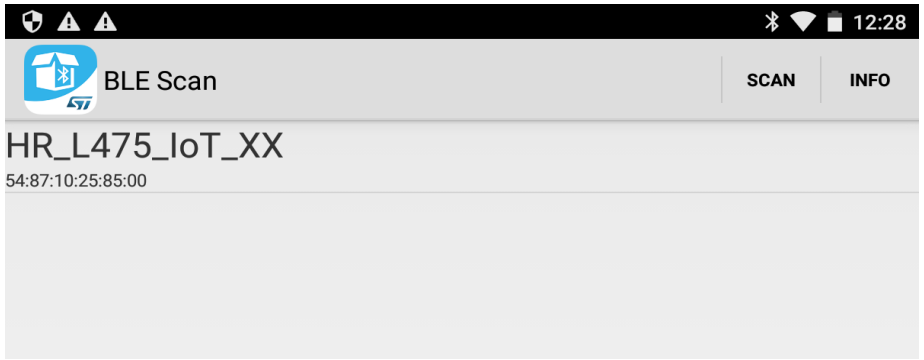


Pair with **STM32 BLE Profiles** App

iOS



Android

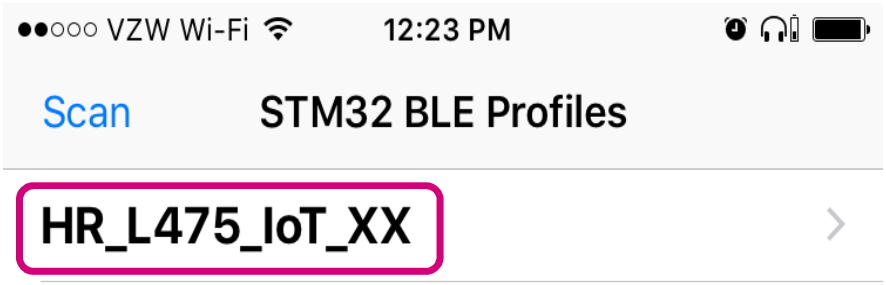


- Make sure Bluetooth® is active on your phone
- Using your phone, open the **STM32 BLE Profiles** app.
- For iOS users click on **Scan**.

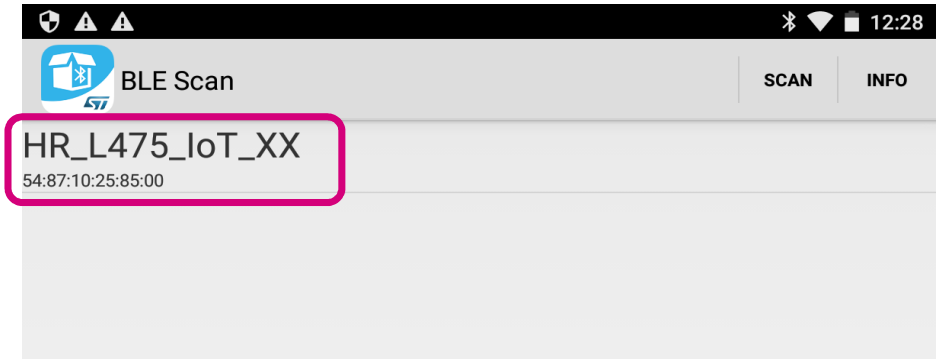
life.augmented

Pair with **STM32 BLE Profiles** App

iOS



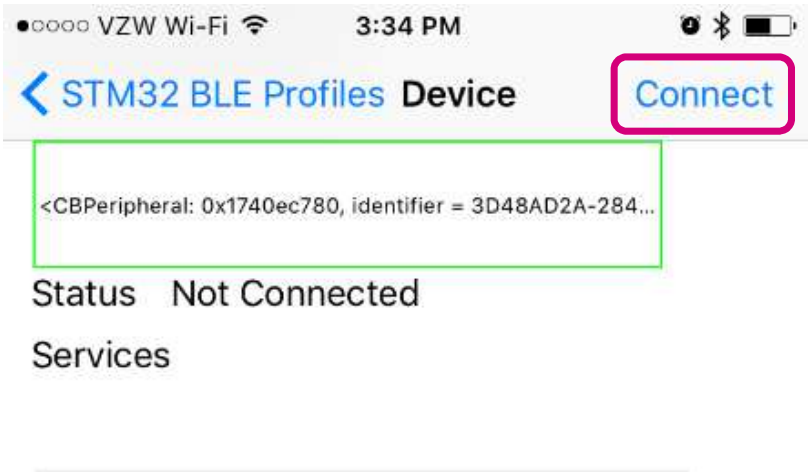
Android



- 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

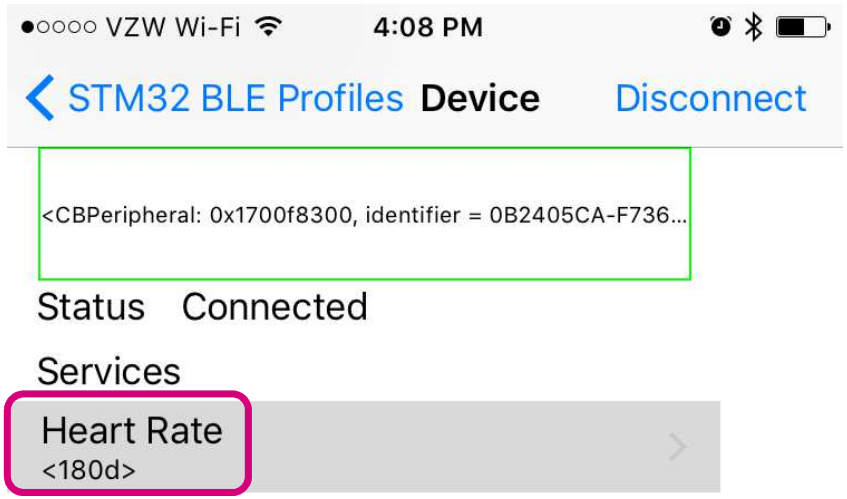
iOS



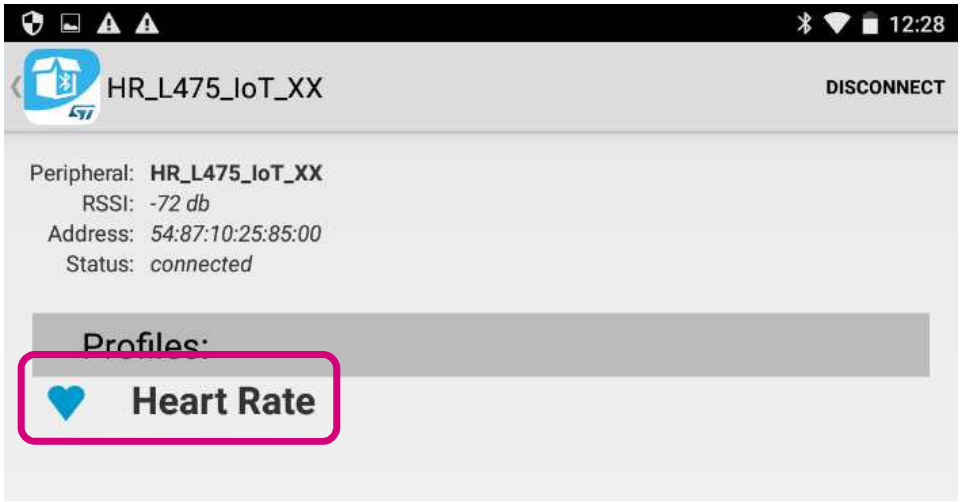
- iOS users, please click **Connect** on the next screen (iOS)

Select the Heart Rate Profile

iOS



Android

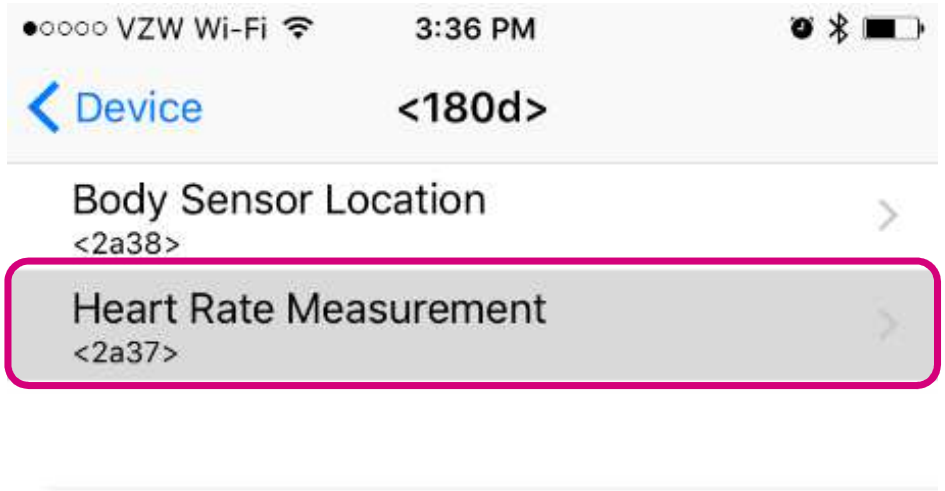


- Click on **Heart Rate** under **Services** (iOS) or **Profiles** (Android)

life.augmented

Select the Heart Rate Profile

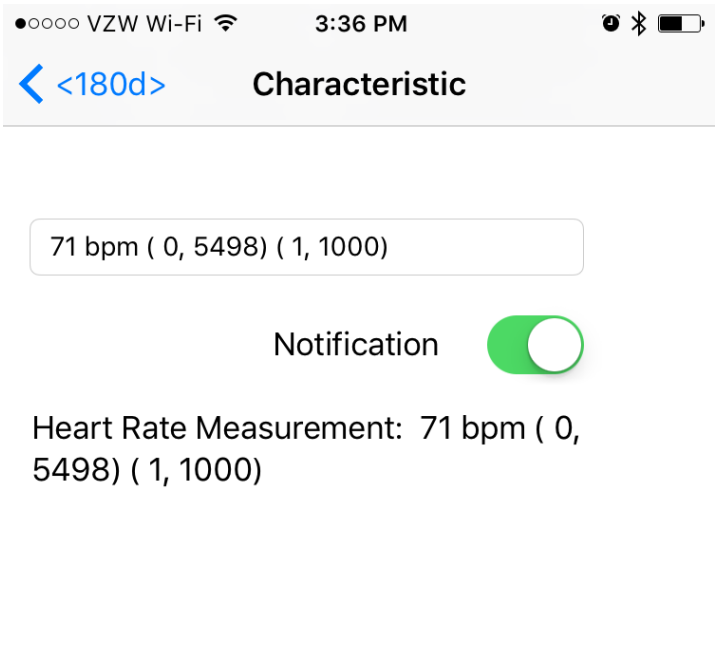
iOS



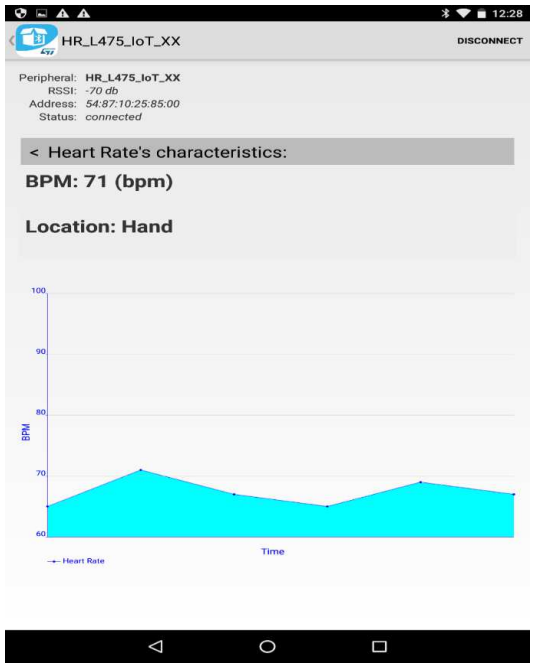
- iOS users, please click on **Heart Rate Measurement**

Display HR Data

iOS

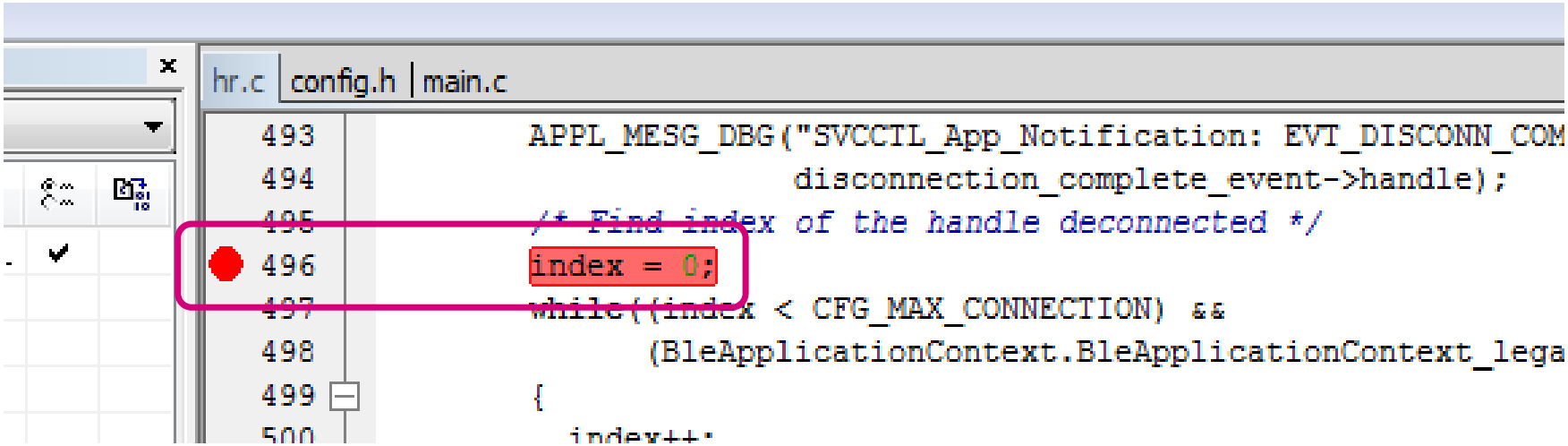


Android



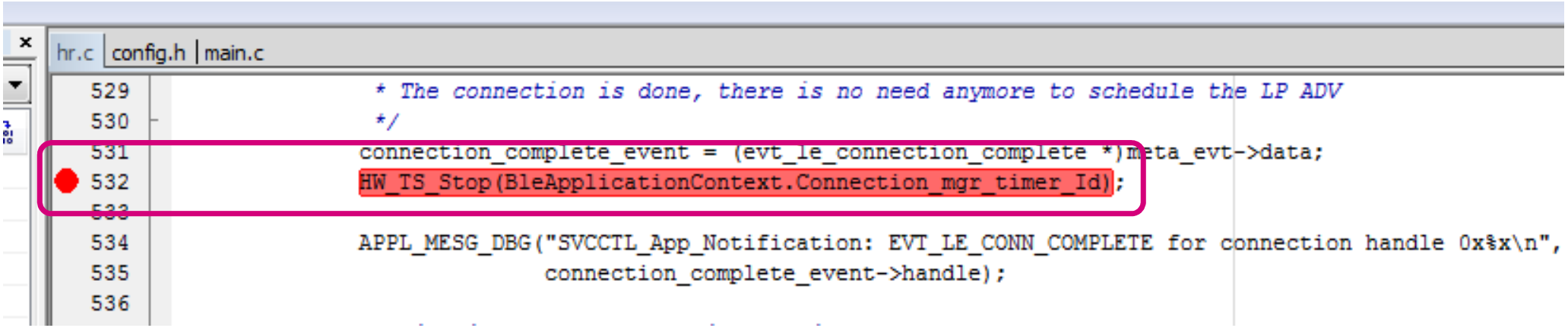
- You should see the simulated heart rate.

Debug the Firmware



- In IAR, double-click on the **hr.c** file
- Set a break point at line **496.**

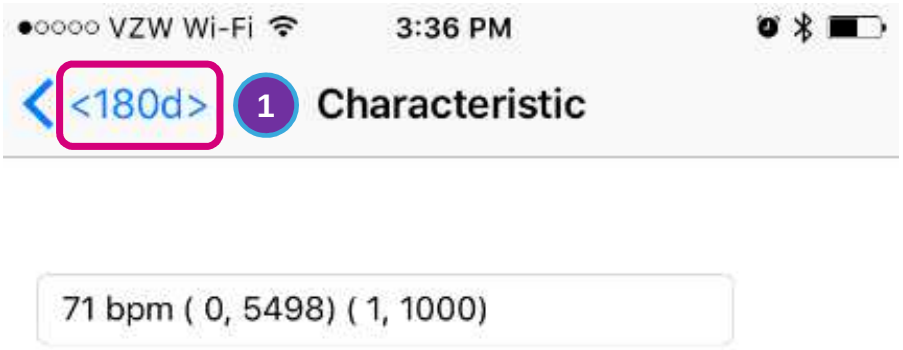
Debug the Firmware



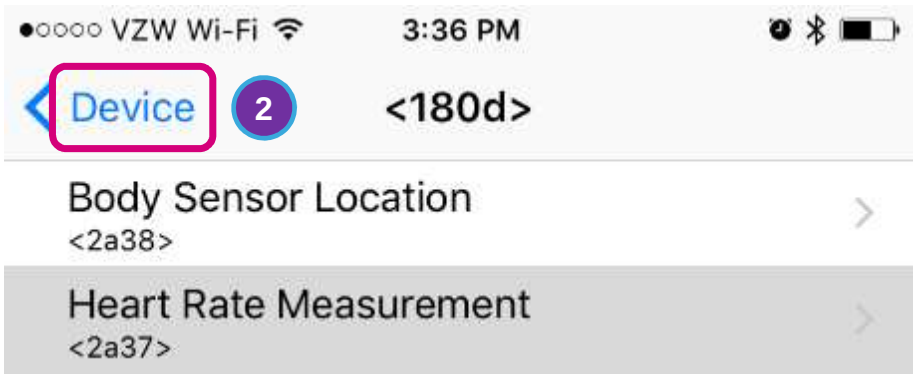
- Set another break point at line **532.**
- These break points will halt program execution when a client is connected & disconnected to the device.

Debug the Firmware (Disconnect)

iOS



iOS

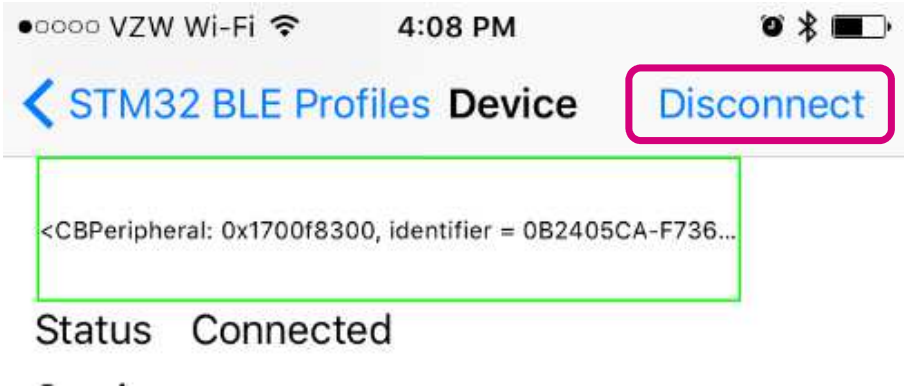


- iOS users, disconnect from your device by:
  1. Clicking on **<180d>**
  2. Clicking on **< Device**

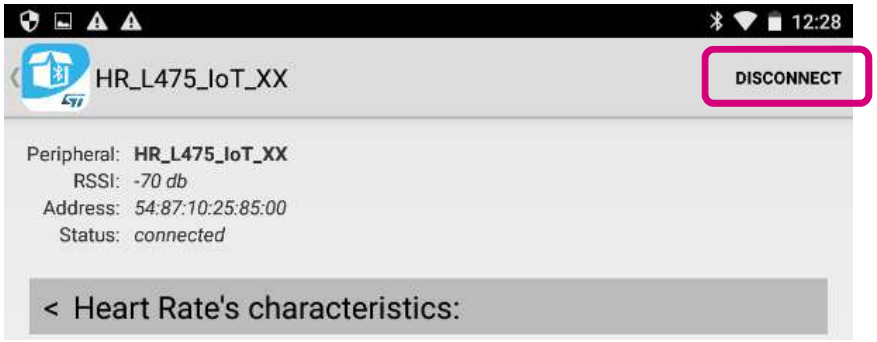
life.augmented

Debug the Firmware (Disconnect)

iOS

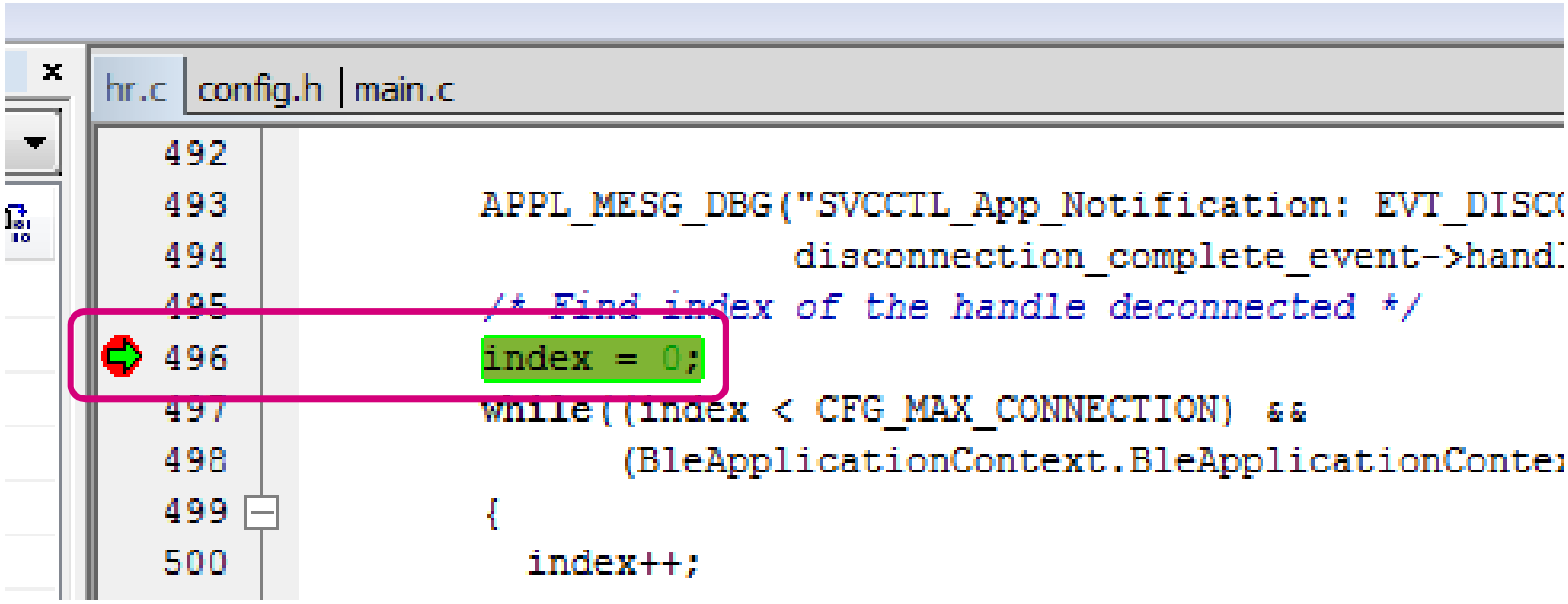


Android



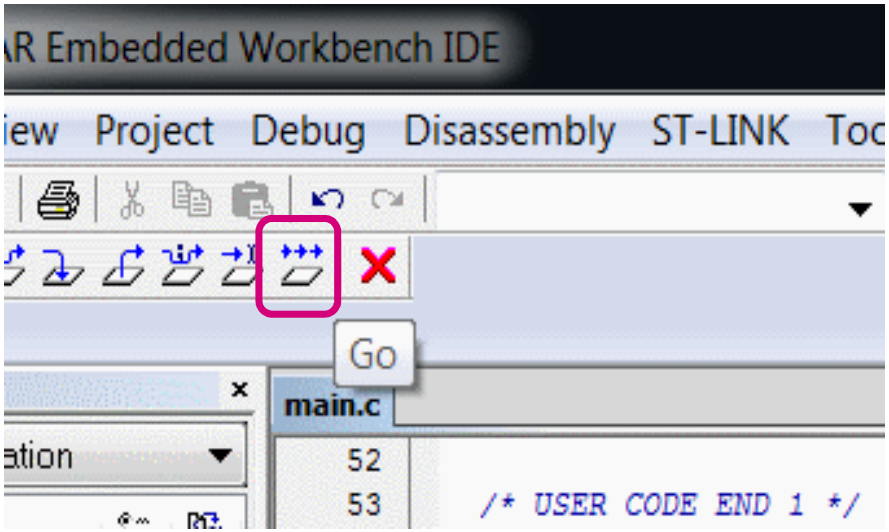
- Click **Disconnect** on your device

Debug the Firmware (Disconnect)



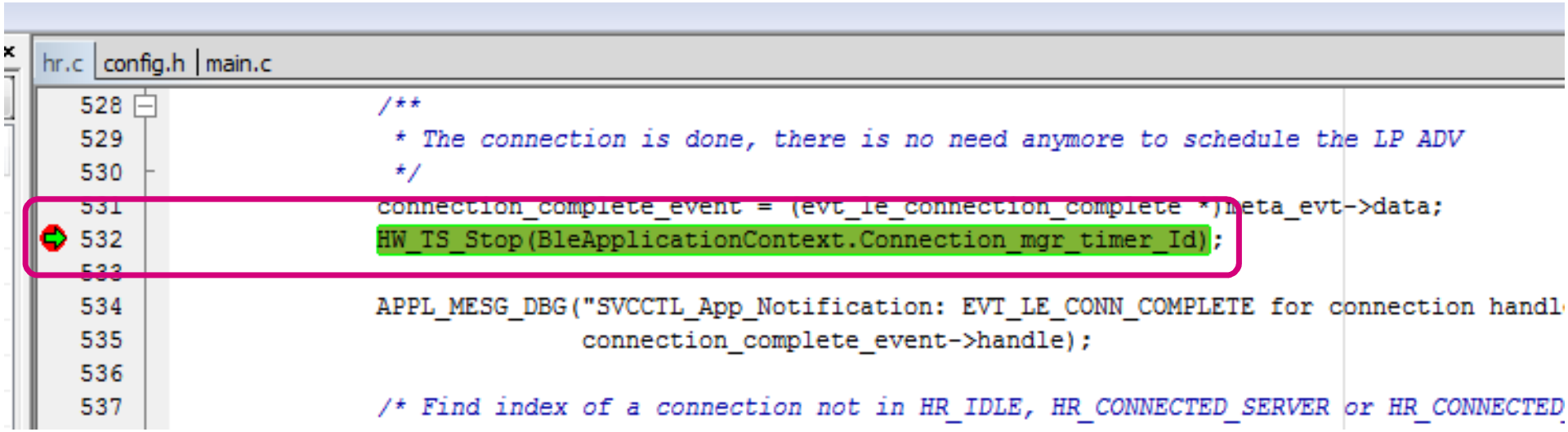
- In IAR, you will hit the previously set breakpoint at line **496**.

Debug the Firmware (Connect)

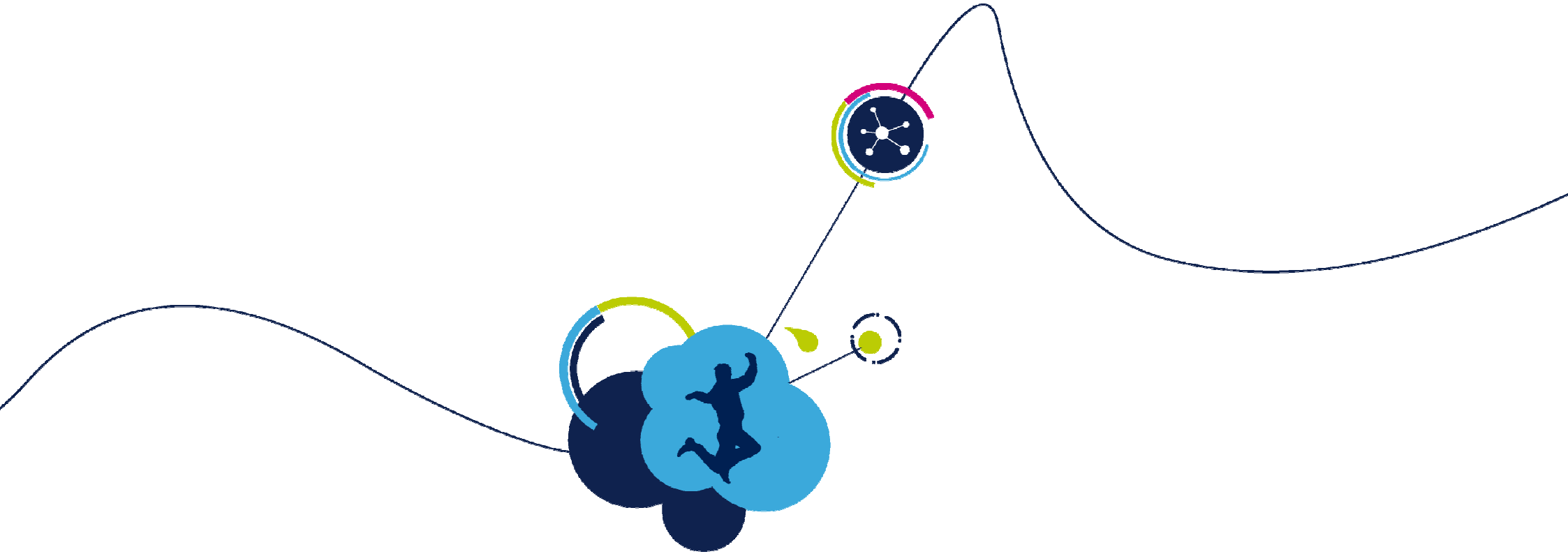


- Resume code execution by pressing the **Go** button (**F5**) on IAR.
- Re-connect to the device from your phone.

Debug the Firmware (Connect)



- In IAR, you will hit the previously set breakpoint at line **532**.



Lab 4 : Bluetooth® Low Energy P2P Communication

life.augmented

Lab Goal

- This lab will demonstrate Point To Point communication using Bluetooth® Low Energy.
- In this lab, I will work with a pair of kits. One board will run as a server, the second as a client.

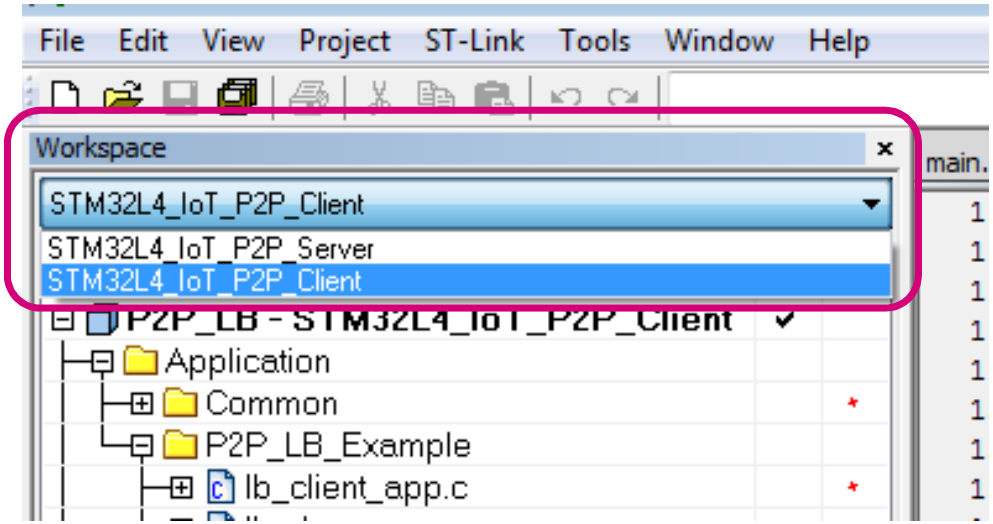
Open the P2P_LedButton Project

- We are going to configure the **P2P_LedButton** project with a unique BlueNRG MAC address for the client and server boards.
- We will also specify the remote MAC address on the client board.

1. Close the previous IAR project.
2. Double click on the **P2P_LB.eww** file located here:

C:\STM32L4_DK_IoT_Node_Seminar\Hands-on\Labs\Projects\B-L475E-IOT01\Applications\BLE\P2P_LedButton\EWARM

Select Your Configuration



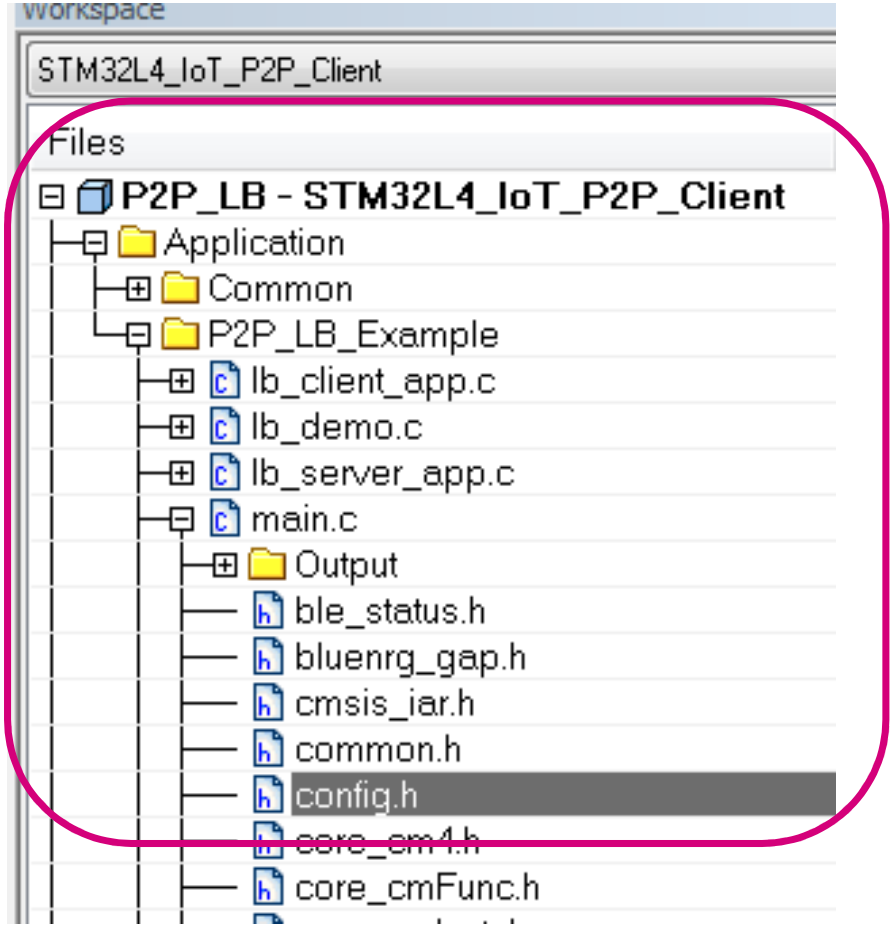
- From the Workspace project configuration drop list select your configuration
  - `STM32L4_IoT_P2P_Client` if you are running the client firmware
  - `STM32L4_IoT_P2P_Server` if you are running the server firmware

life.augmented

BlueNRG Module configuration (1/4)

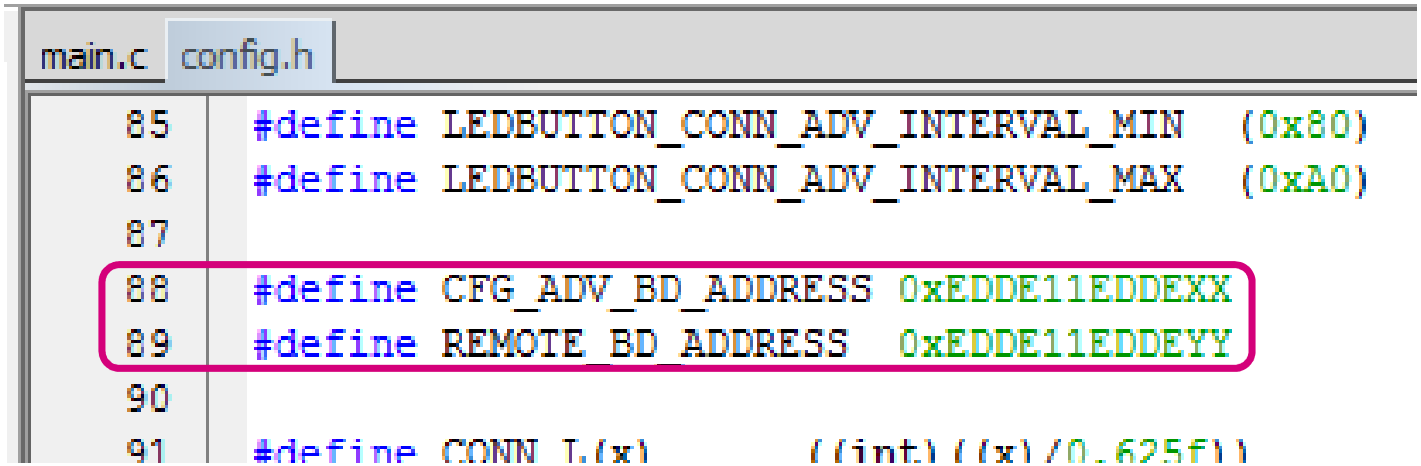
Expand the file tree:

1. Expand
   P2P_LB - STM32L4_IoT_P2P_Client
   or
   P2P_LB - STM32L4_IoT_P2P_Server
2. Expand **Application**
3. Expand **P2P_LB_Example**
4. Expand **main.c**
5. Double-click **config.h**



life.augmented

MAC Address Configuration (Client)



1. Replace the '**XX**' in the **CFG_ADV_BD_ADDRESS** (line 88) with any number (2 characters).
2. Replace the '**YY**' in the **REMOTE_BD_ADDRESS** (line 89) with any other different number (Server) (2 characters).

life.augmented

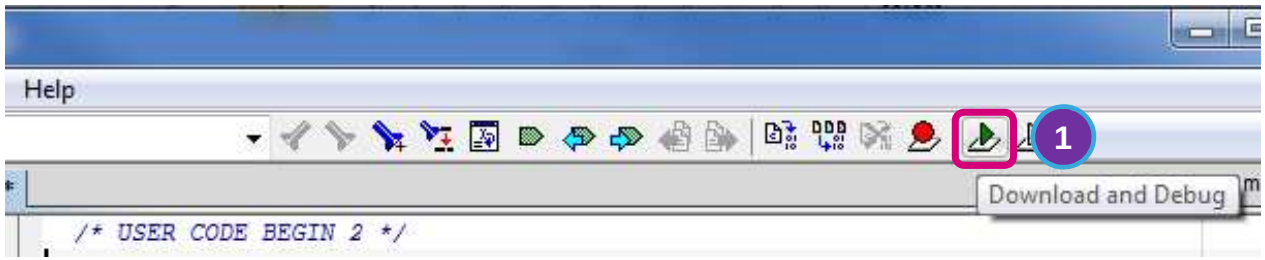
MAC Address Configuration (Server)


```
119 #if (LB_SERVER == 1)
120
121 #define LEDBUTTON_CONN_ADV_INTERVAL_MIN   (0x1F4)   /**< 312.5ms */
122 #define LEDBUTTON_CONN_ADV_INTERVAL_MAX   (0x640)   /**< 1000ms */
123
124
125 #define CFG_ADV_BD_ADDRESS 0xEDDE11EDDEXX
126
```

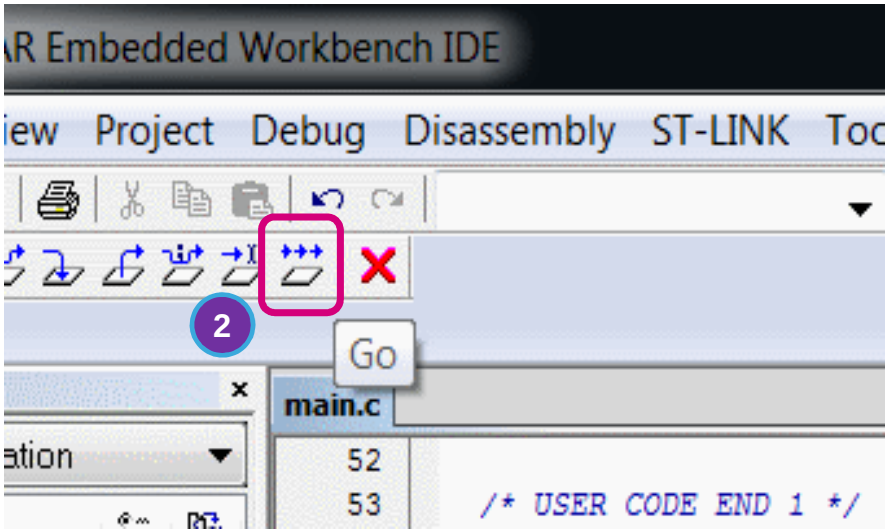
1. Replace the '**XX**' in the **CFG_ADV_BD_ADDRESS** (line 125) with the same previous XX number (2 characters).

Load and Run

1. Click the GREEN ARROW to Build the Project, **Download** and start the **debugger**. (Ctrl + D)



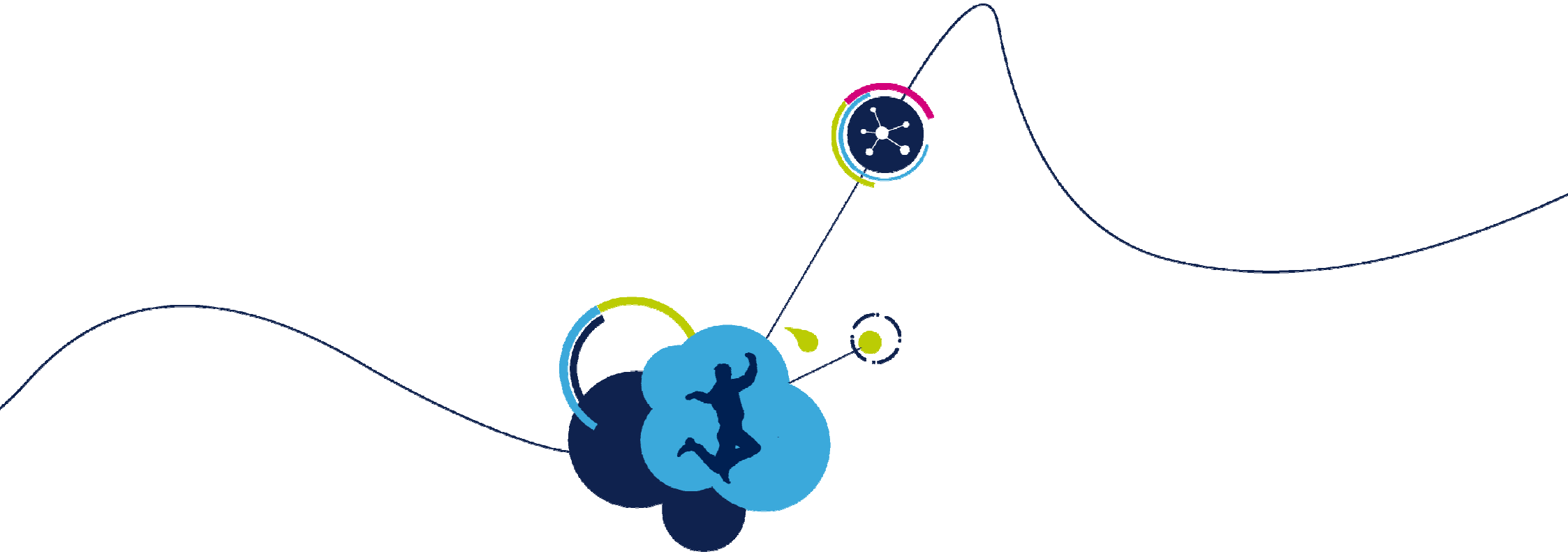
2. Click the triple-arrow **GO** button! (F5)



Run The Example

- After about 10 seconds the two boards will establish connection.
- Every time you press the blue button on your board, a message will get sent to the second board and toggle the LED.

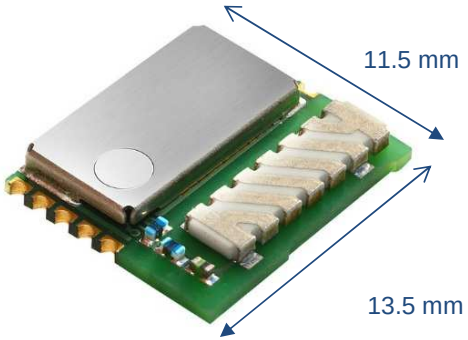
life.augmented

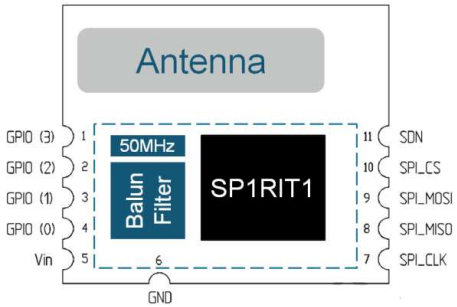


SPIRIT Sub-GHz Overview

life.augmented

SPSGRF-915 and 868 modules

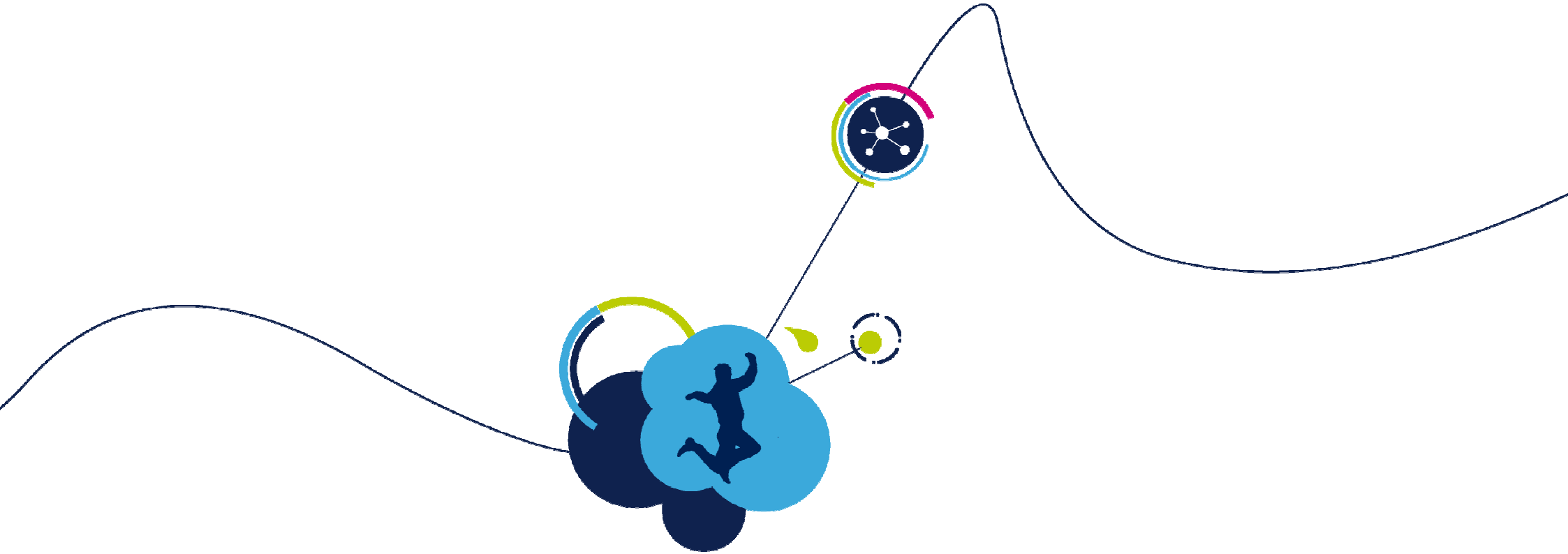




SPIRIT1 Modules for 868MHz and 915MHz

- Module based on:
  - **SPIRIT1** low power subGHz transceiver
  - BALF-SPI-01D3 balun and filter
  - Surface Mount antenna
- Tiny size: **13.5 x 11.5mm**
- 500Kbps data rate
- Temp. range from -40 °C to 85 °C
- Receiver sensitivity: **-118 dBm**
- Output power up to **+11.6 dBm**
- **RX: 9mA, Tx: 21mA** @ +11dBm
- **Shut Down: 2.5nA**
- **SPI** host interface
- **CE** compliant
- SPSGRF-915 is an FCC certified module (FCC ID: **S9NSPSGRF**)

life.augmented



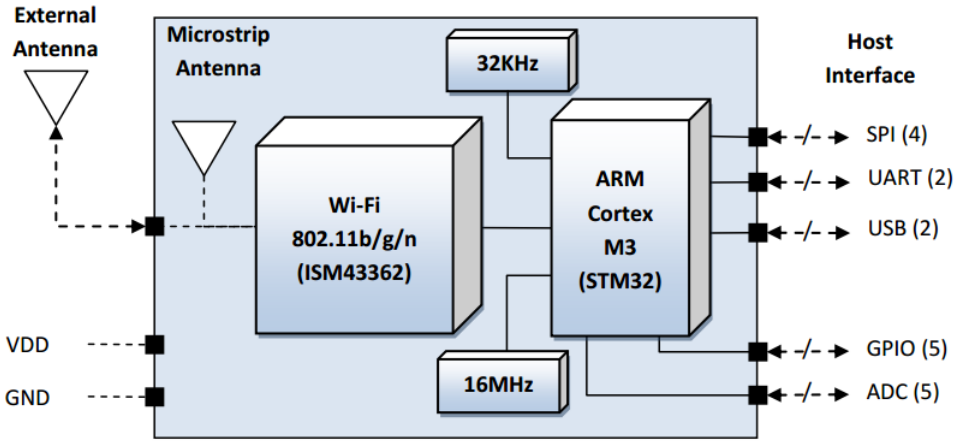
Wi-Fi Module Overview

life.augmented

ISM43362-M3G-L44-E/U

- The ISM43362-M3G-L44-E/U is an embedded 2.4 GHz Wi-Fi module from Inventek. The Wi-Fi module consists of a Broadcom BCM43362, an integrated antenna and an STM32F205 host processor with standard USB, SPI and UART interface capabilities.

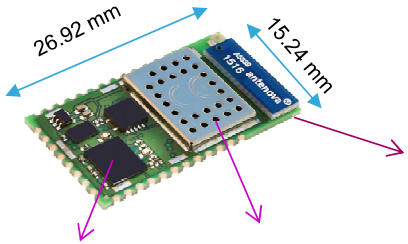
- The Wi-Fi module has an integrated TCP/IP stack and requires only simple AT commands to establish connectivity for your wireless product.



life.augmented

SPWF04S & X-NUCLEO-IDW04A1

Front

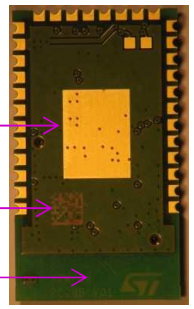


Antenna
(SPWF04SA)
or
U.Fl connector
(SPWF04SC)

STM32F439

Shielded 802.11 b/g/n
transceiver, 38MHz XTAL

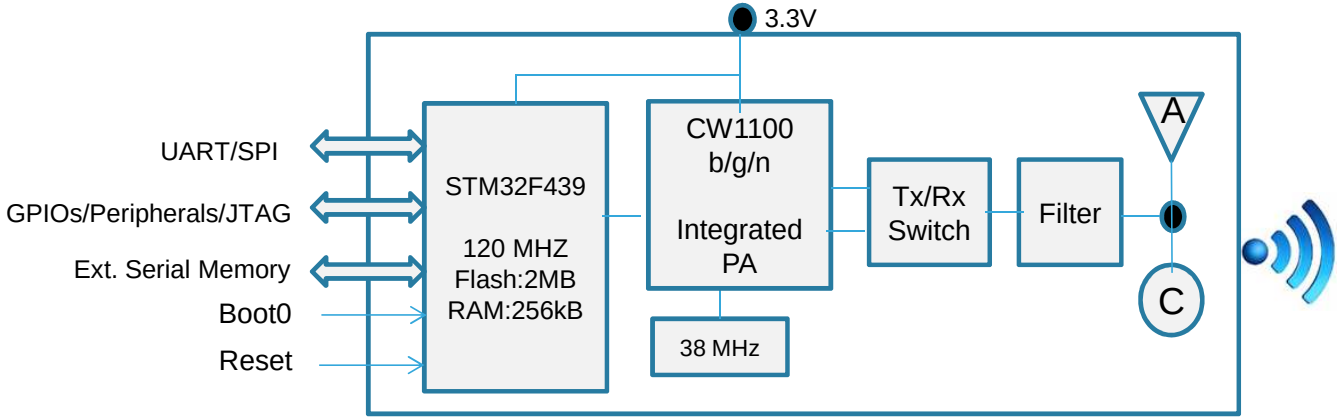
Bottom

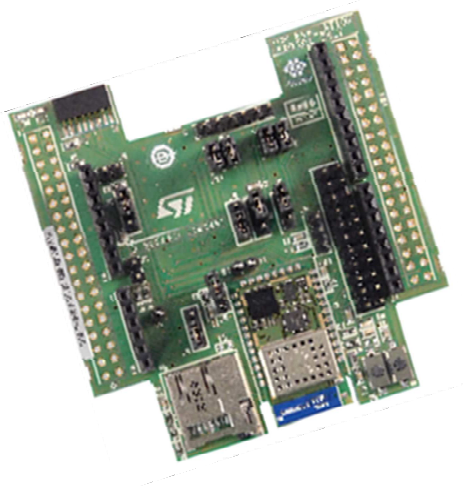


Ground Pad

Data Matrix

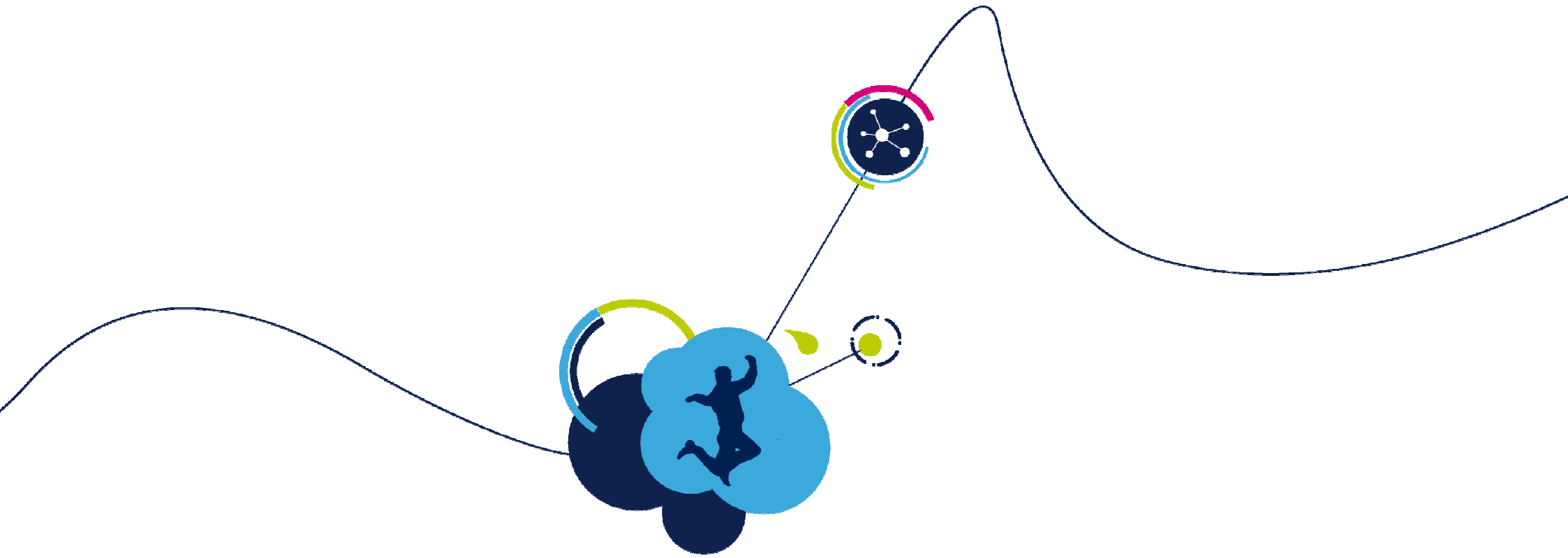
Model Series
FCC and IC reference





- ✓ **FW / FS upgrade possible through UART, JTAG, FOTA or SFOTA**
- ✓ **3.3V IO interface : no external level shifter needed**
- ✓ **Dedicated IOs function : UART / SPI / µPython selection, Wi-Fi indication**
- ✓ **SPWF01 and SPWF04 are HW compatible**

life.augmented



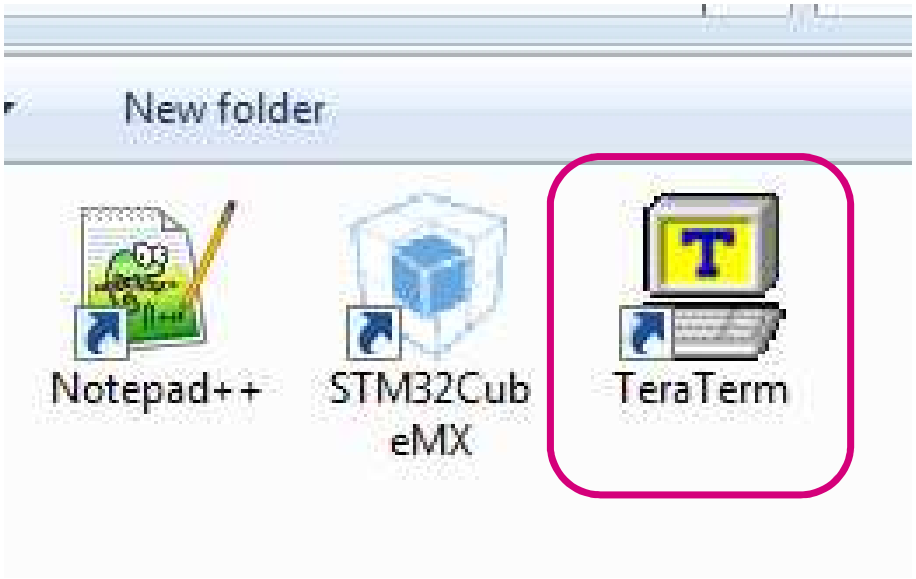
Lab 5 : Wi-Fi Client Server

life.augmented

Lab Goal

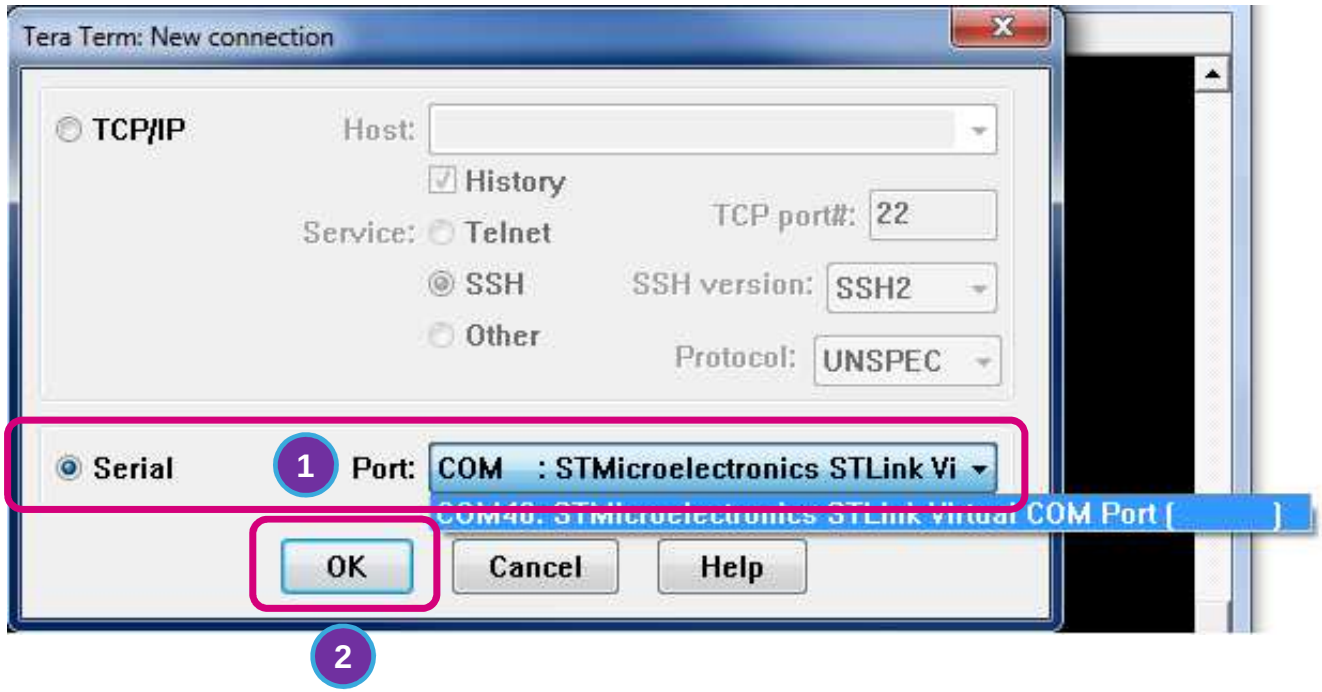
- In this example we are going to demonstrate how to:
  - Connect to a Wi-Fi access point
  - Become the server and receive messages from a client (Cellphone)
- Enjoy sending messages

Open TeraTerm



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

Tera Term Configuration 1/5

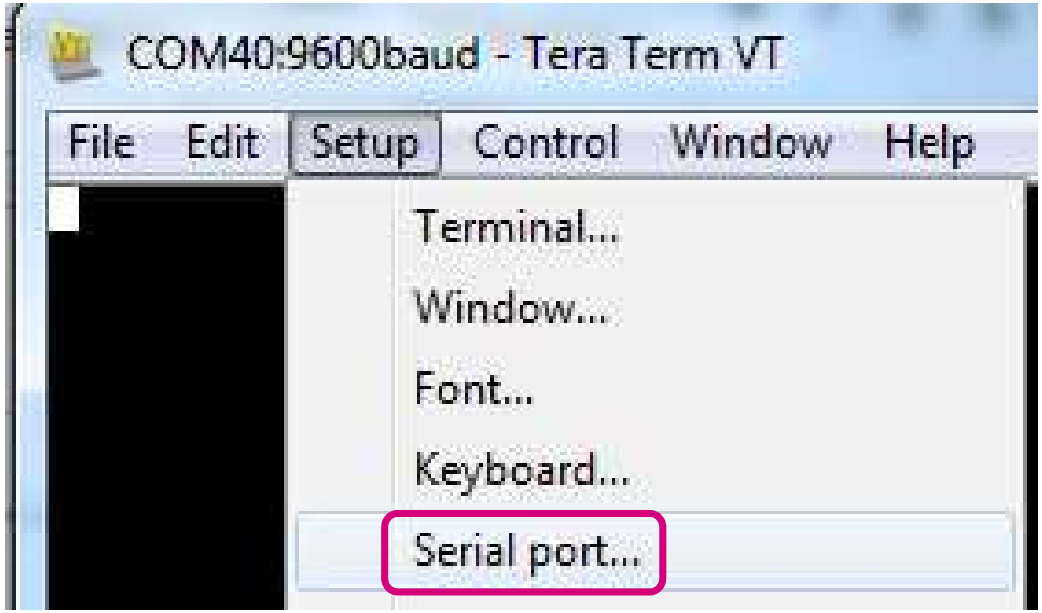


1. Select the **STMicroelectronics STLink Virtual COM Port**
2. Click **OK**

life.augmented

Tera Term Configuration 2/5

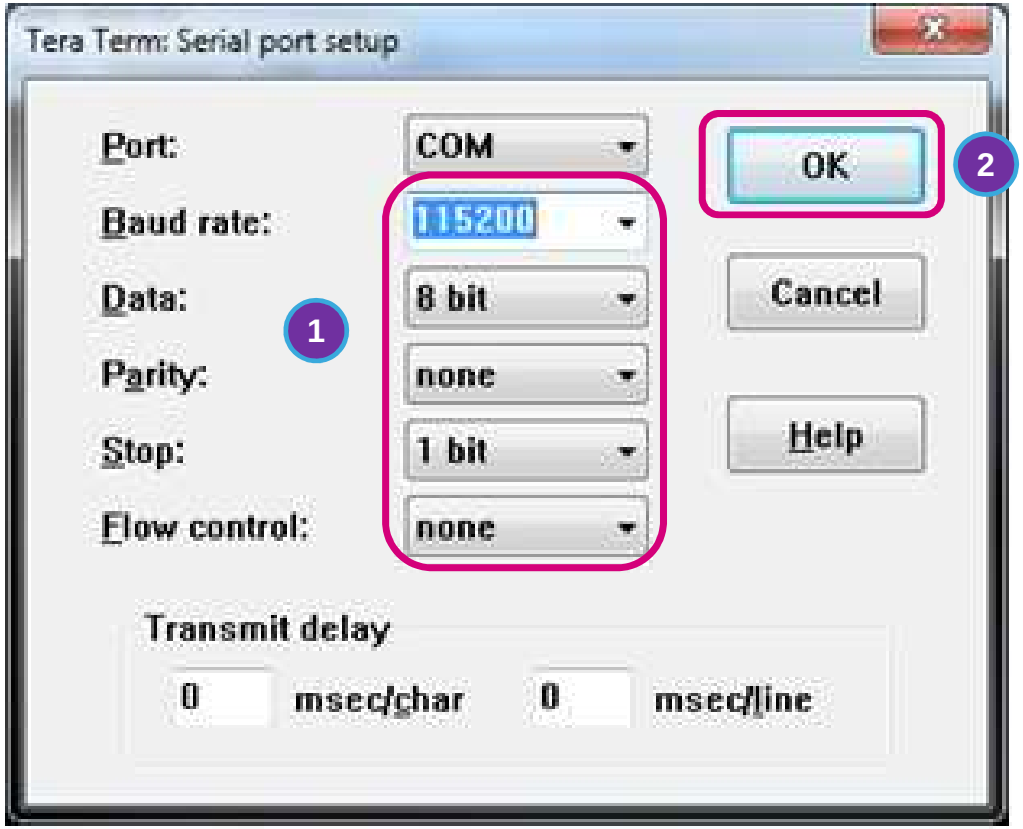
1. Click **Setup** -> **Serial port…**



Tera Term Configuration 3/5

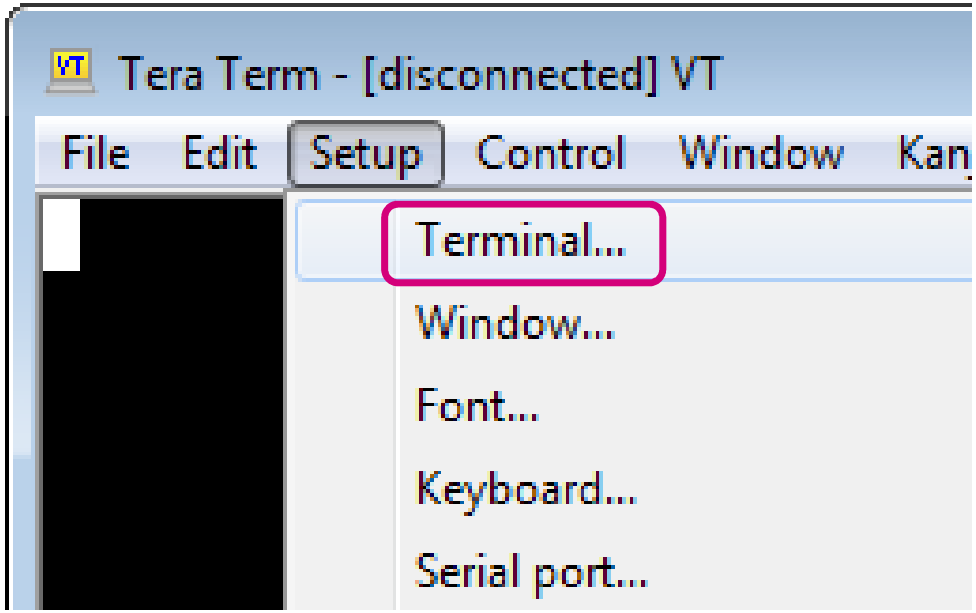
1. Set the following:
   - Baud rate : **115200**
   - Data : **8 bit**
   - Parity : **none**
   - Stop : **1 bit**
   - Flow control : **none**

2. Click **OK**



Tera Term Configuration 4/5

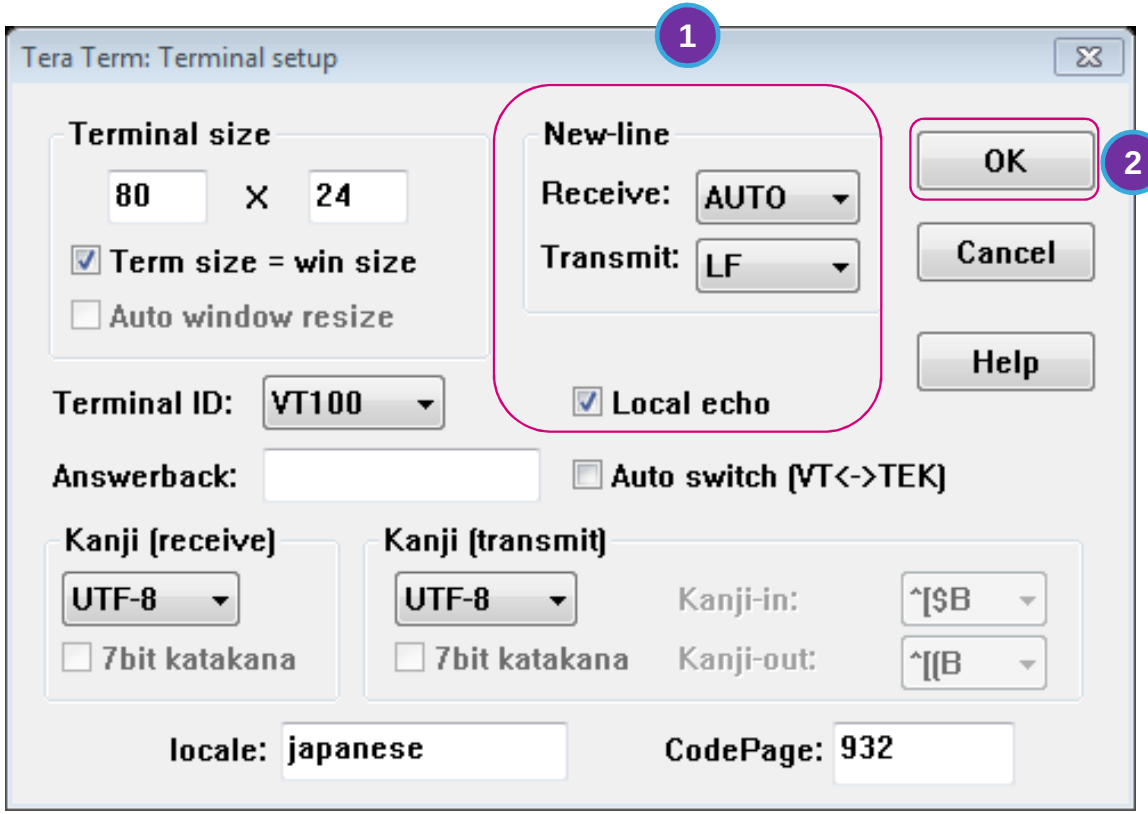
1. Click **Setup** -> **Terminal…**



Tera Term Configuration 5/5

1. Set the following:
   - New-Line Receive : **AUTO**
   - New-Line Transmit : **LF**
   - Enable **Local echo**

2. Click **OK**



Open the Wi-Fi Project

- We are going to configure the **`WiFi`** program to view debug information and to view data coming from a server.

1. Close the previous IAR project.
2. Double click on the **`Project.eww`** file located here:

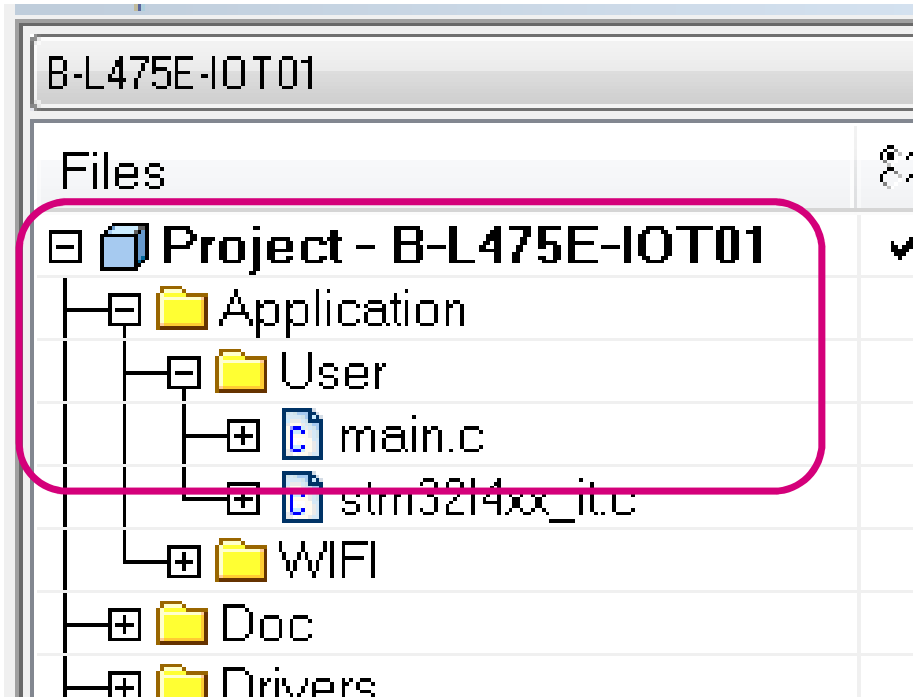
`C:\STM32L4_DK_IoT_Node_Seminar\STM32CubeExpansion_WIFI1_V3.0.2\Projects\Multi\Applications\Server_Socket\EWARM\B-L475E-IOT01`

life.augmented

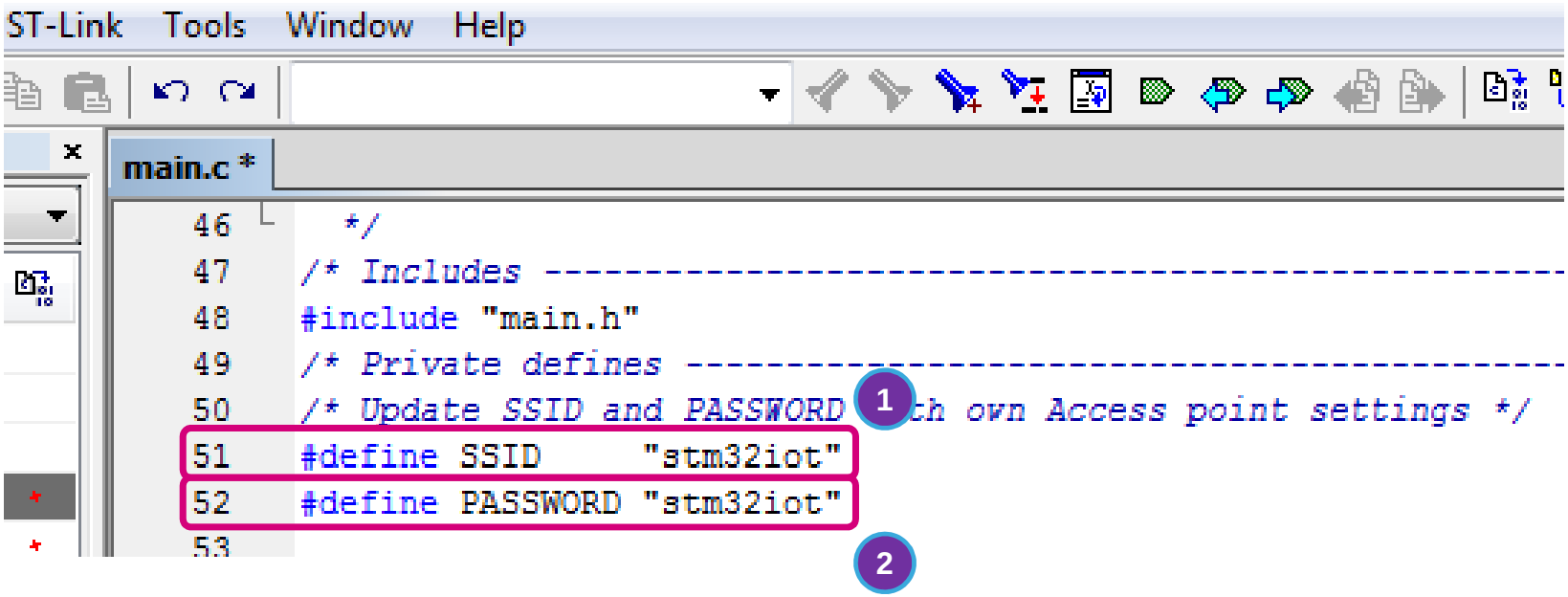
Setting Wi-Fi Credentials

Expand the file tree:

1. Expand **Project – B-L475E-IOT1**
2. Expand **Application**
3. Expand **User**
4. Double-click **main.c**



Setting Wi-Fi Credentials



1. Set the Wi-Fi SSID credential on line **51** to **stm32iot**
2. Set the Wi-Fi password credential on line **52** to **stm32iot**

Setting the Server Address

main.c *

```
61  #if defined (TERMINAL_USE)
62  extern UART_HandleTypeDef hDiscoUart;
63  #endif /* TERMINAL_USE */
64  uint8_t RemoteIP[] = {aaa,bbb,ccc,ddd};
65  uint8_t RxData [500];
66  char* modulename;
67  uint8_t TxData[] = "STM32 : Hello!\n";
```

- On line **64**, set the server IP address to the presenters' host IP address.

Setting the Server Address

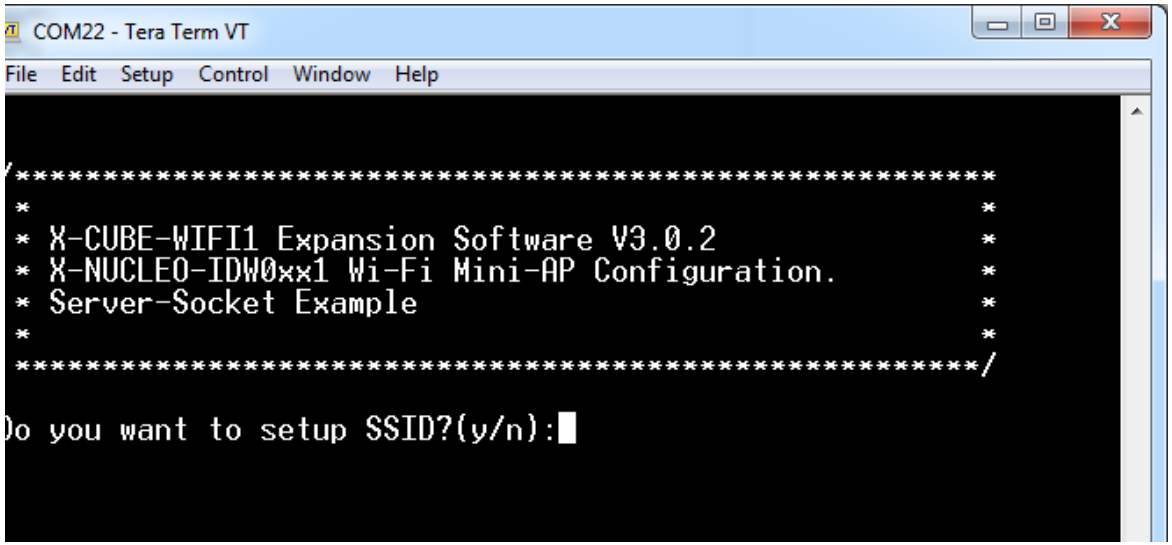
The IP address is:

192.168.0.50

life.augmented

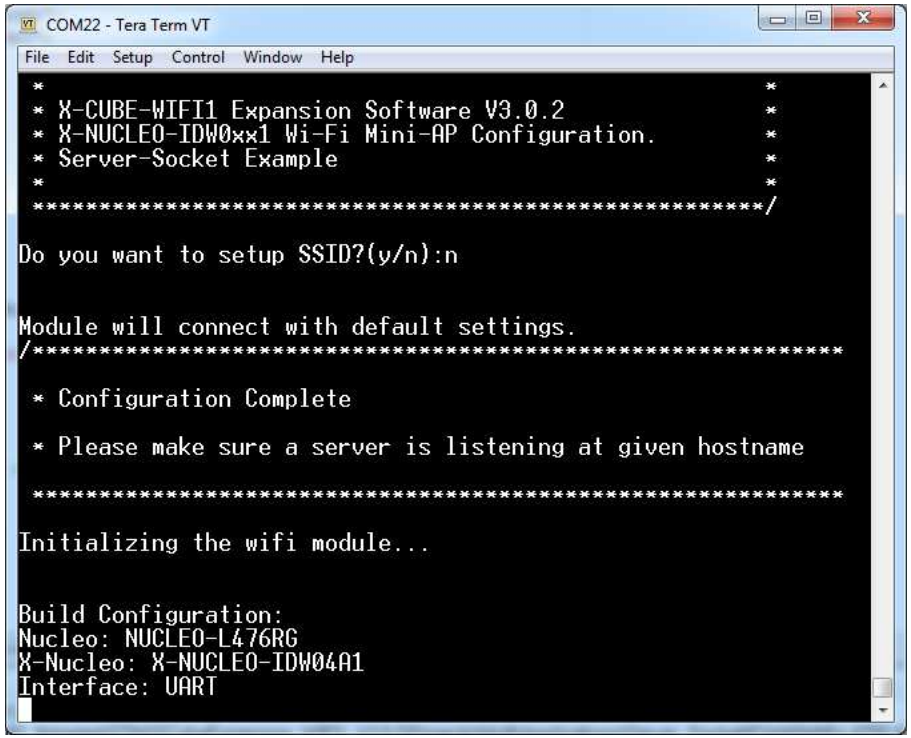
Output

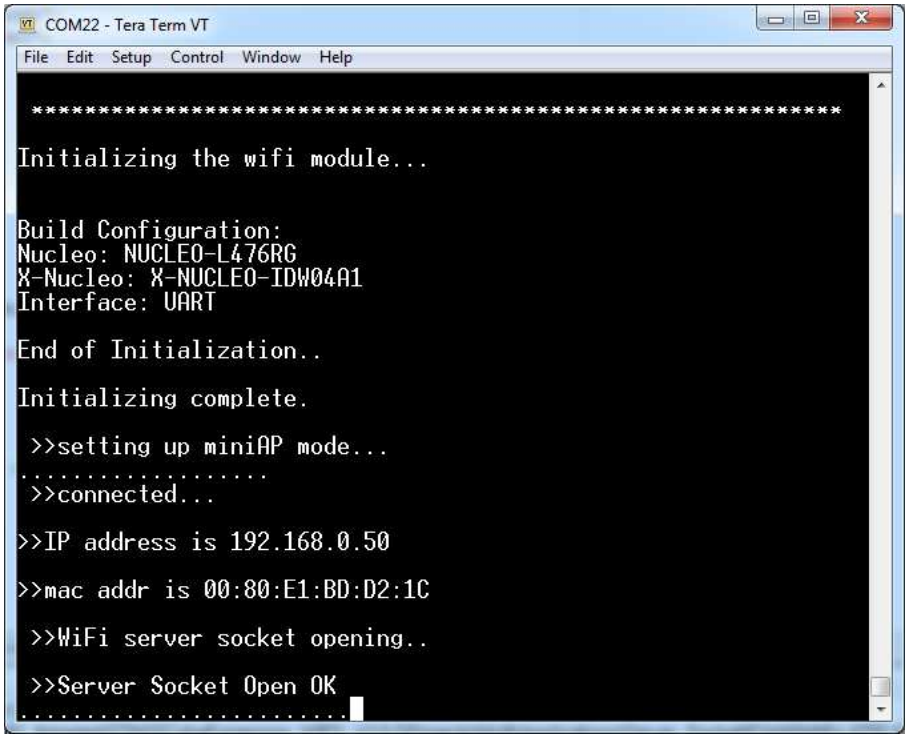
- Don’t need to configure the SSID, as it’s already set in our code



Output

- It'll start the configuration of the MiniAP and Open the socket at the given IP

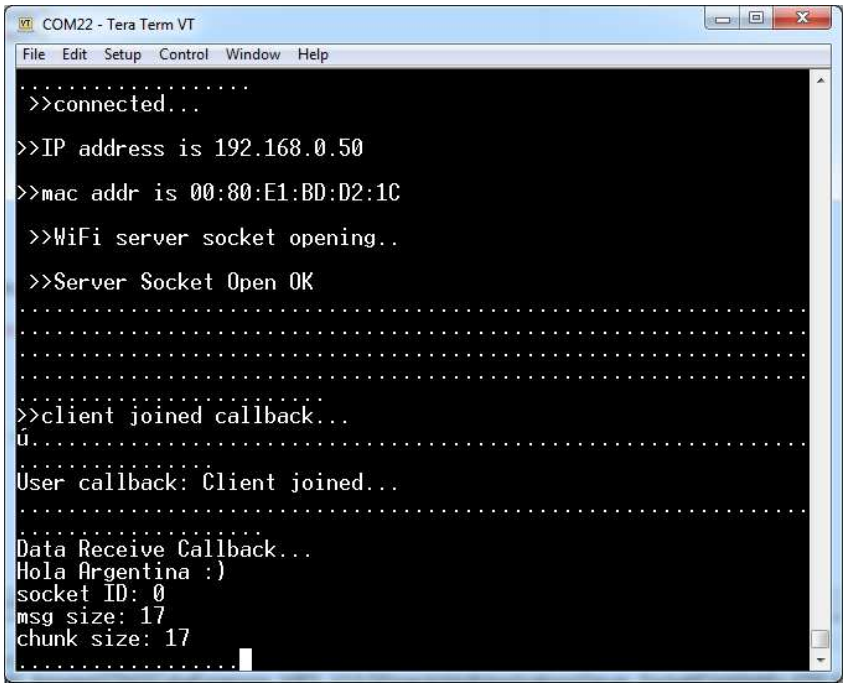




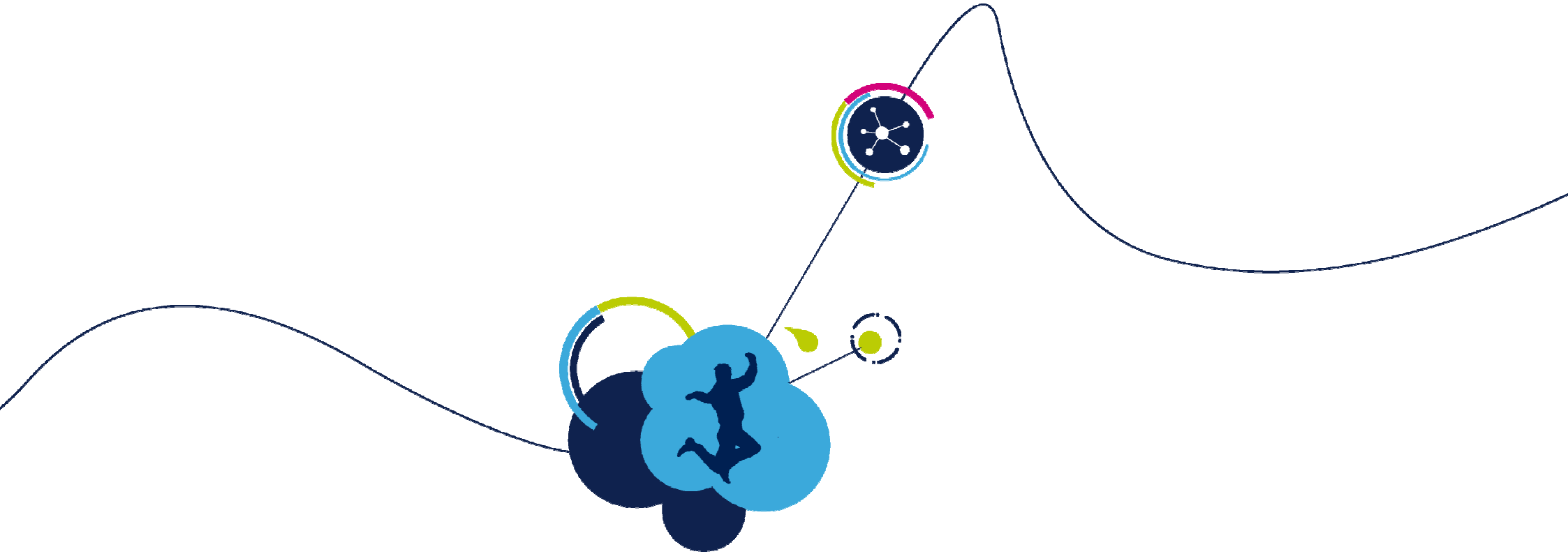
life.augmented

Output

- Using a simple App in my phone, I can enter the IP and connect as a client and send messages:



life.augmented



LoRa Overview

life.augmented

What is LoRa™ ?

1. A Sub-GHz wireless technology enabling low data rate communication over long distances

2. Targeting M2M and Internet of Things, IoT applications

3. LoRa™ technology is a solution providing a WAN capability, using a MAC protocol named LoRaWAN

Long range

- Greater than cellular
- Deep indoor coverage
- Star topology

Max lifetime

- Low power optimized
- **10 to 20-year** lifetime
- >10x vs cellular M2M

Multi-usage

- High capacity
- Multi-tenant
- Public network

Low cost

- Minimal infrastructure
- Low-cost end-node
- Open software

True location

- Indoor and outdoor
- Accurate

Bidirectional

- Bidirectional
- Scalable capacity
- Broadcast

Global mobility

- True mobility
- Seamless
- Roaming

Security

- Unique ID
- Application
- Network

life.augmented

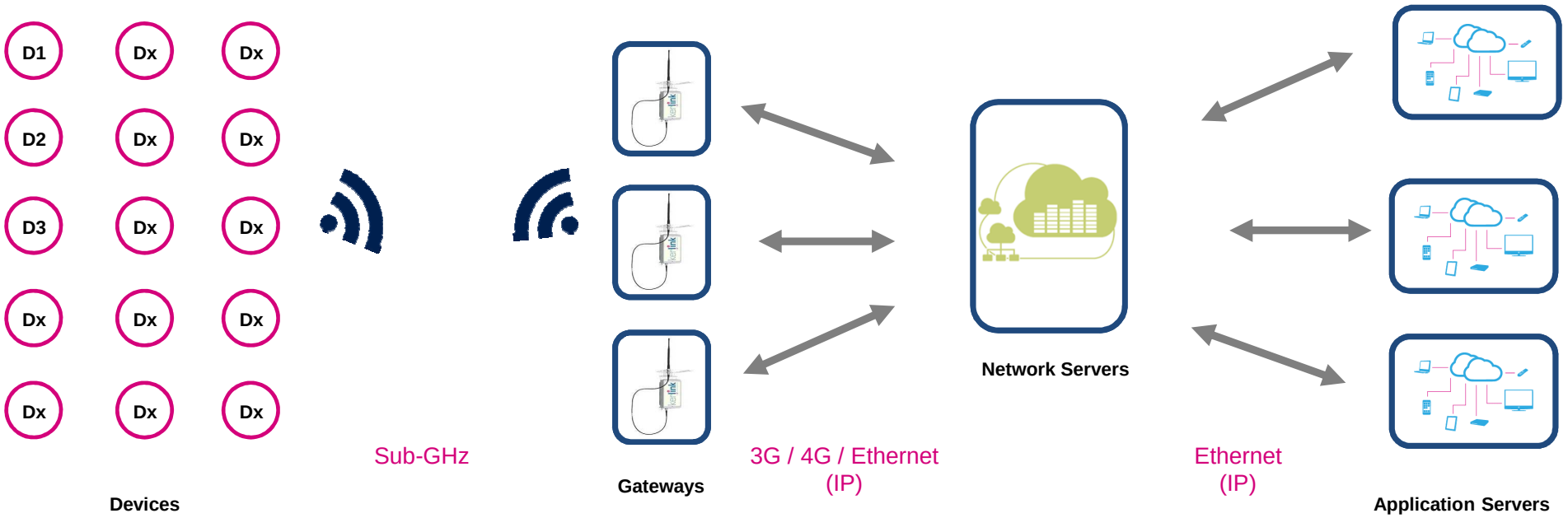
LoRaWAN™ Devices Classes

3 classes to cover all use cases

Class name	Intended usage	
A (“all”)	**Battery powered sensors** (or actuators with no latency constraint) Most energy efficient communication class. Must be supported by all devices.	***Mainly uplink with two potential downlink slots after each uplink***
B (“beacon”)	**Battery powered actuators** Energy efficient communication class for latency controlled downlink. Based on slotted communication synchronized with a network beacon.	***Programmed downlink slots to allow control within certain latency limits***
C (“continuous”)	**Main powered actuators** Devices witch can afford to listen continuously. No latency for downlink communication.	***Lowest latency command and control for less power critical devices***

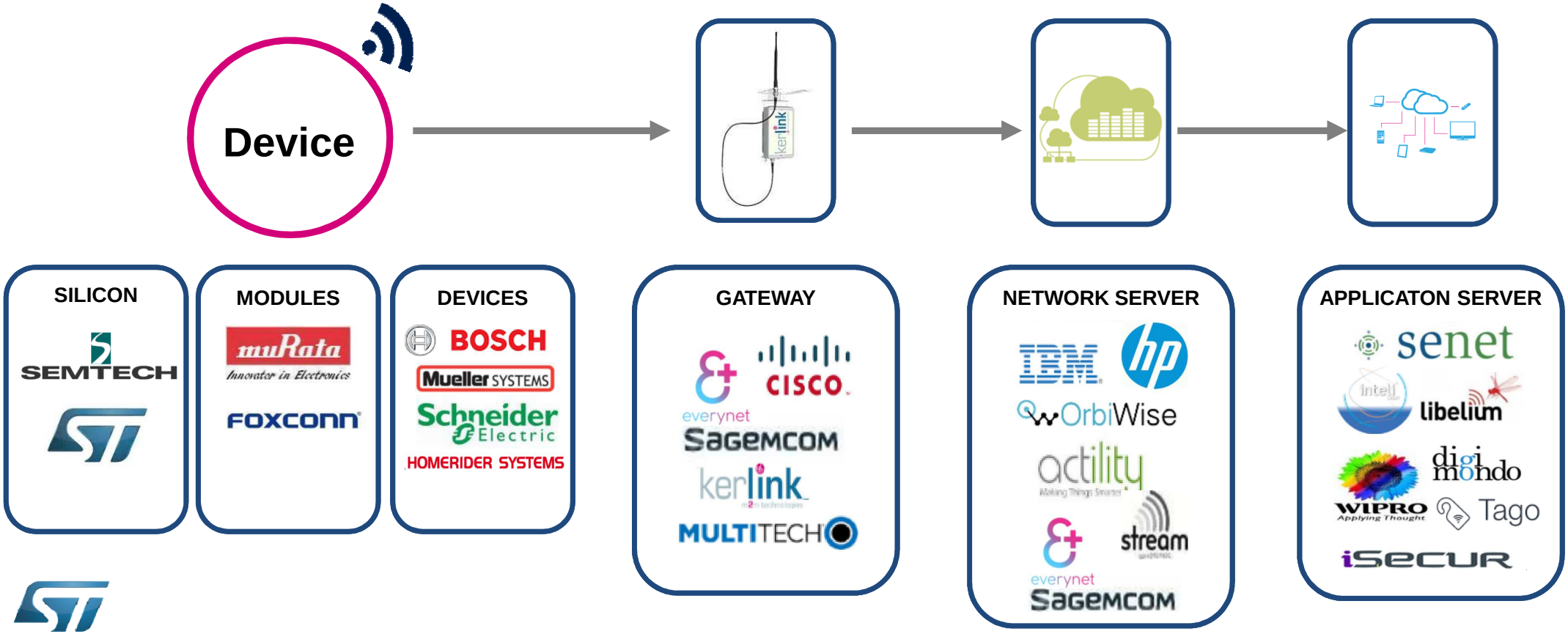
LoRa™ Network Protocol

Network Topology Overview



LoRa™ Network Protocol

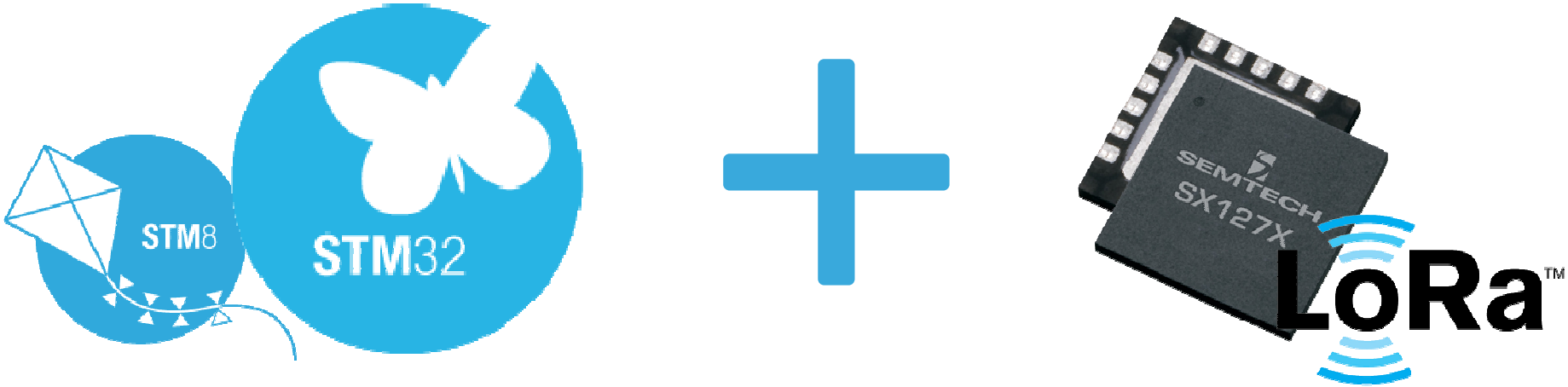
Solutions providers



life.augmented

The Perfect Match: STM32/STM8 + LoRa™

More than 4000 possible combinations



STM32/STM8 are available in more than **1000 references**
LoRa SX127x offers **4 different** lines

life.augmented

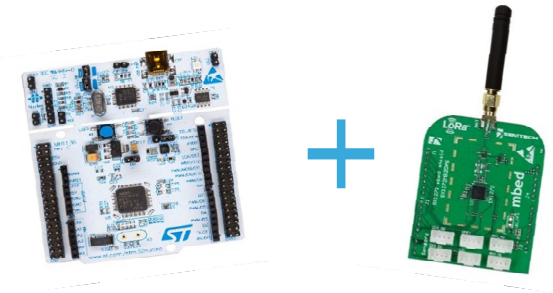
SEMTECH

Let's Get Started

With a wide and existing ecosystem

(Click on the icon or link)

HW tools



P-NUCLEO-LRWAN1
B-L072-LRWAN1

life.augmented | SEMTECH

Dev tools

STM32CubeMX

ST-Link Utility

Partners IDE

KEIL Tools by ARM

ARM mbed enabled

System Workbench for STM32

IAR SYSTEMS

LoRaWAN™ stack

life.augmented

GitHub

IBM

ARM mbed

+ demo code examples

LoRaWan – Joining Process

2 activation methods exist to join an existing network

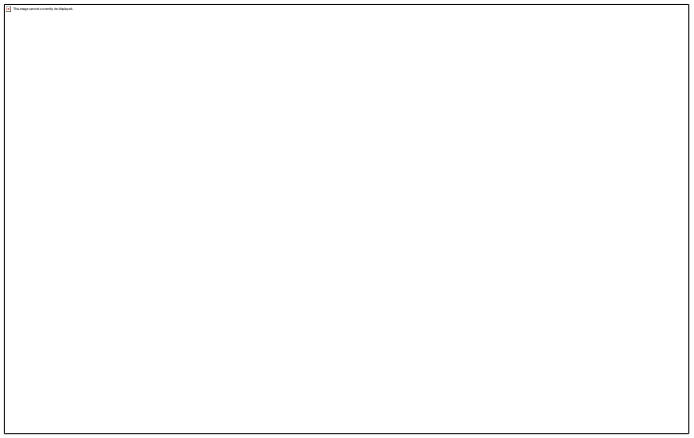
1. OTTA: Over the Air Activation = Over the air handshaking
   - End-device send a Join Request (J.S) message to A.S including DevEUI, AppEUI, AppKey
     - DevEUI (Global unique end-device identifier) will allow the network to identify the device
     - AppEUI to identify which A.S the end-device is talking to
     - AppKey to identify which application on the A.S the end-device is referring to
   - End device receives the Join Accept from the A.S and proceed with
     - Join Accept (authenticates and decrypt the Join Accept)
     - Extract and store DevAddr (the 32-bit device address)
     - Derives the 2 security: Network session Key (NwkSKey) and Application Session Key (AppSKey)

2. ABP: Activation By Personalization (no Over the air handshaking)
   - The Device ready out-of-the-box to talk on the network
   - The DevAddr, NwkSKey and AppSKey are programed at factory level (no Over the air handshaking)

life.augmented

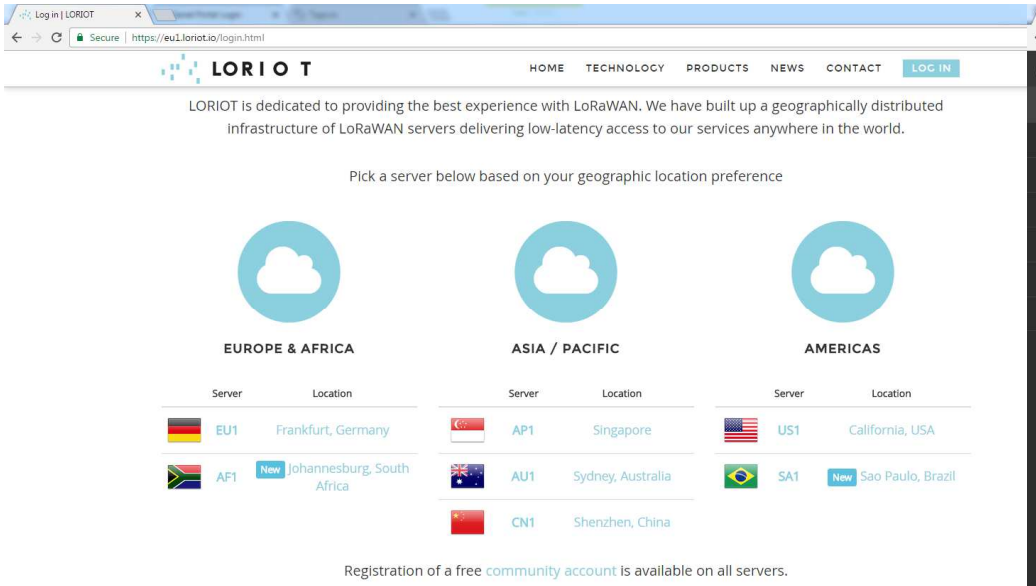
Setting up the Demo - Gateway

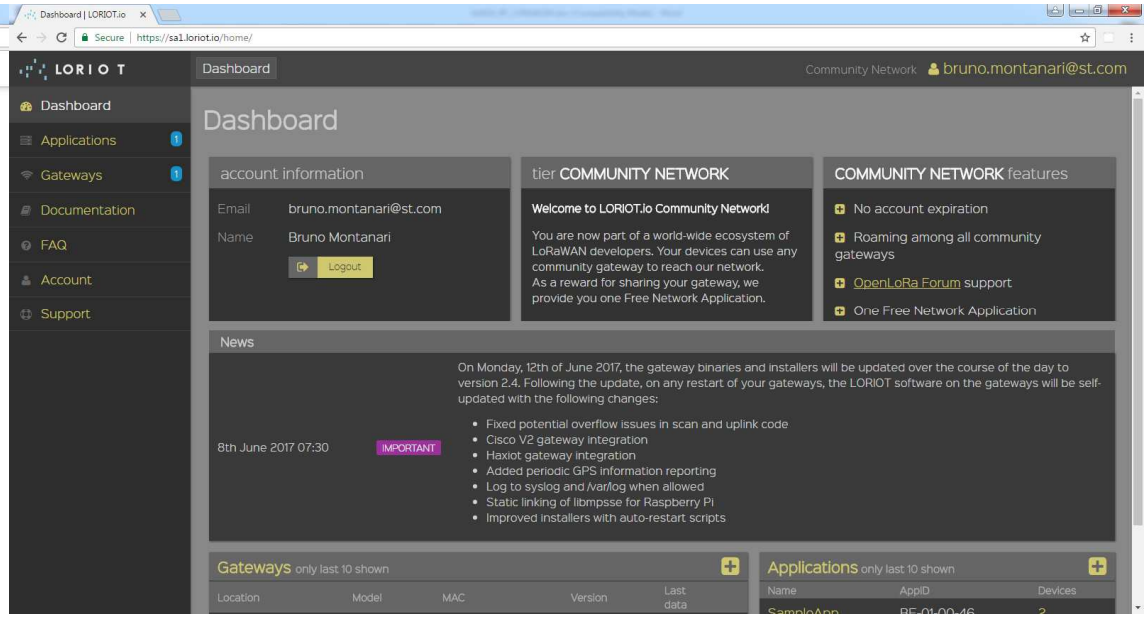
- Multitech - Gateway
  - It is a LoRa Gateway
  - Connects to LoRa Network Server through 3G-LTE, 3G, 2G and Ethernet
  - Optimized for use in the 868 MHz **and** 915 MHz
  - Ideal to demonstrate LoRa where no network available around
  - http://www.multitech.net/developer/products/multiconnect-conduit-platform

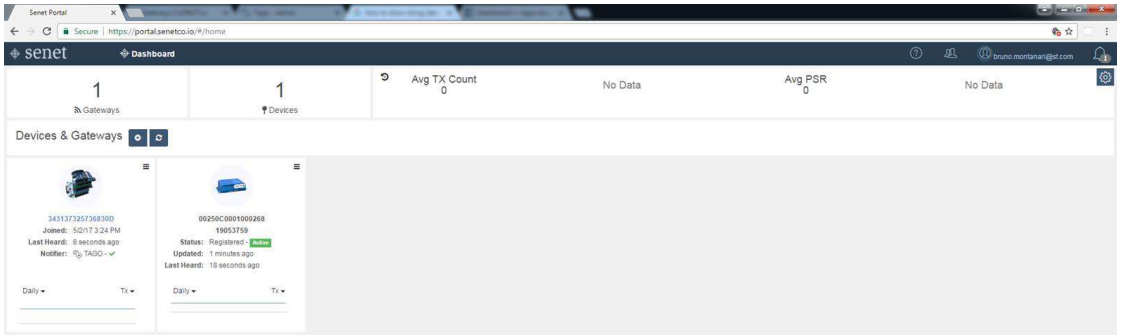


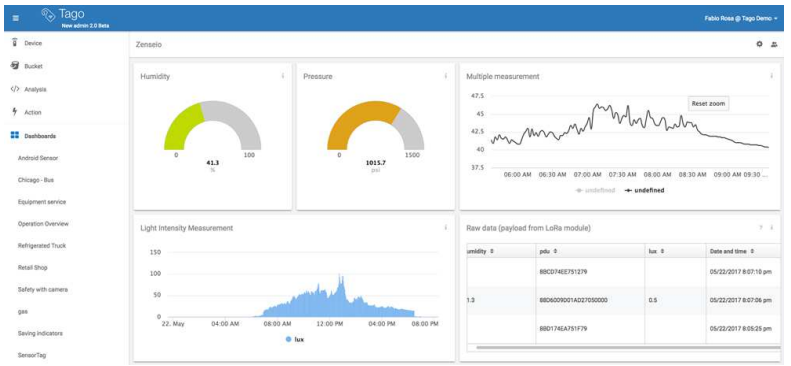
life.augmented

Setting up the Demo – Network Server









life.augmented

Setting up the Demo – FW and HW

- Firmware Side
  - Download the STM32CubeExpansion_LRWAN package
  - Edit the LoraTxData function to send the information you want, in this case, we'll transmitt:

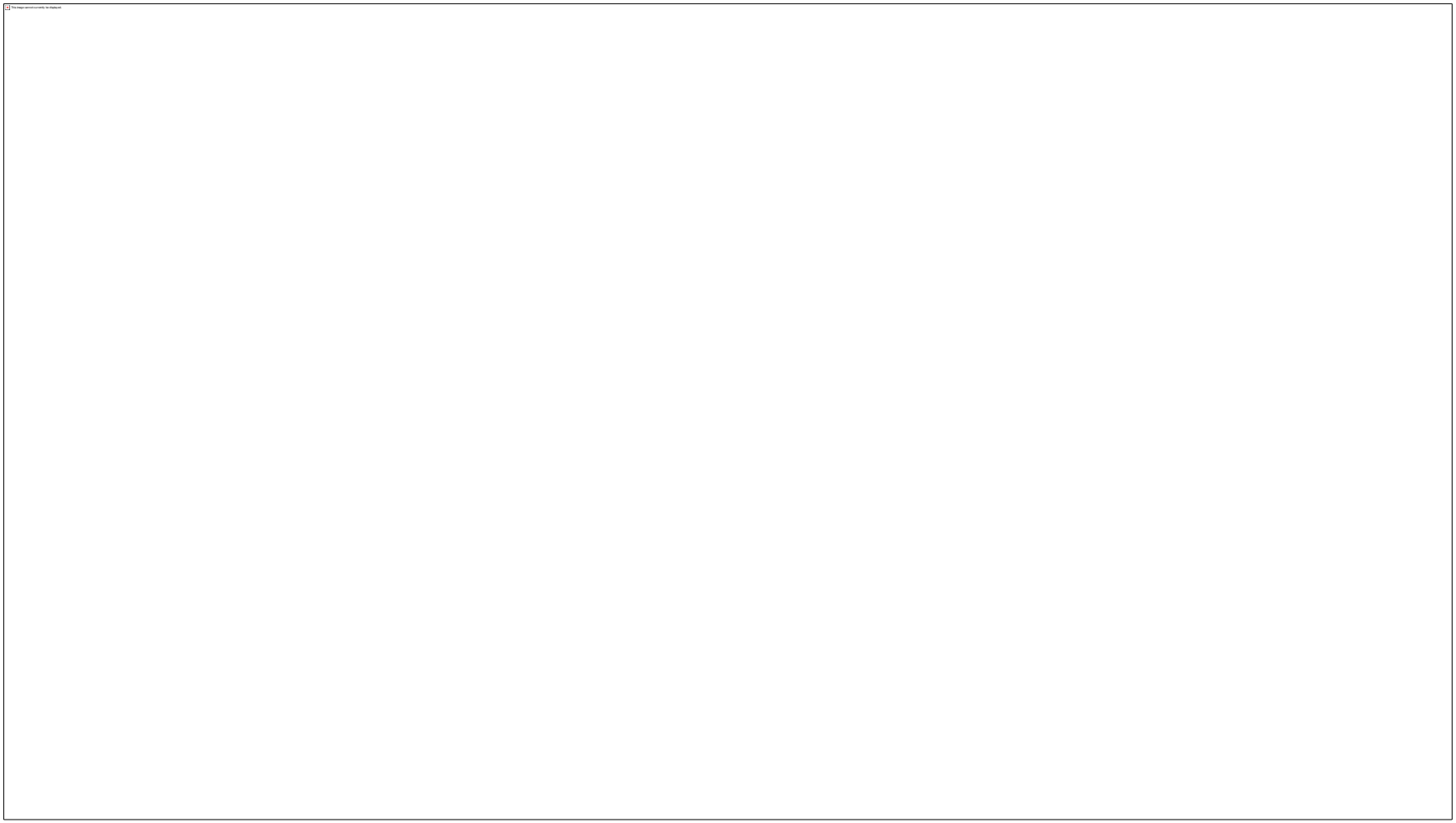
```
sprintf((char *)AppData->Buff,"STM32");
```

  - Select the joining mode and the device information.
  - This info can be seen in the serial port once compiled and donwloaded:

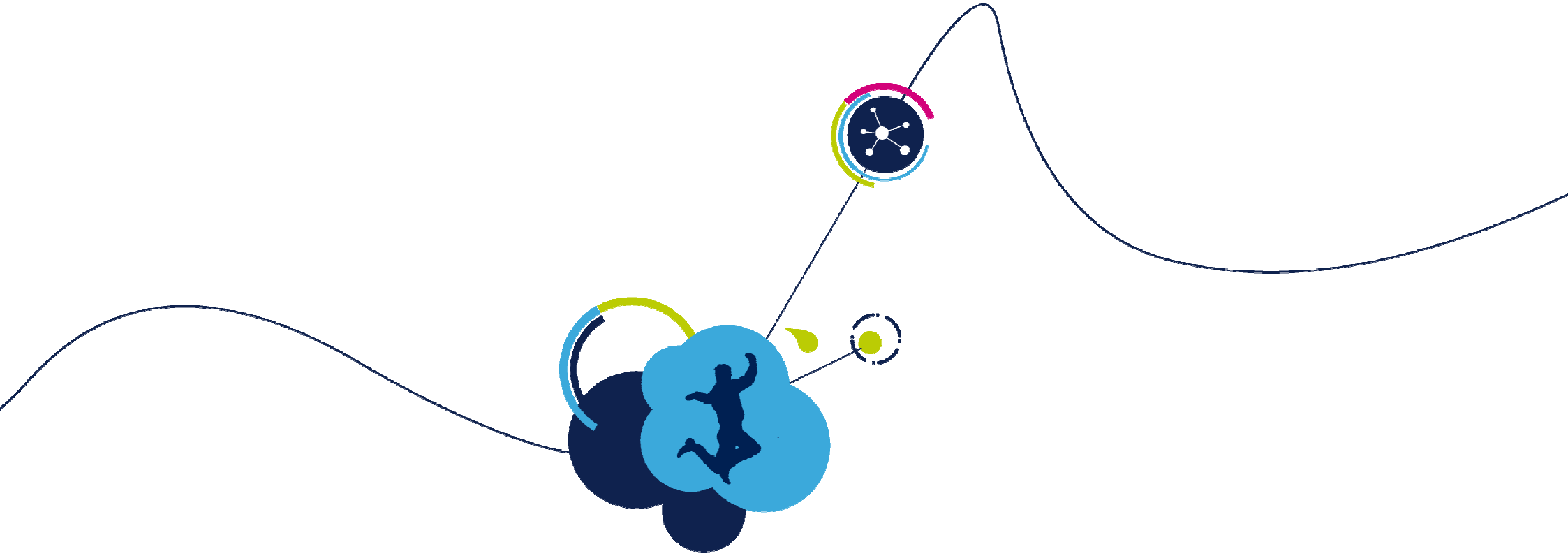
Hardware Side

B-L072Z-LRWAN1

The Dash Board+JavaScript – Loriot



life.augmented



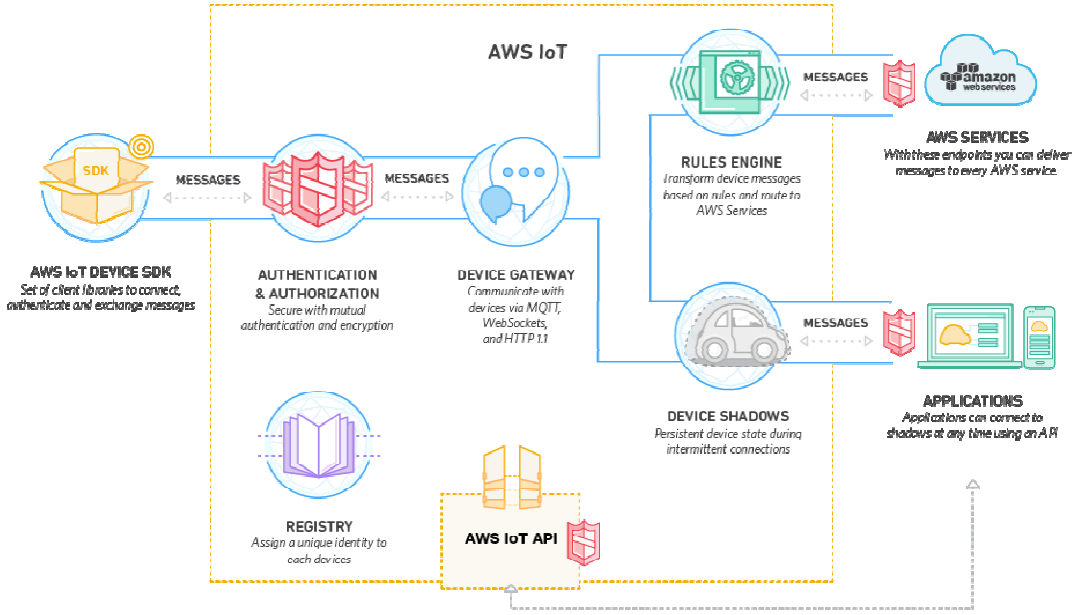
Amazon AWS IoT Overview

life.augmented

What is Amazon AWS IoT?

- The Amazon AWS IoT service enables secure, bidirectional communication between IoT devices and the cloud over MQTT, HTTP and WebSockets.

- AWS IoT devices are authenticated using TLS mutual authentication with X.509 certificates. Once a certificate is provisioned and activated it can be installed on a device. The device will then use that certificate for all requests to AWS MQTT.



life.augmented

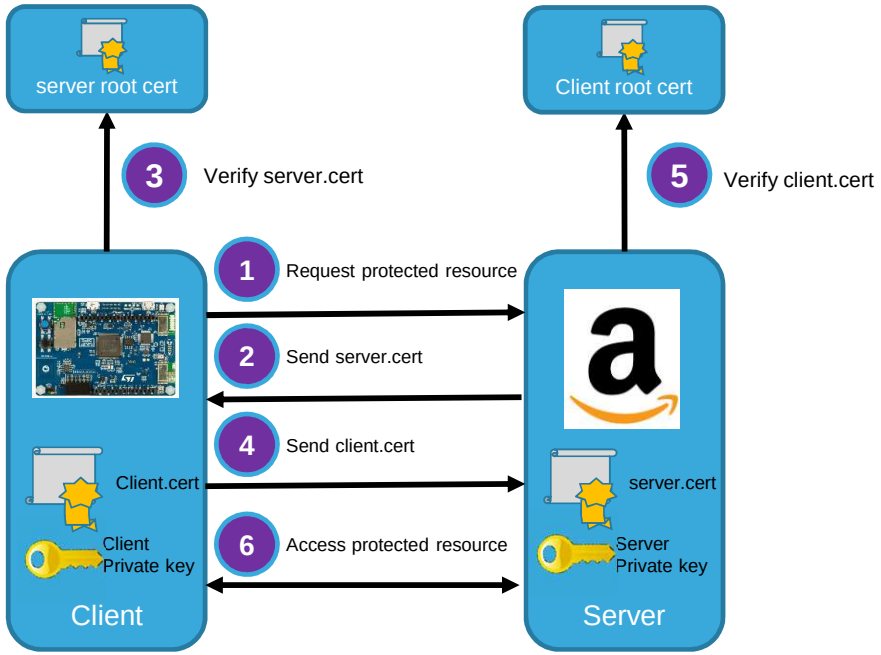
What is TLS Mutual Authentication?

- TLS mutual authentication is used to establish trust between two parties.
- Each party verifies the certificate provided by the other.
- Certificate Authorities (CA) like Verisign are an important part of the mutual authentication.

life.augmented

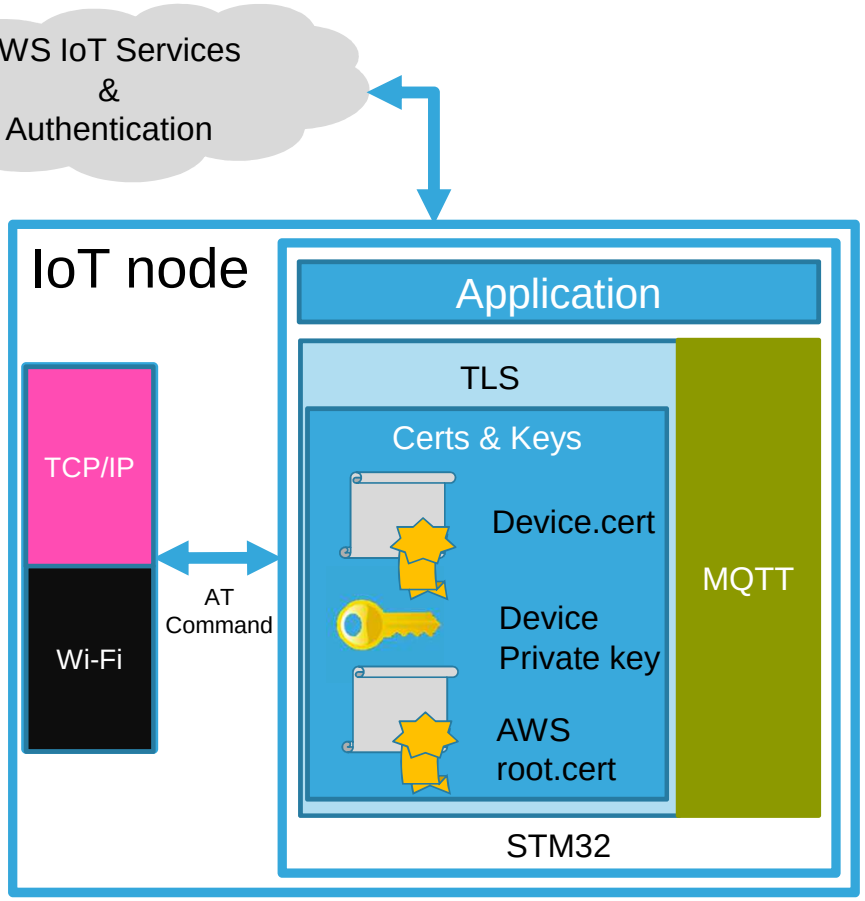
TLS Mutual Authentication Flow

- TLS mutual authentication follows these steps:
  1. A client request access to a protected resource
  2. The server presents its certificate to the client
  3. The client verifies the server's certificate
  4. The client sends it's certificate to the server
  5. The server verifies the client's certificate
  6. The server gives access to the protected resource requested by the client



AWS Security Overview

- AWS provides the device certificate and keys (AWS is the CA for the IoT device).
- The certificates and keys should be installed on the device. The device will then use that certificate and key to authenticate itself and send all requests to AWS IoT.
- To perform AWS IoT operations with your device, you must create an AWS IoT policy and attach it to your device certificate.



life.augmented

AWS IoT Policies

- AWS IoT policies are JSON* documents that authorize your device for performing AWS IoT operations.
- AWS IoT defines a set of policy actions describing the operations and resources for which you can grant or deny access. For example:
  - iot:Connect represents permission to connect to the AWS IoT message broker.
  - iot:Subscribe represents permission to subscribe to an MQTT topic or topic filter.
  - iot:GetThingShadow represents permission to get a thing shadow.

* JavaScript Object Notation

AWS IoT Thing Shadow

- A Thing Shadow (sometimes referred to as a Device Shadow) is a JSON document that is used to store and retrieve current state information for a Thing (device, app and so on).

- The Thing Shadow service maintains a persistent state of the device regardless of whether the device is connected to the Internet or not.

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.

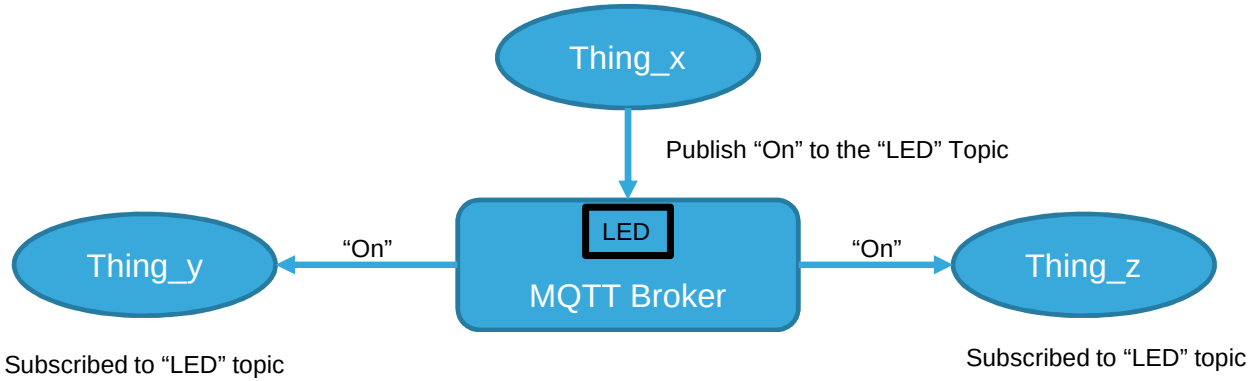
- Designers use **MQTT** to minimize network bandwidth and device resource requirements, while ensuring reliability and some degree of delivery assurance. These principles also turn out to make the protocol ideal in the emerging “machine-to-machine” (M2M) or “Internet of Things” world of connected devices,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

Source: http://mqtt.org/

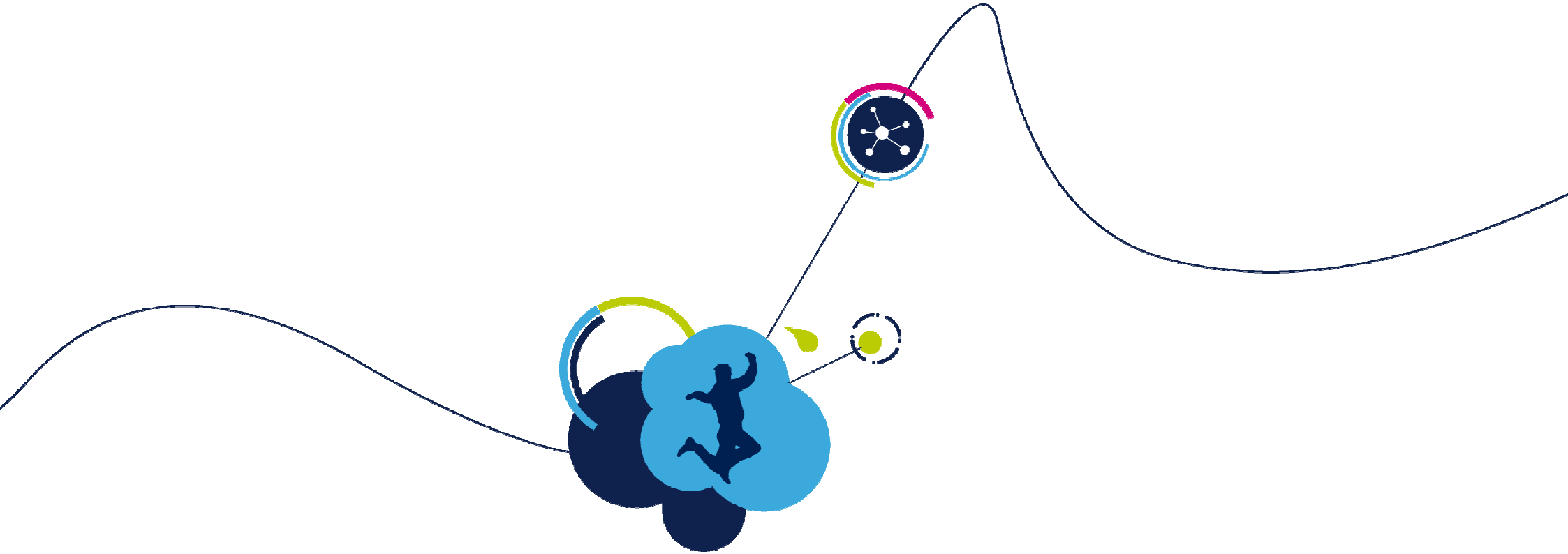
How MQTT Works

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

- 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.



life.augmented



Lab 6 : Creating your Device (“Thing”) on AWS

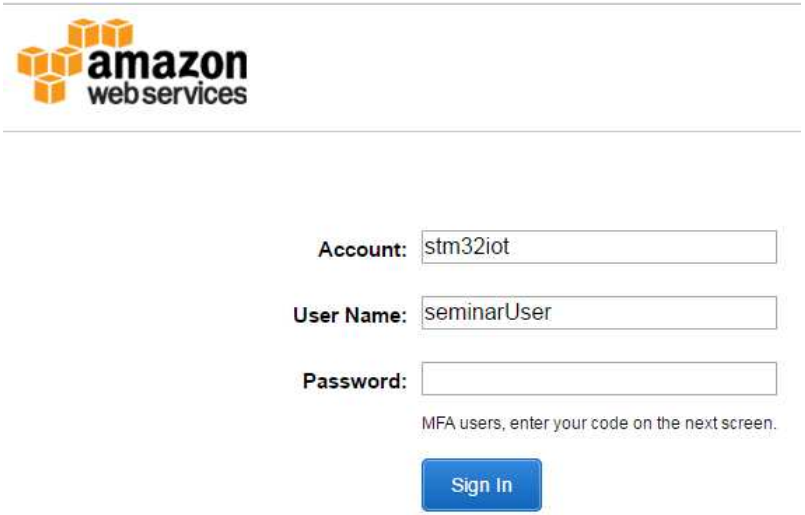
life.augmented

Lab Goal

- In this lab you are going to:
  1. Create a device on AWS IoT
  2. Generate device certificate and keys
  3. Create a policy for your device
  4. Connect the certificate to the policy

Sign into AWS

- We have created an AWS account for you to use today.
- Open the following URL: https://stm32iot.signin.aws.amazon.com/console
- User Name: seminarUser
- Password: stm32iot



life.augmented

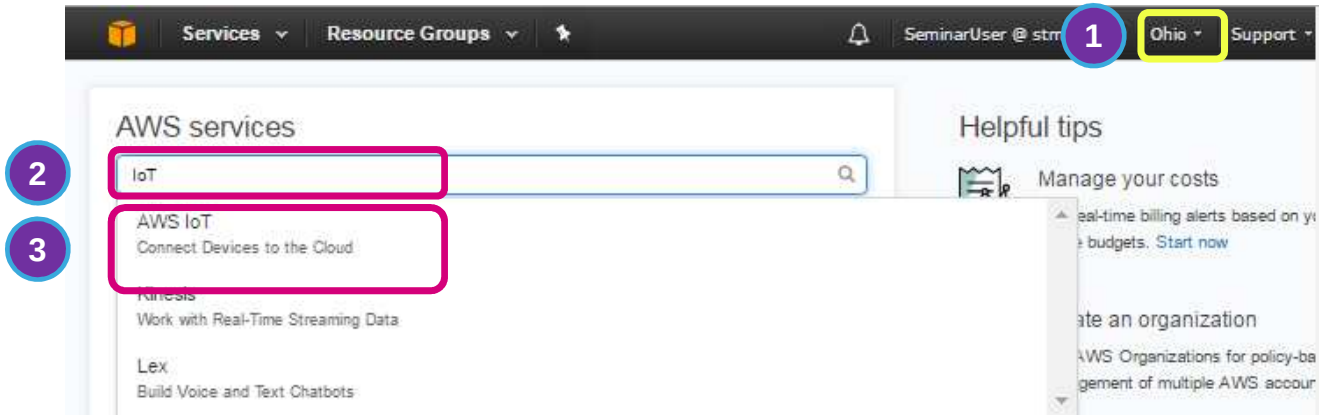
Note

- Although we have created an AWS account for you to use today, you will need to sign up for your own account to continue to test AWS functionality.

- Sign up instructions can be found at: https://aws.amazon.com

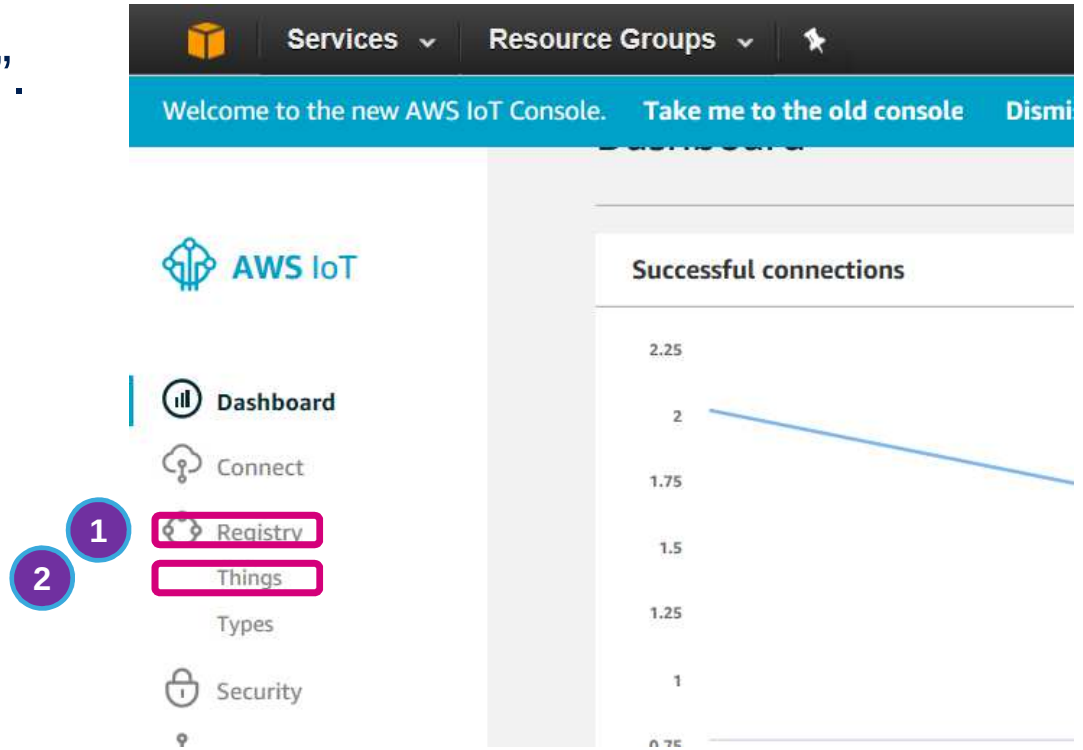
Open AWS IoT Service

1. Make sure that you are connected to the Ohio server
2. Connect to the AWS IoT service by typing AWS IoT in the AWS services search bar.
3. Click on AWS IoT.



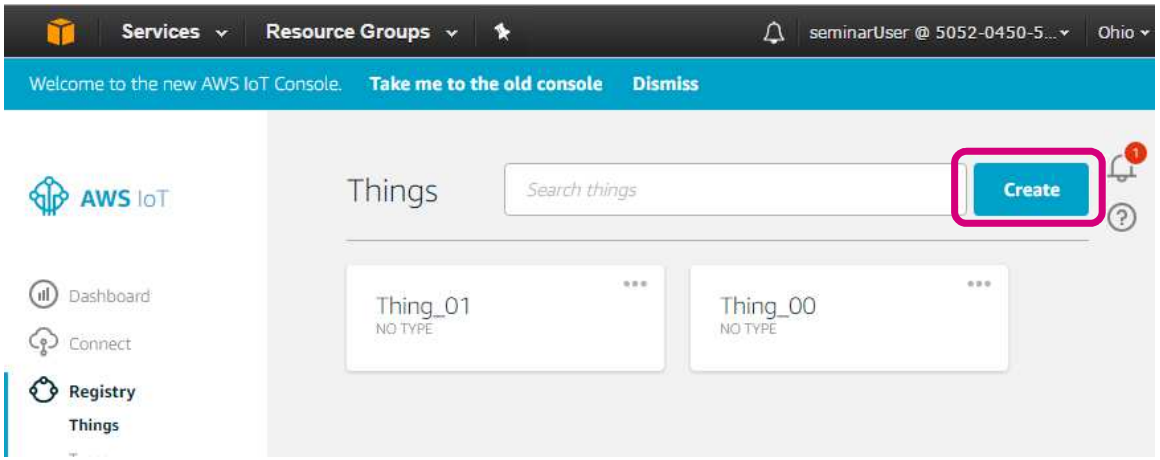
Create a Thing (1/3)

1. Start creating your device by selecting “Registry”.
2. Then select “Things”.



Create a Thing (2/3)

- Click the “Create” button on the top right.

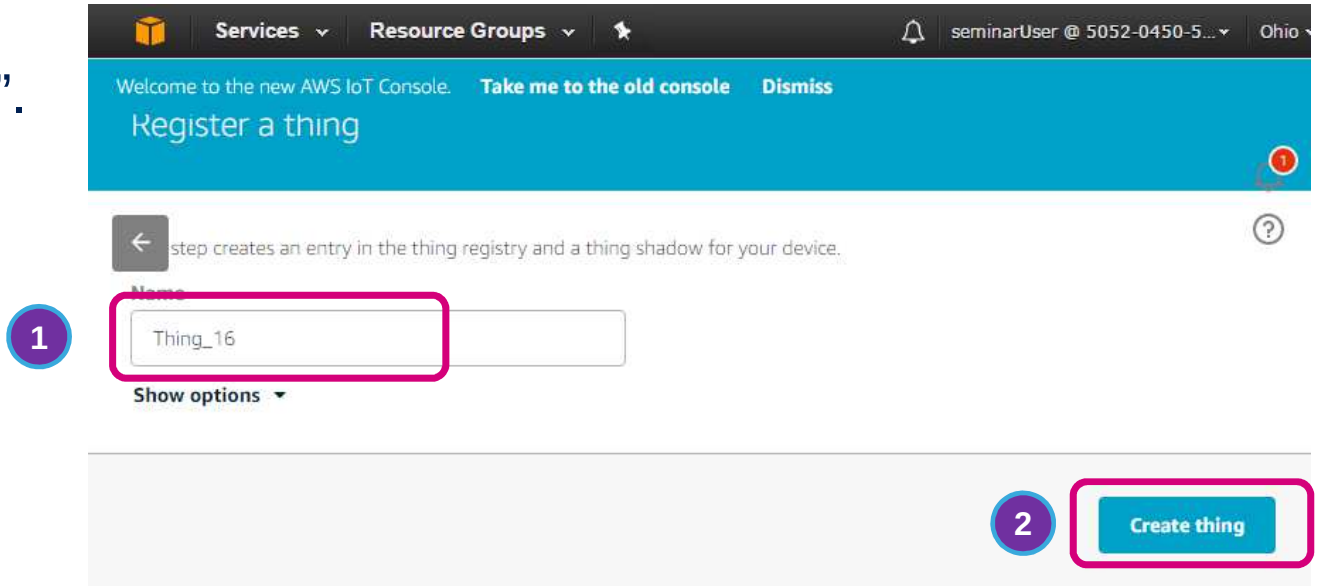


Create a Thing (3/3)

1. Name your “Thing_xx” with xx being your participant number (2 characters).

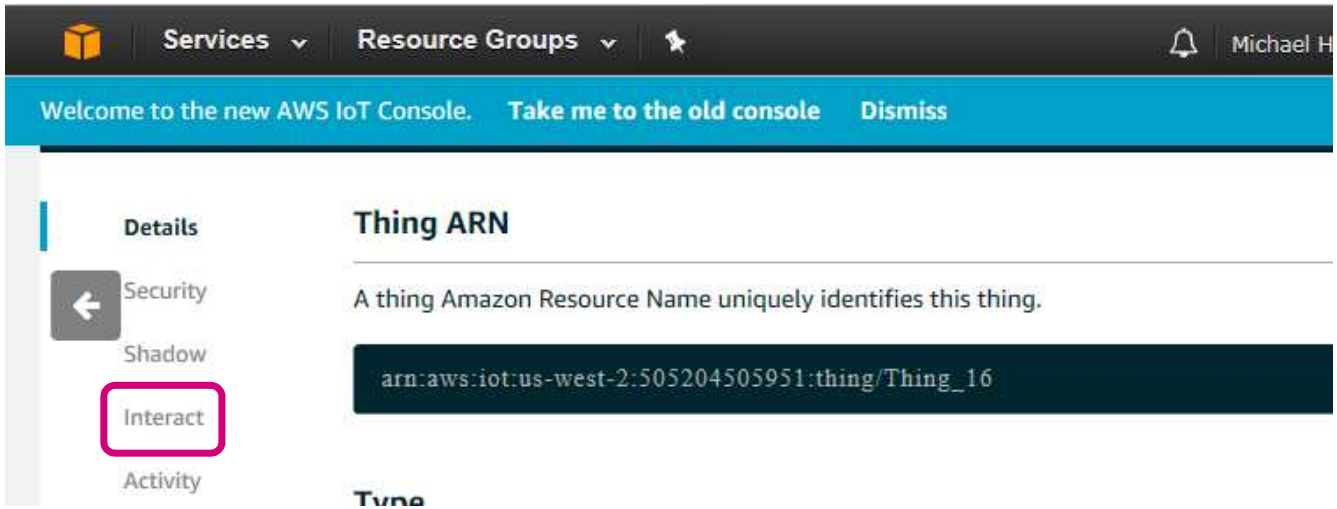
   Example: Thing_08, Thing_16

2. Click “Create thing”.



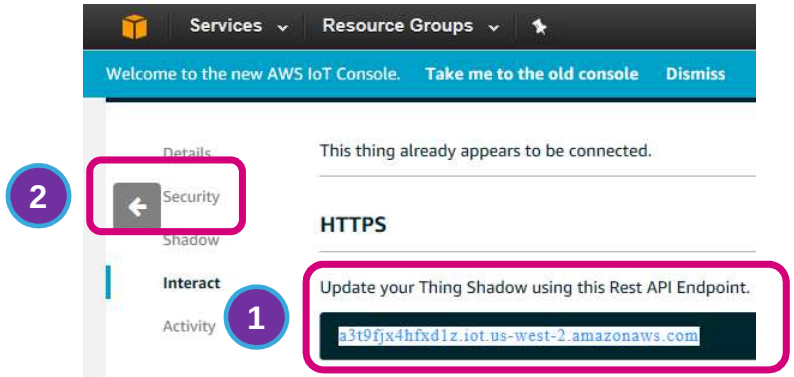
Create a Certificate (1/6)

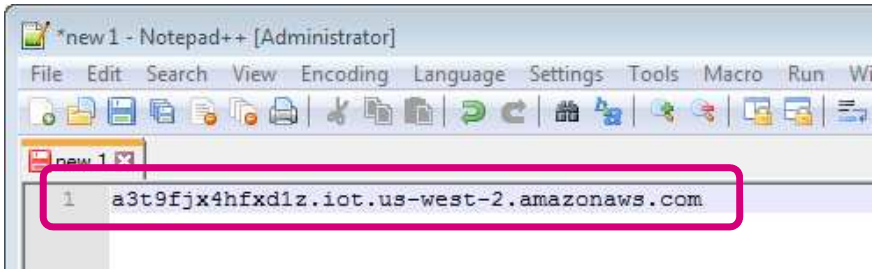
- Your thing has now been created. Select “Interact”.



Create a Certificate (2/6)

1. Copy the Rest API Endpoint information to Notepad++.
2. Then, click “Security”

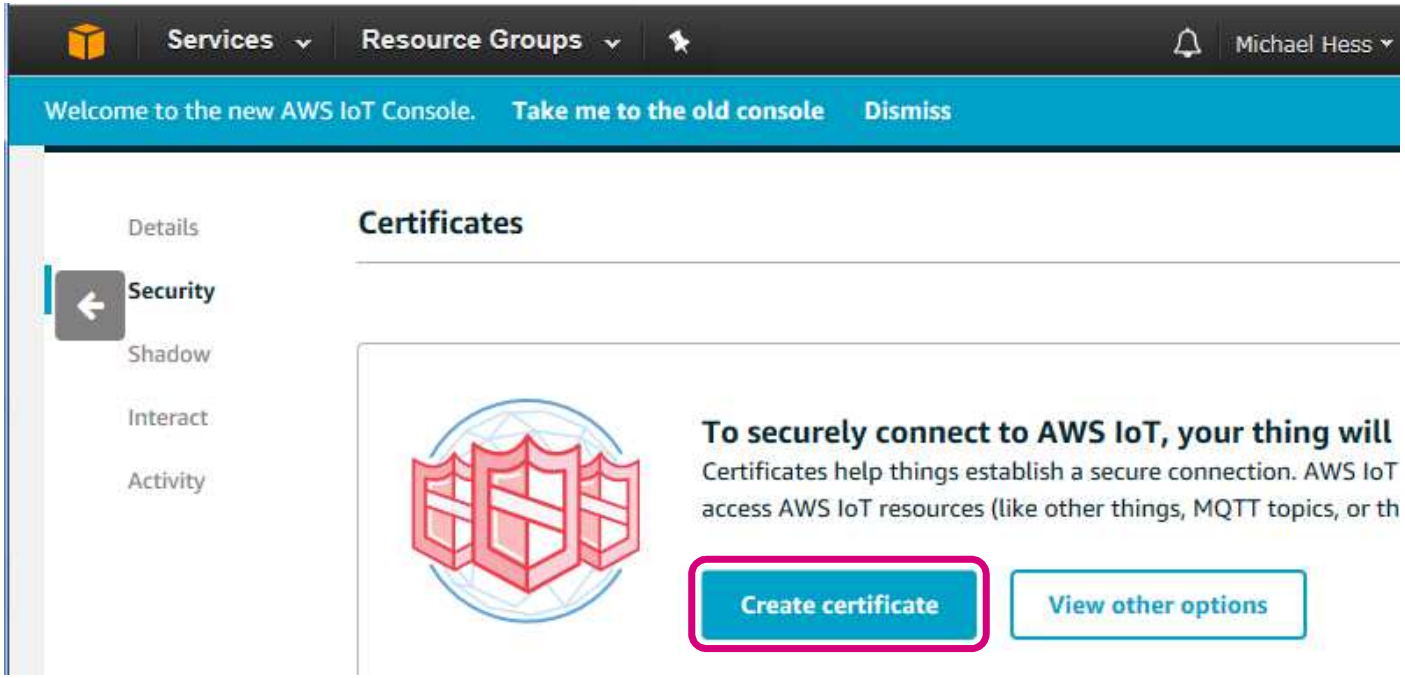




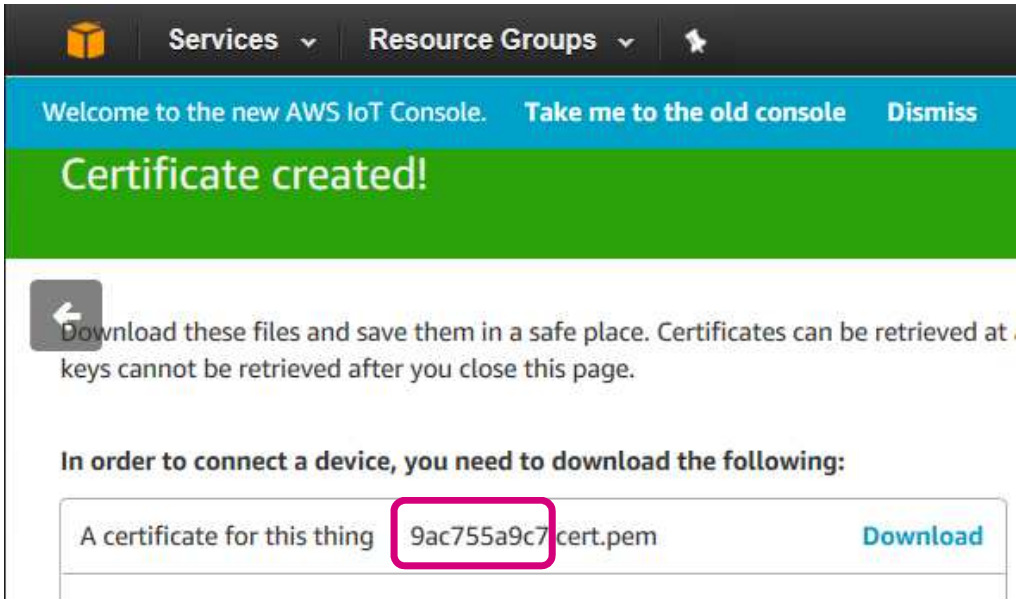
life.augmented

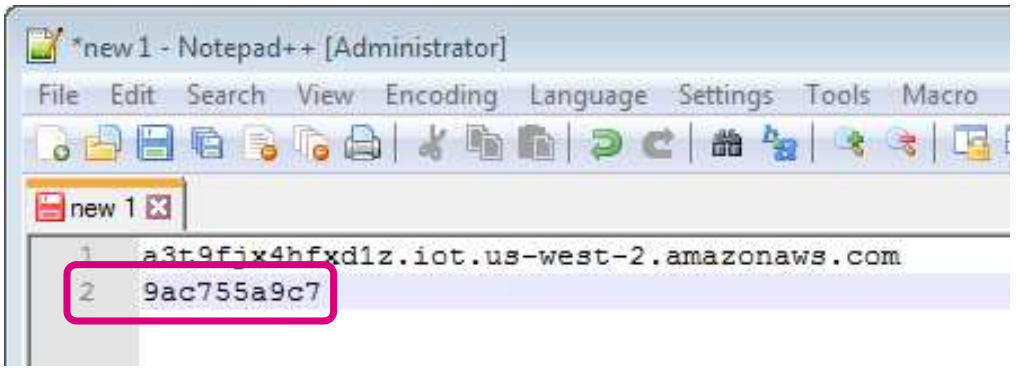
Create a Certificate (3/6)

- Click “Create certificate”.



Create a Certificate (4/6)

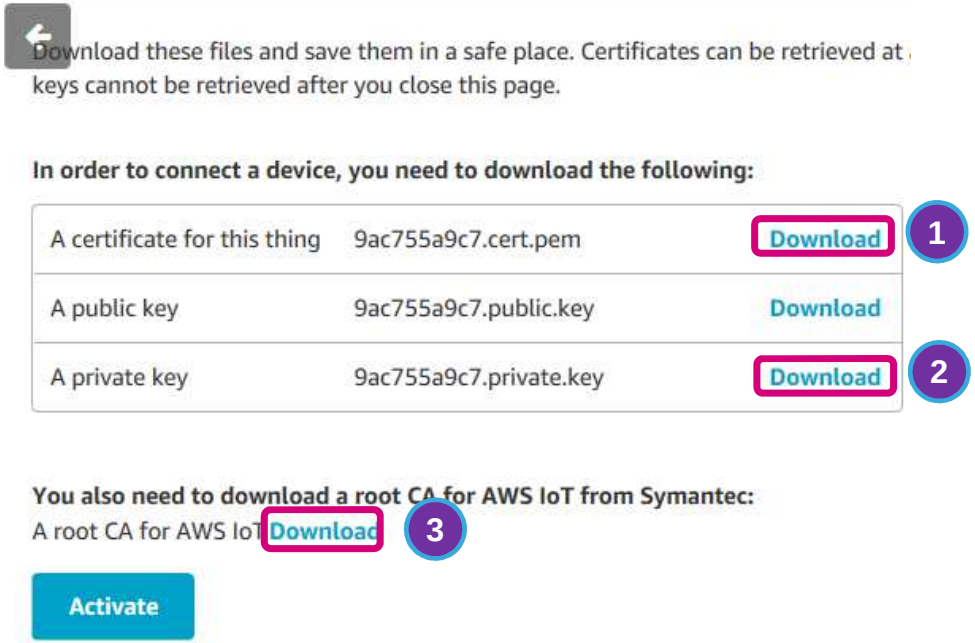




- Copy the certificate number to Notepad++ (just the number)

life.augmented

Create a Certificate (5/6)



- Save the following files to:

`C:\STM32L4_DK_IoT_Node_Seminar\Certificates`

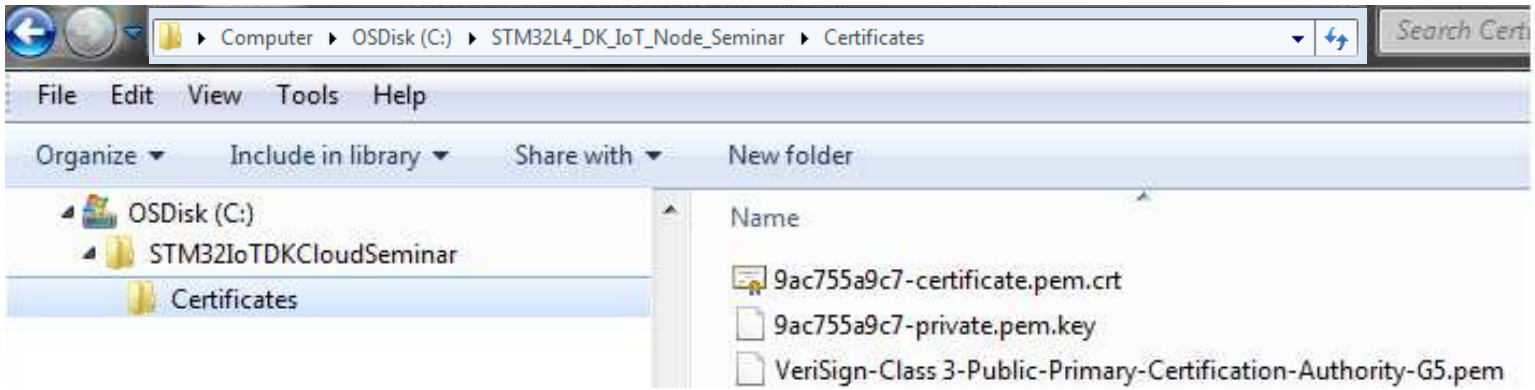
1. Your **Thing certificate** (xxxxx.cert.pem),
2. Your **Thing Private key** (xxxxx.private.key)
3. AWS IoT **Root CA**

   (Right click on Download then save link as)

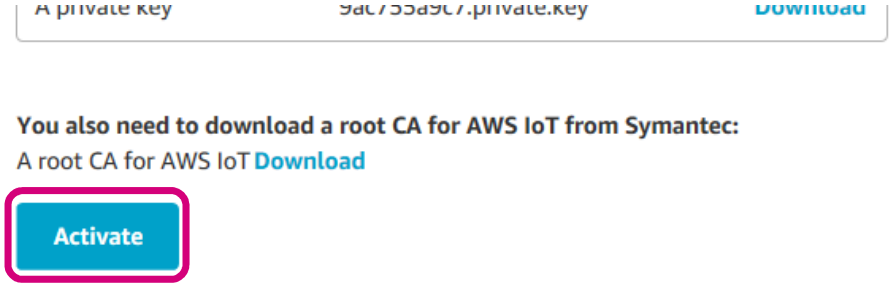
life.augmented

Create a Certificate (6/6)

- Your Certificate folder should look similar to this:



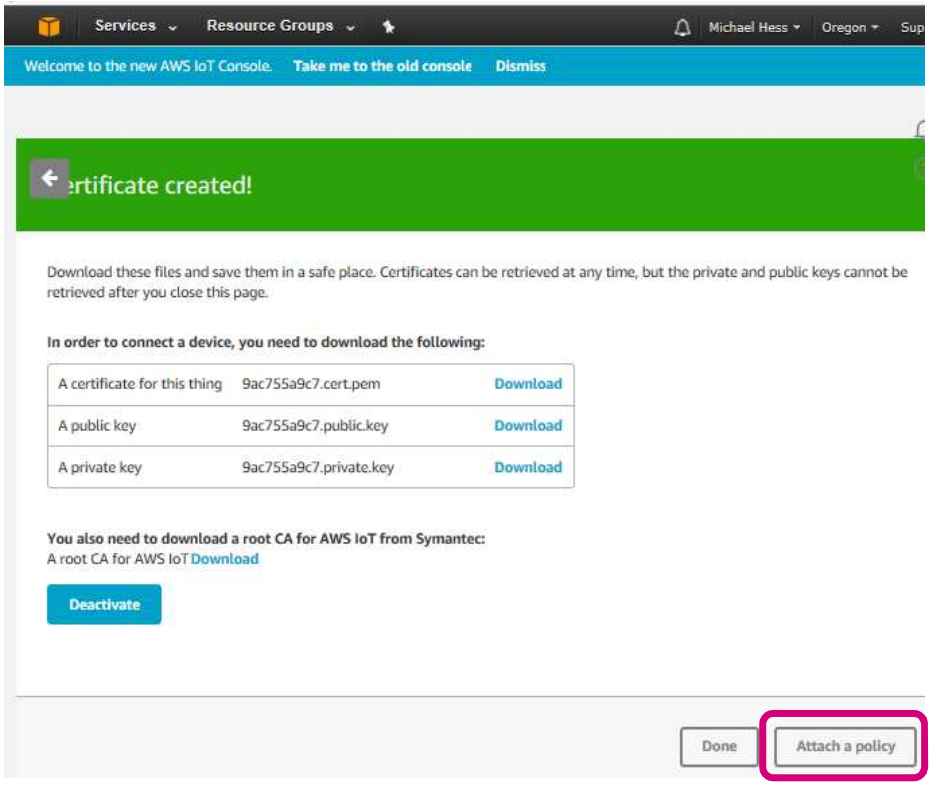
- Click Activate to activate the certificate.



life.augmented

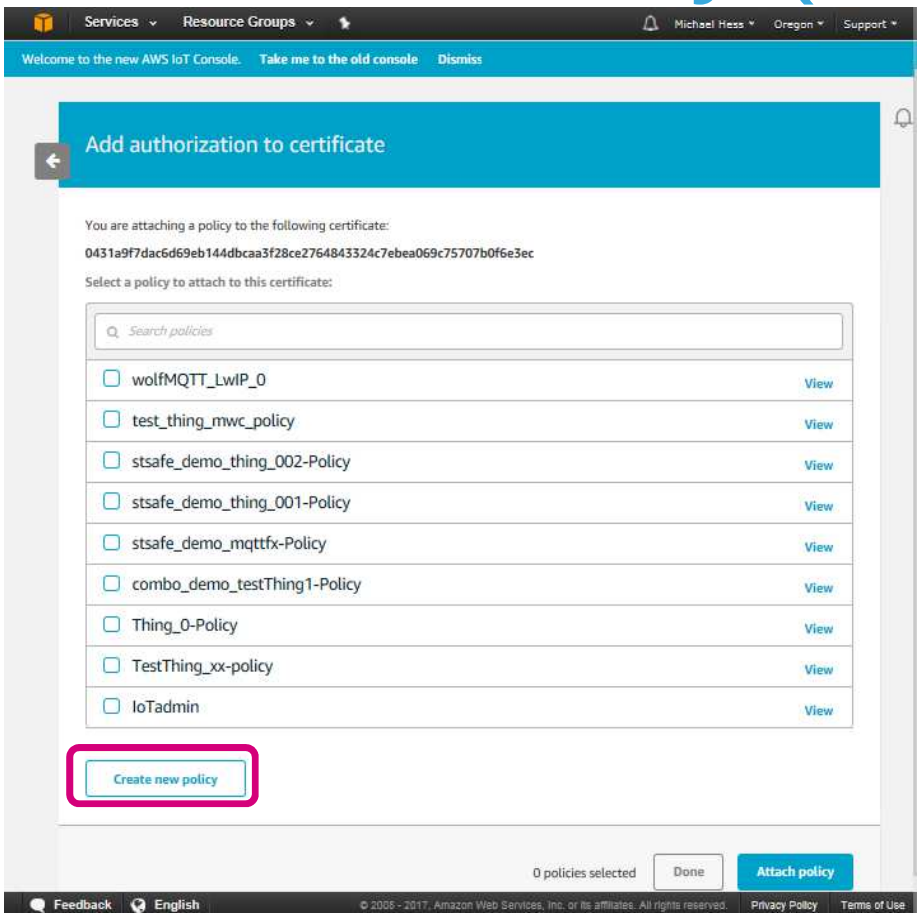
Create a Policy (1/4)

- Attach a policy to your device by clicking “Attach a policy”



Create a Policy (2/4)

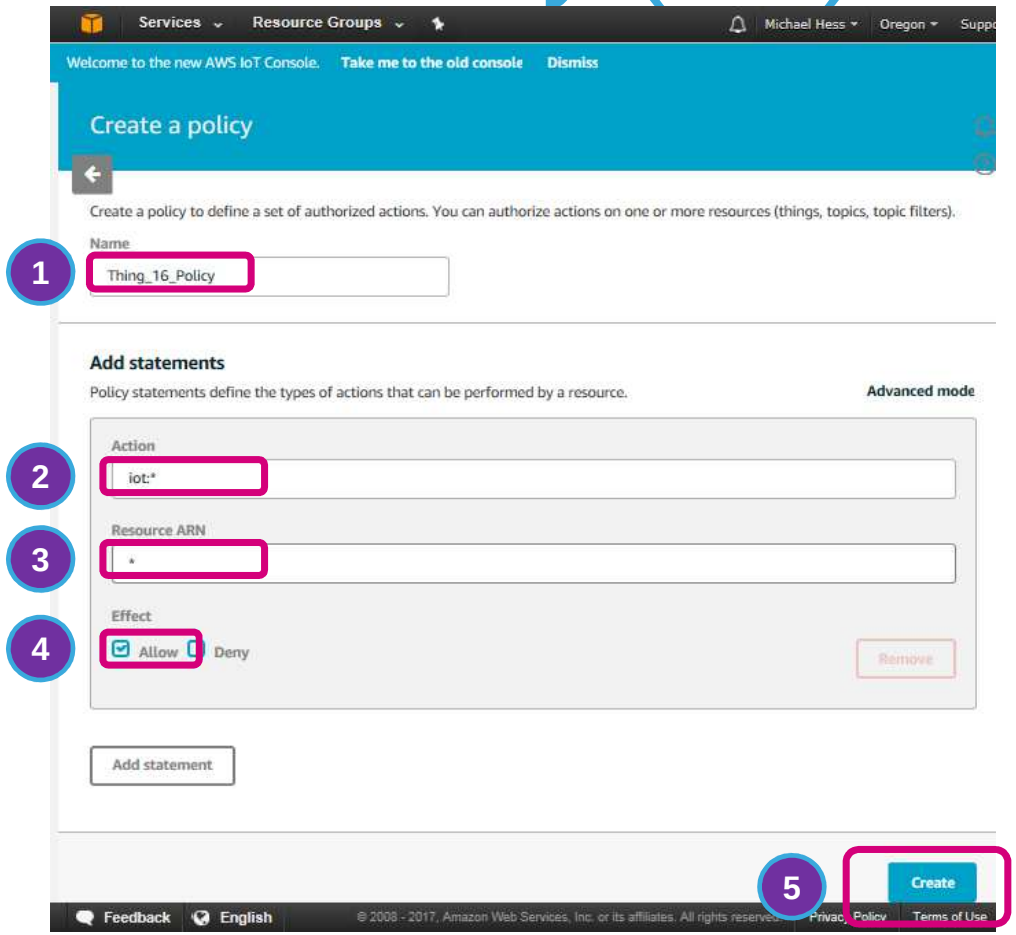
- Click “Create new policy”



life.augmented

Create a Policy (3/4)

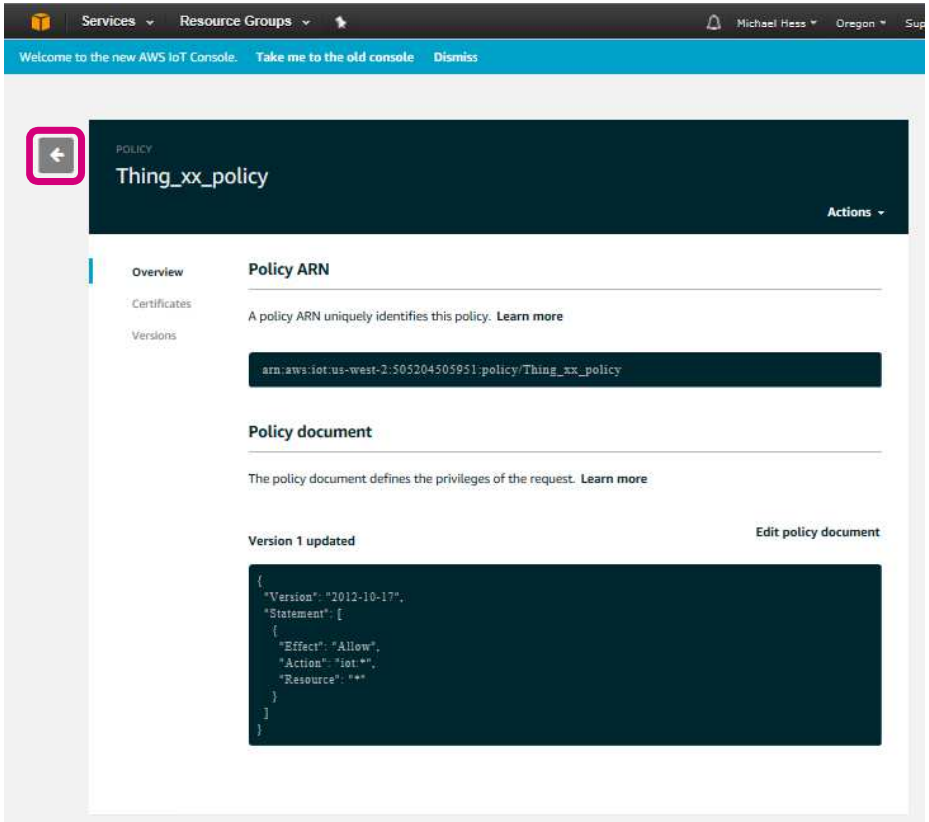
1. Name your policy “Thing_xx_policy” with xx being your participant number
2. Under Action, type “iot:*”
3. Under Resource ARN, type “*”
4. Click “Allow”
5. Click “Create”



life.augmented

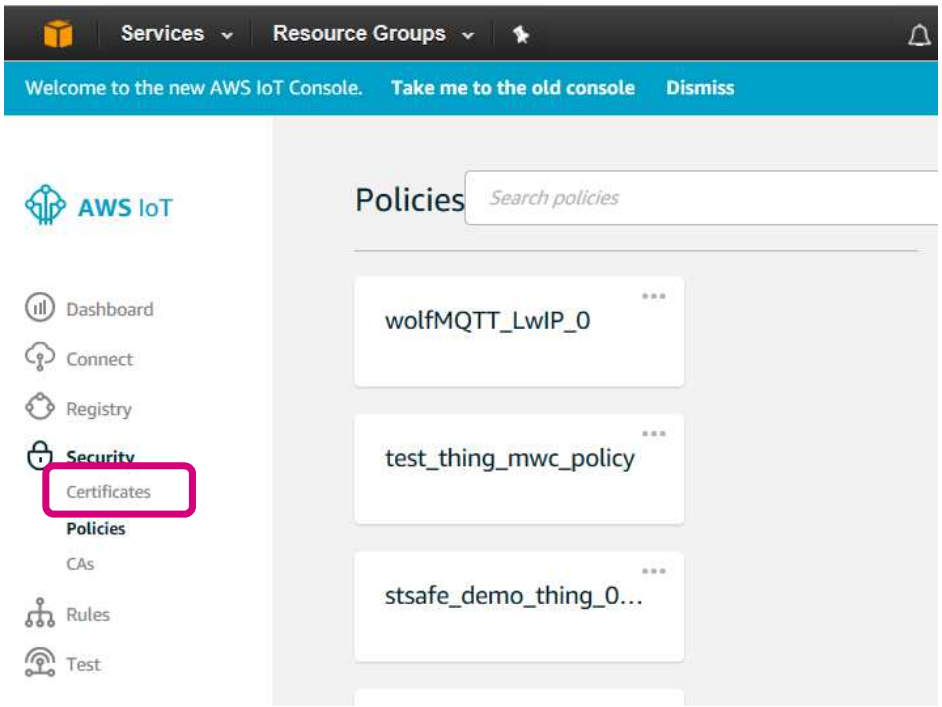
Create a Policy (4/4)

- Your policy has now been created.
- Click the gray arrow.



Attach a Policy (1/5)

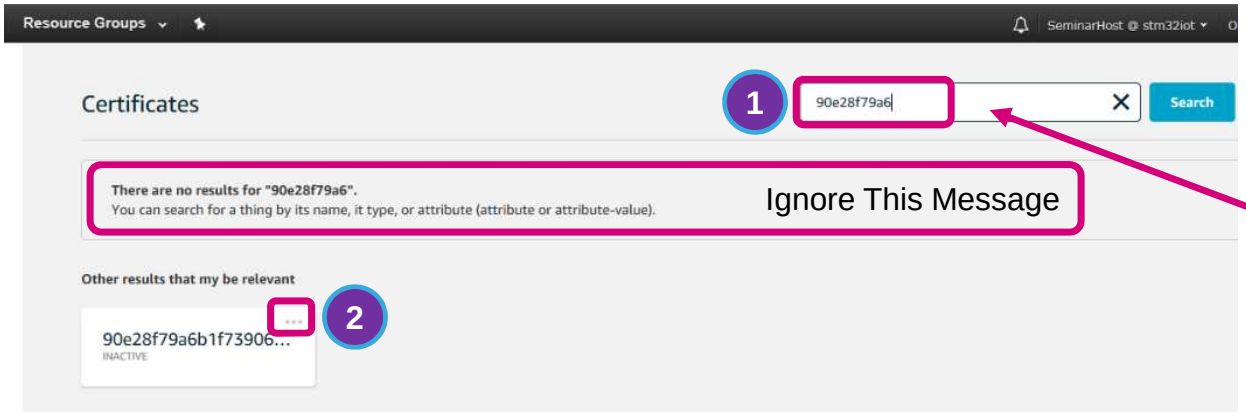
- Select “Certificates” under the “Security” menu.

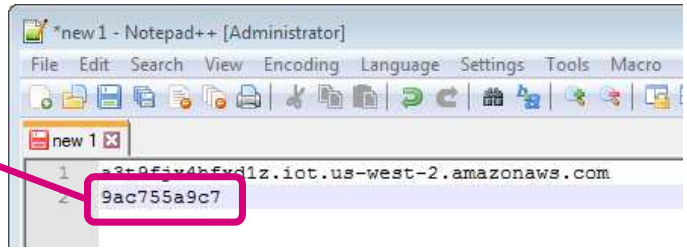


Attach a Policy (2/5)

1. Use the search bar to locate your certificate using the certificate number.

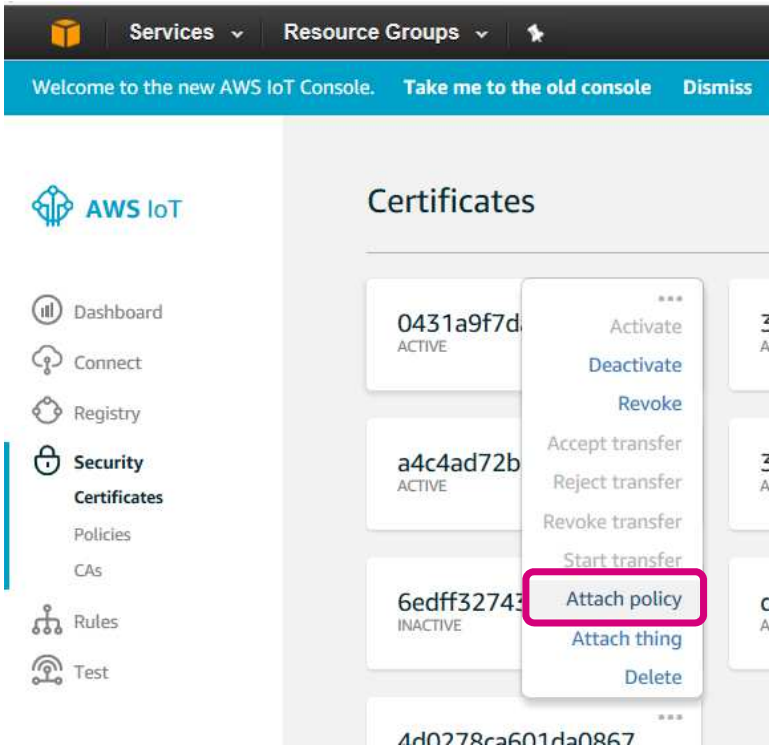
2. Click the “…” in the top right corner of your certificate.





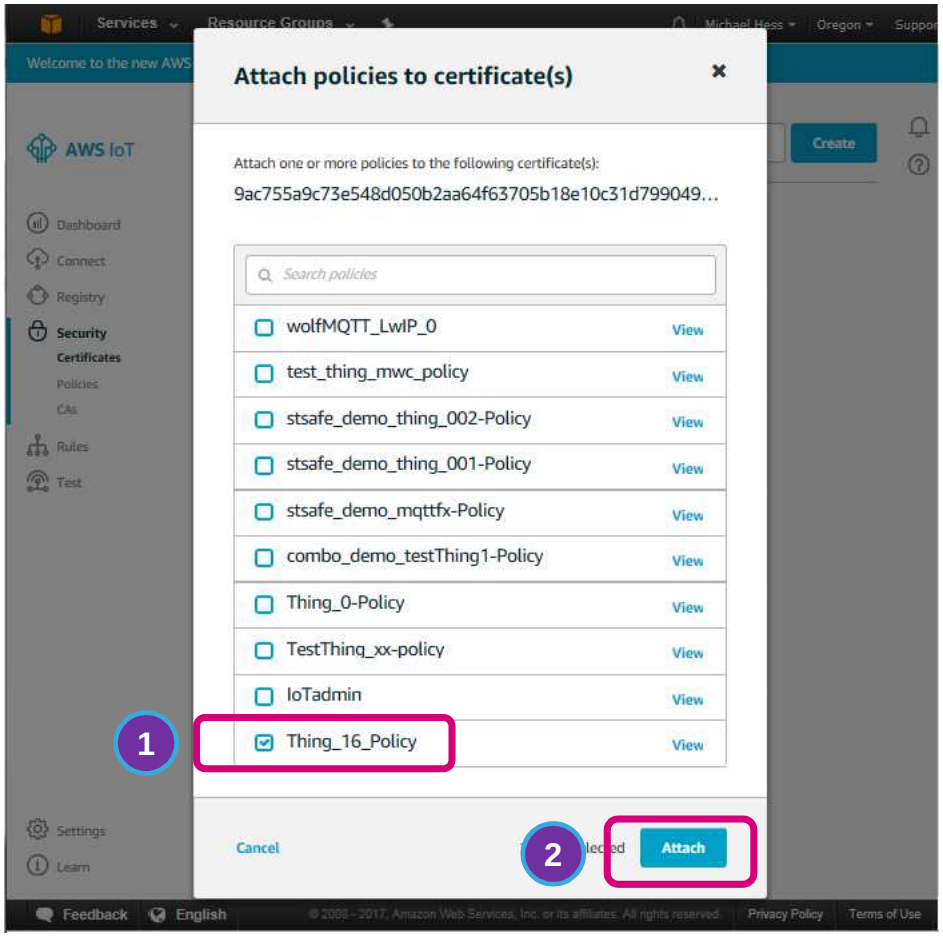
Attach a Policy (3/5)

- Click “Attach policy”.



Attach a Policy (4/5)

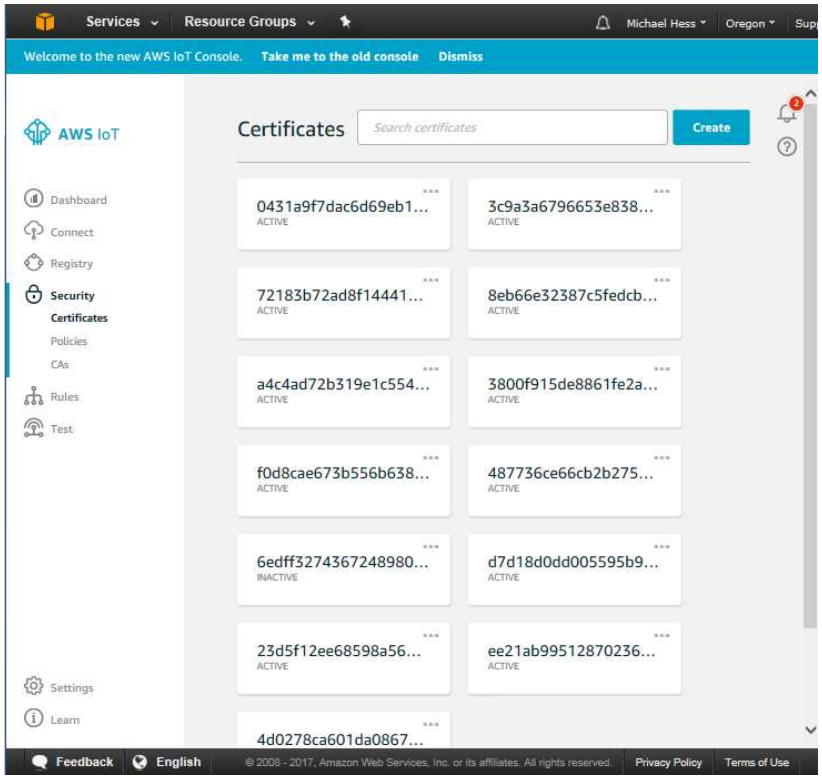
1. Select your policy
2. Click “Attach”

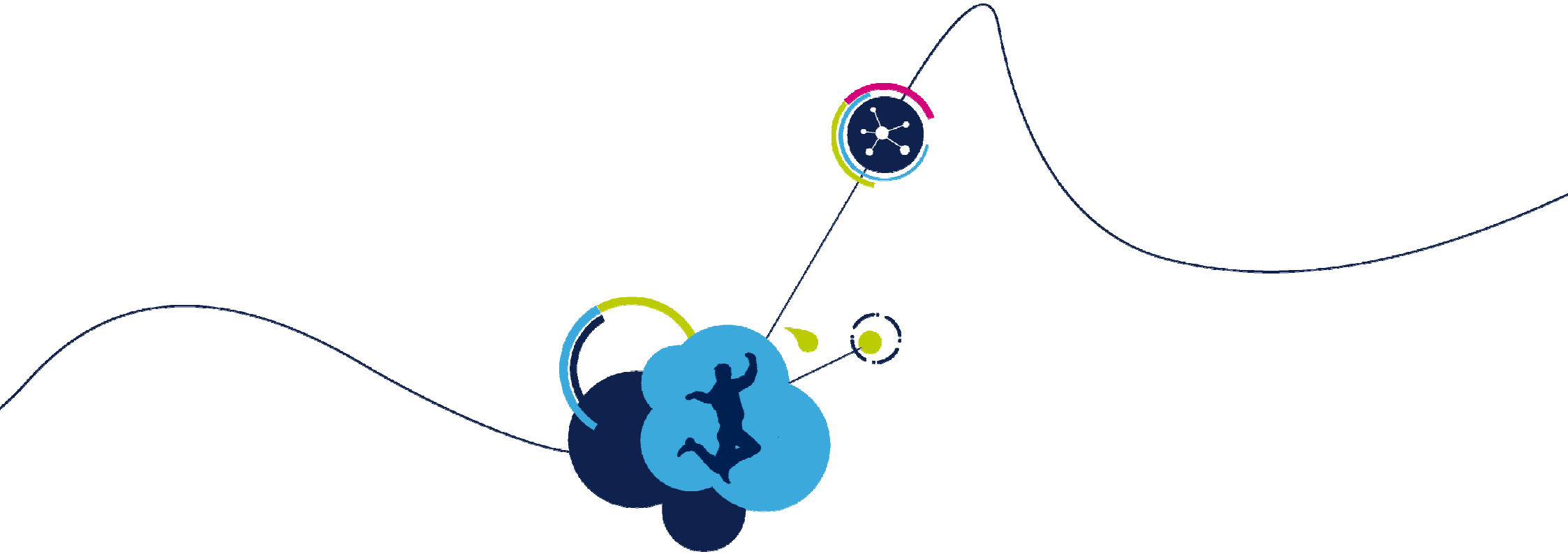


life.augmented

Attach a policy (5/5)

- Your policy has now been attached to your device certificate.
- We are done with the Thing creation on AWS.
- Next is firmware configuration.





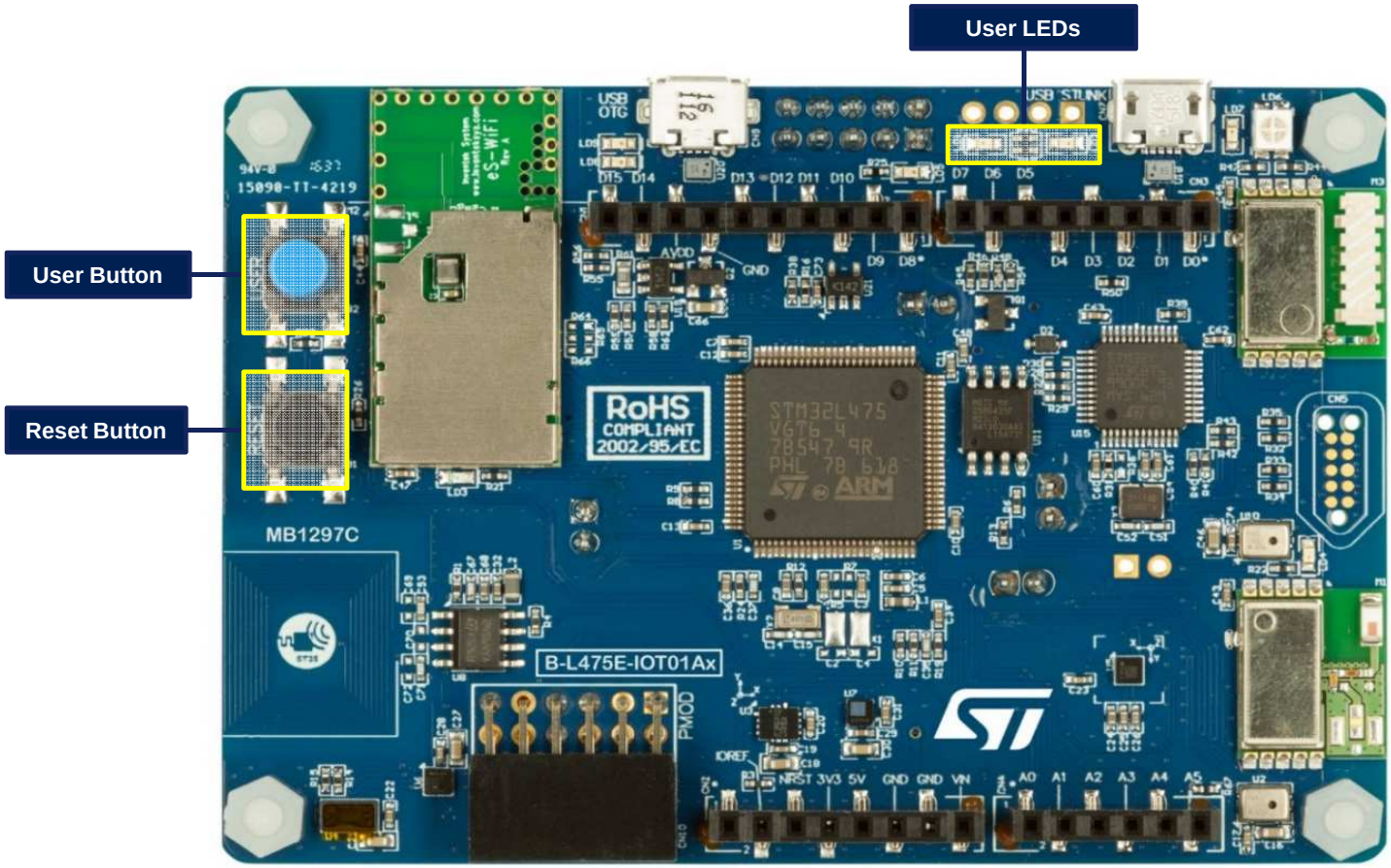
Lab 7 : Connect to AWS IoT & Send Sensor Data

life.augmented

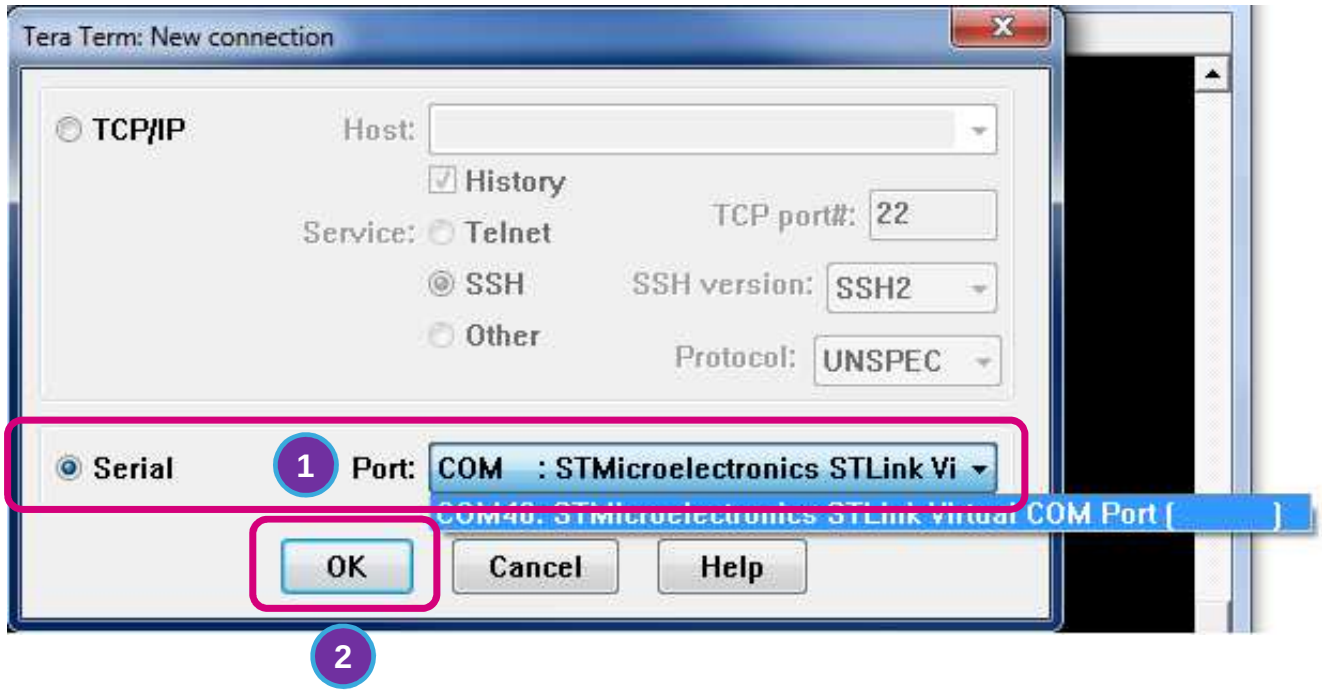
Lab Goal

- In this lab you are going to connect your board to AWS IoT:
  1. Load firmware on the STM32L4 Discovery Kit IoT Node
  2. Set the Wi-Fi credentials
  3. Set the AWS Root Certificate, the device Certificate and the device Private Key
  4. Set the MQTT server address and your device name
  5. Interact with your Thing

STM32L4 Discovery Kit IoT Node



Tera Term Configuration 1/5

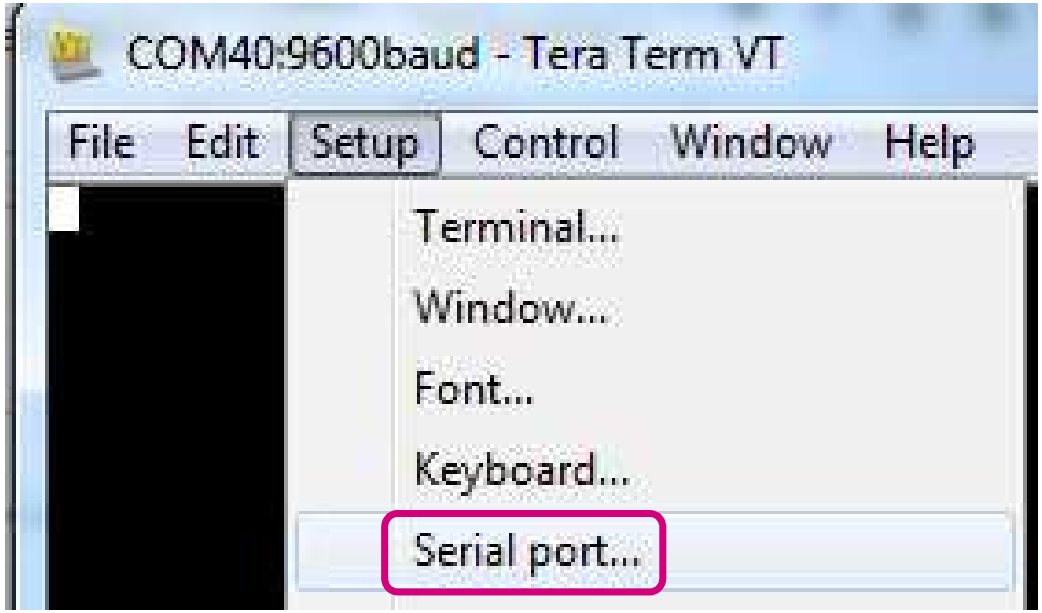


1. Select the **STMicroelectronics STLink Virtual COM Port**
2. Click **OK**

life.augmented

Tera Term Configuration 2/5

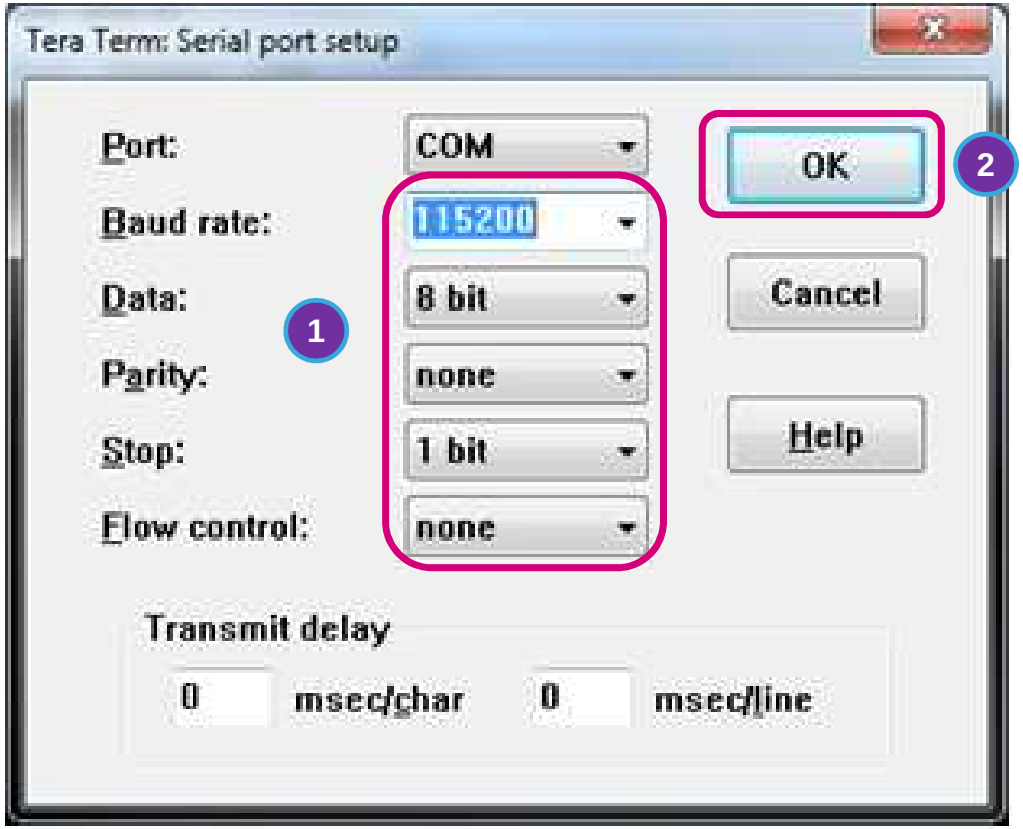
1. Click **Setup** -> **Serial port…**



Tera Term Configuration 3/5

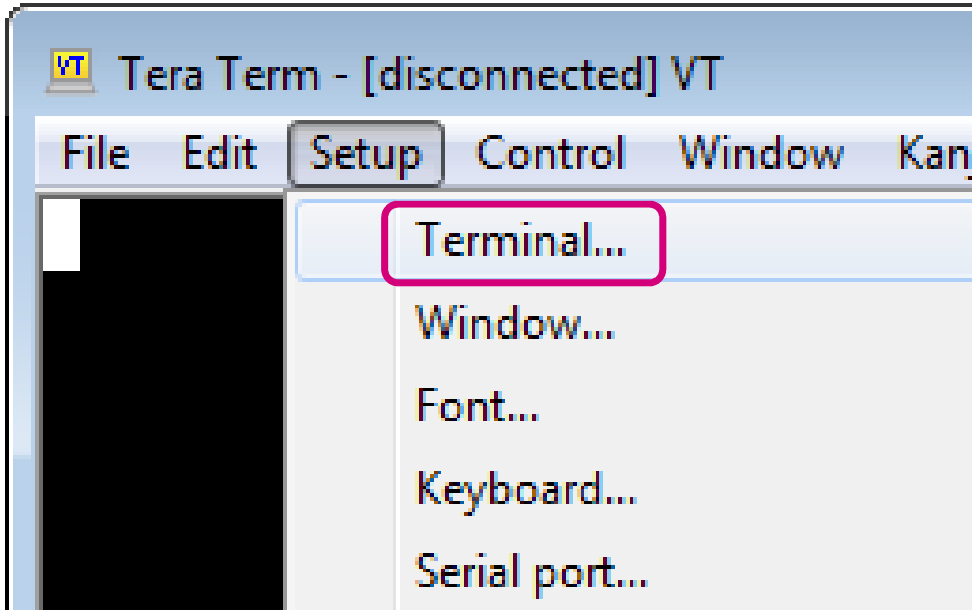
1. Set the following:
   - Baud rate : **115200**
   - Data : **8 bit**
   - Parity : **none**
   - Stop : **1 bit**
   - Flow control : **none**

2. Click **OK**



Tera Term Configuration 4/5

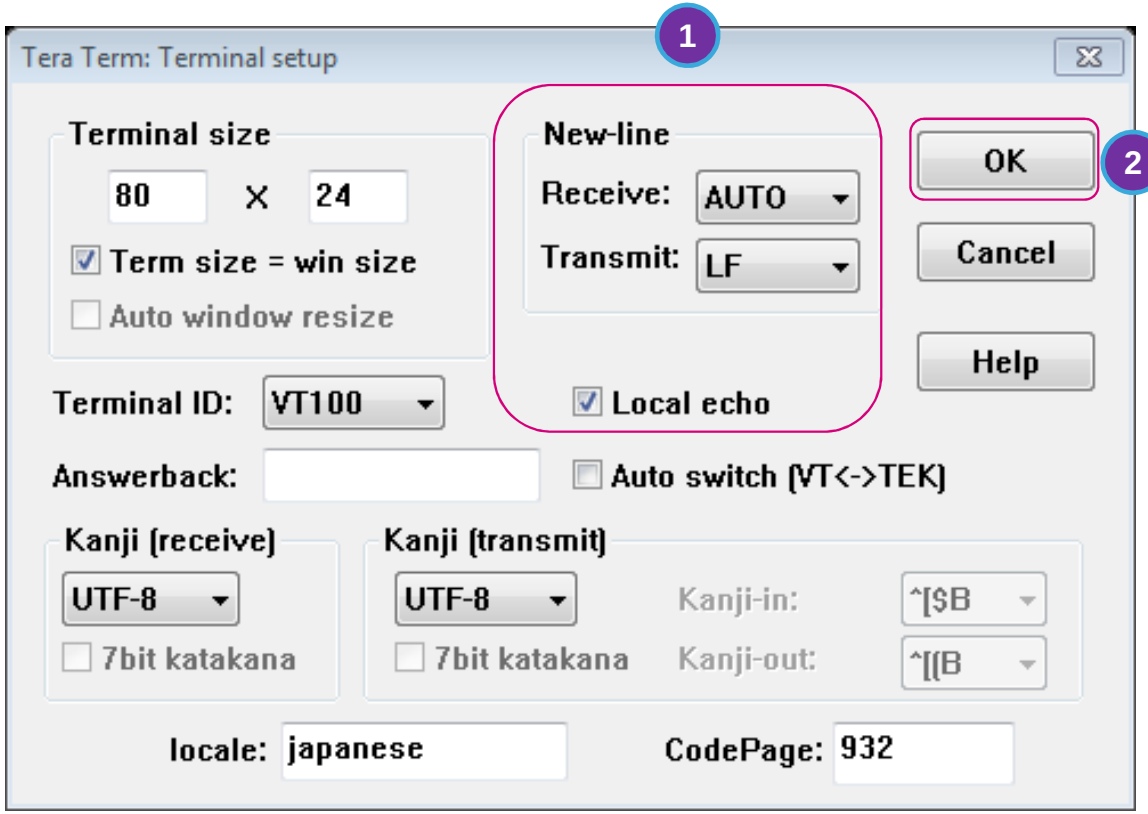
1. Click **Setup** -> **Terminal…**



Tera Term Configuration 5/5

1. Set the following:
   - New-Line Receive : **AUTO**
   - New-Line Transmit : **LF**
   - Enable **Local echo**

2. Click **OK**



Open AWS_HandsOn Project

- Now we are going to configure the AWS firmware program to set the network SSID and password, set the AWS Certificates and establish an AWS IoT MQTT connection.

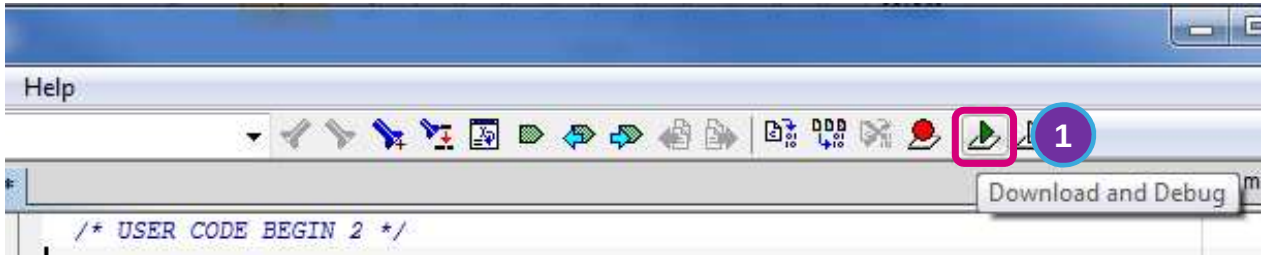
1. Close the previous IAR project.
2. Double click on the **`project.eww`** file located here:

`C:\STM32L4_DK_IoT_Node_Seminar\Hands-on\Labs\Projects\B-L475E-IOT01\Applications\Cloud\AWS\EWARM`

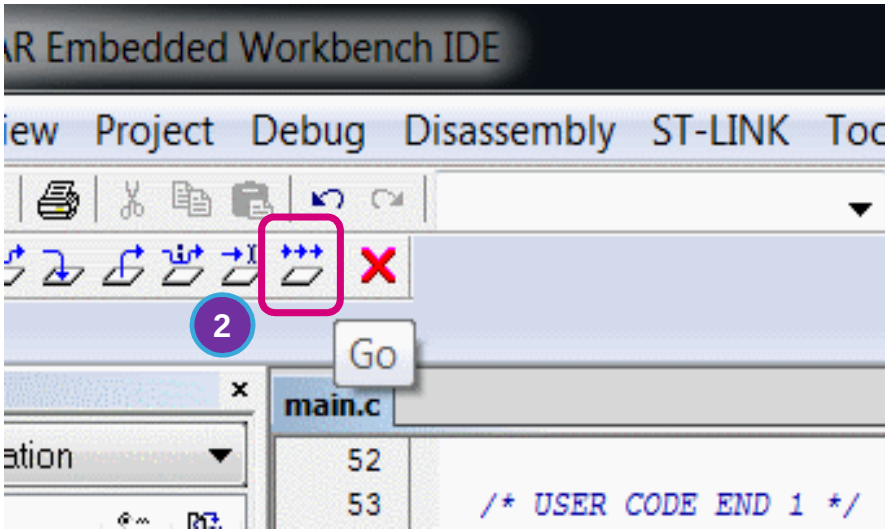
life.augmented

Load and Run

1. Click the GREEN ARROW to Build the Project, **Download** and start the **debugger**. (Ctrl + D)

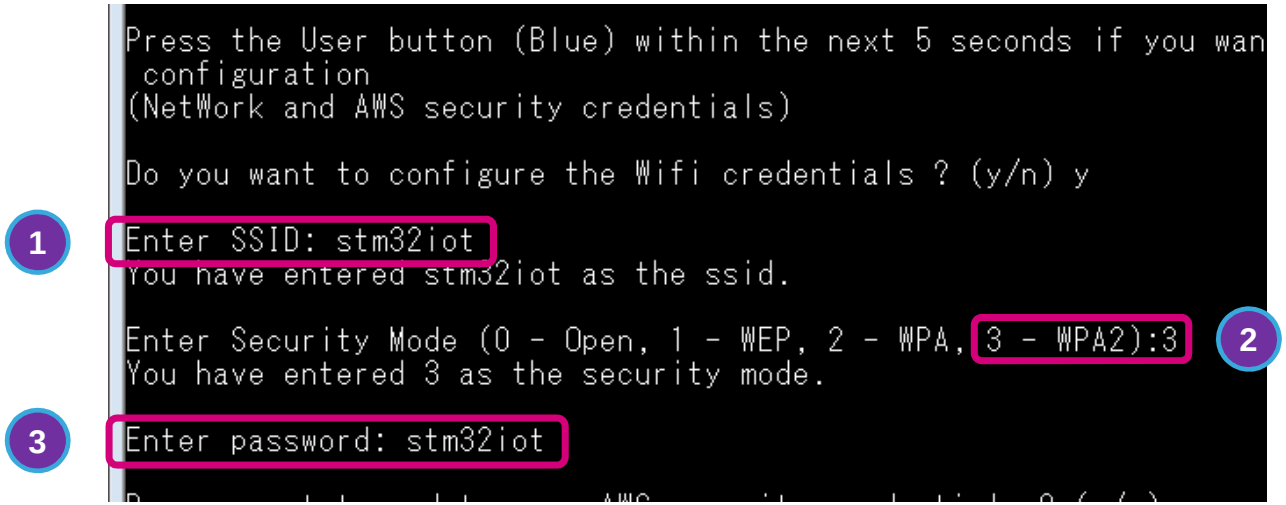


2. Click the triple-arrow **GO** button! (F5)



Wi-Fi Configuration

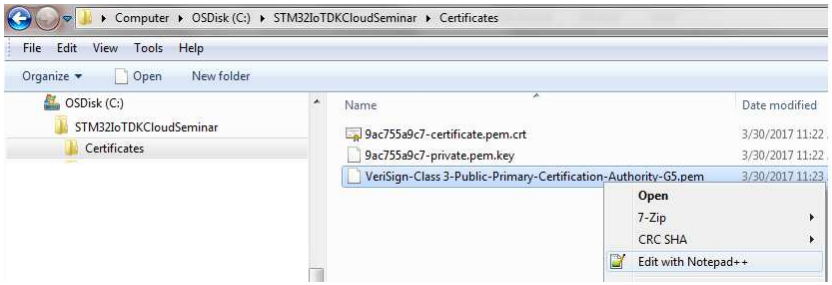
1. Set the SSID to **`stm32iot`**
2. Type **`3`** for the WPA2 Security mode
3. Use **`stm32iot`** for network password

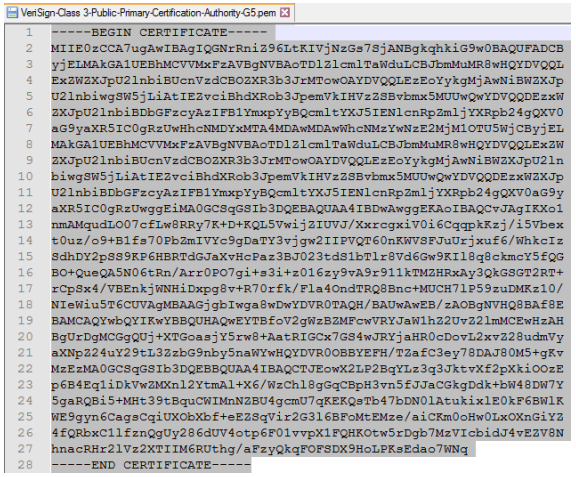


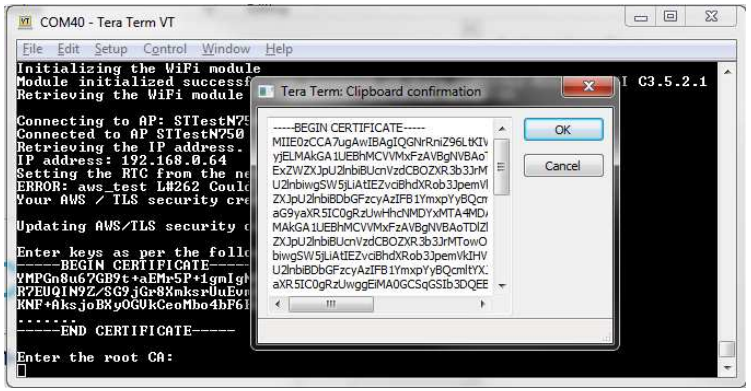
life.augmented

Set the AWS Root-CA Certificate

1. Open the **`VeriSign-Class 3-Public-Primary-Certification-Authority-G5.pem`** with Notepad++
2. Copy the certificate content (Ctrl + A, Ctrl + C)
3. Paste the certificate in Tera Term using “Edit -> Paste” (Do not use Ctrl + V) and click OK
4. Click Enter







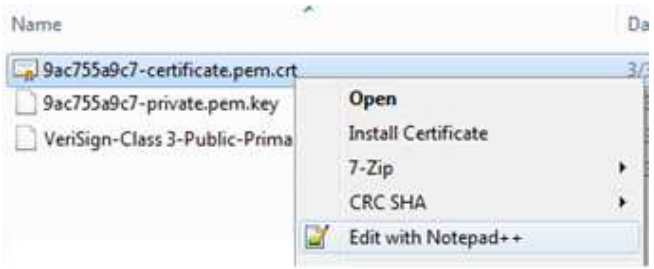
life.augmented

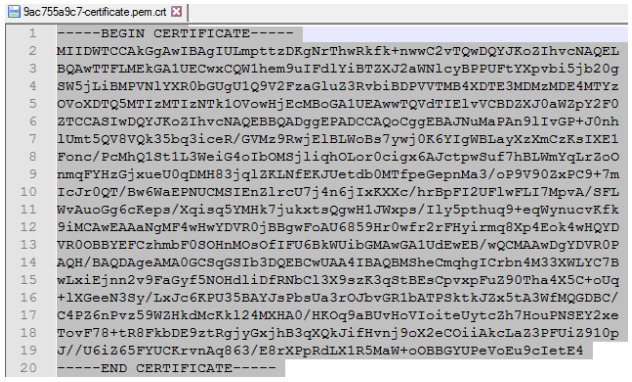
Set your Thing Certificate

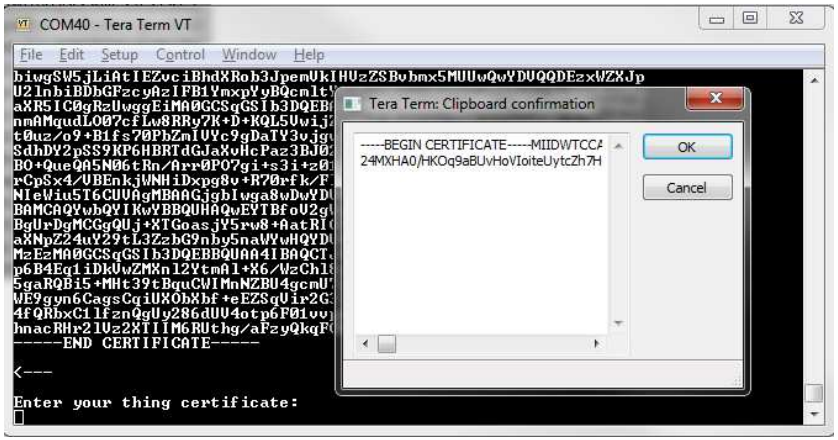
1. Open your **`xxxx-certificate.pem.crt`** with Notepad++

2. Copy the certificate content (Ctrl +A, Ctrl + C)

3. Paste the certificate in Tera Term using “Edit -> Paste” (Do not use Ctrl + V) and click OK







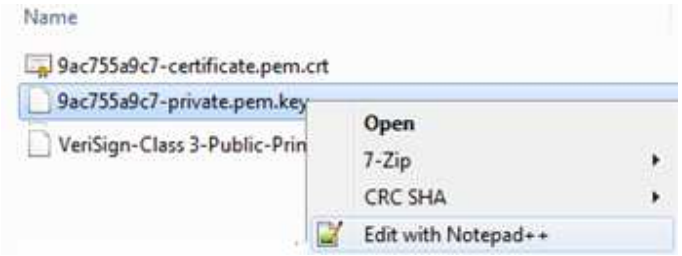
life.augmented

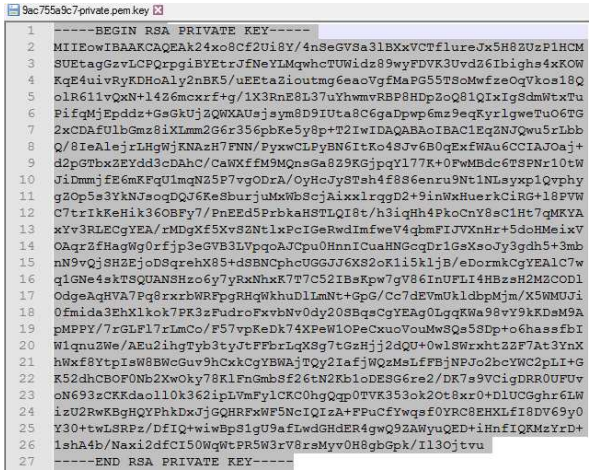
Set your Thing Private Key

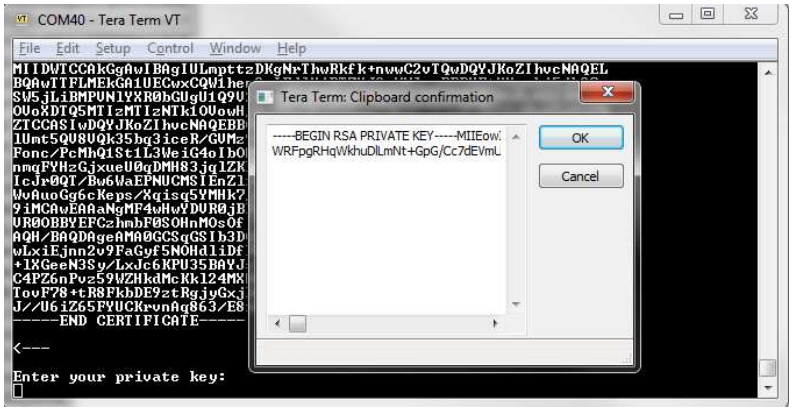
1. Open your **`xxxx-private.pem.key`** with Notepad++

2. Copy the certificate content (Ctrl +A, Ctrl + C)

3. Paste the certificate in Tera Term using “Edit -> Paste” (Do not use Ctrl + V) and click OK



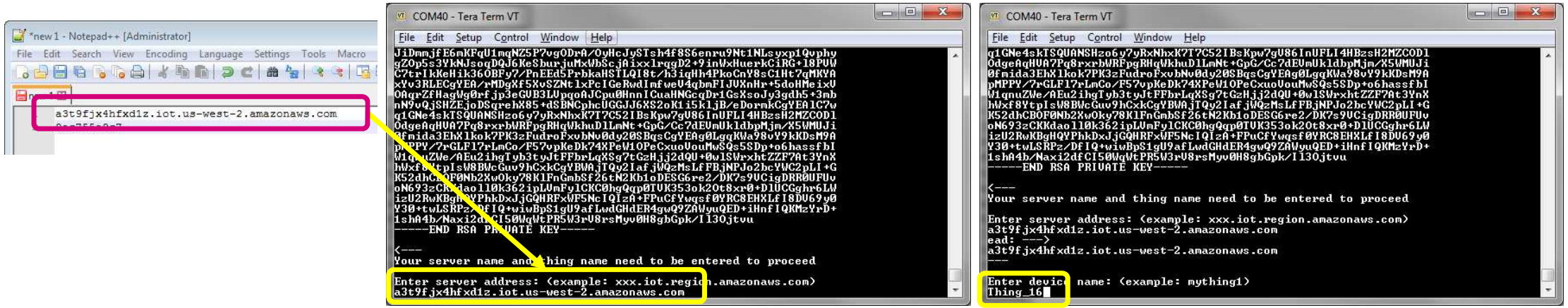




life.augmented

Set the Server and Thing Names

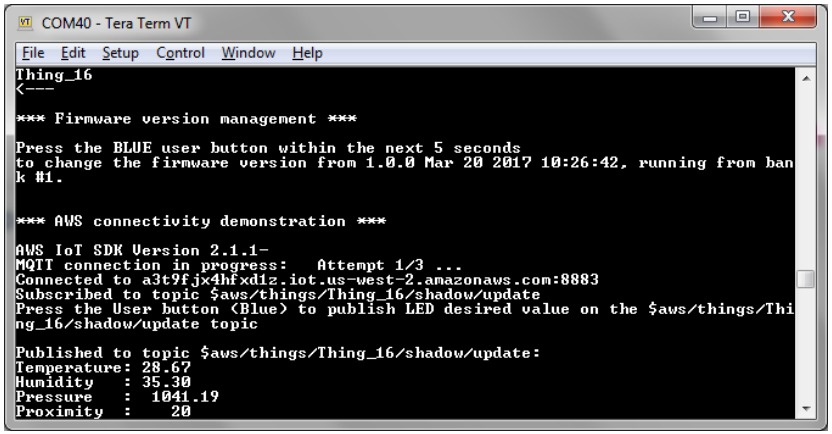
1. Copy the server address from the text file and paste it to Tera Term, then press Enter
2. Set the device name to "**`Thing_xx`**" with **`xx`** being your participant number (2 characters). Example: `Thing_08`, `Thing_16`

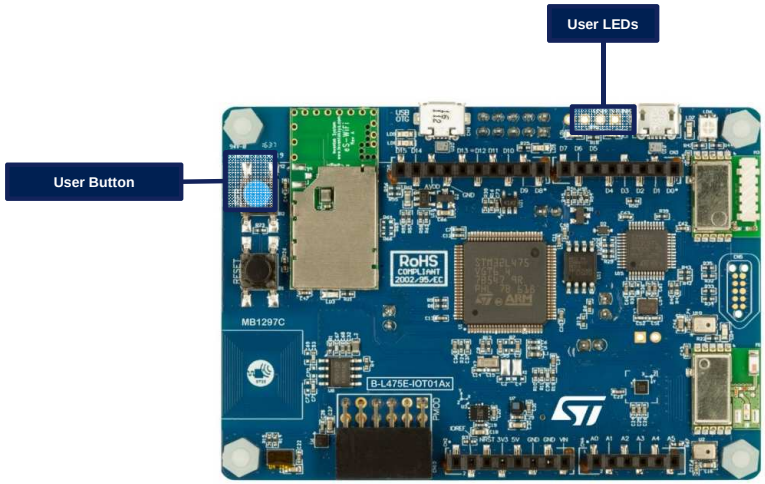


life.augmented

Send Data to AWS

- The board will start sending sensor data once it is connected to AWS (every 10s for 10mn).
- Press the user button (the blue button) and the board send/receive LED messages to/from AWS IoT.
- The LED on your board will toggle every time the LED message is received.

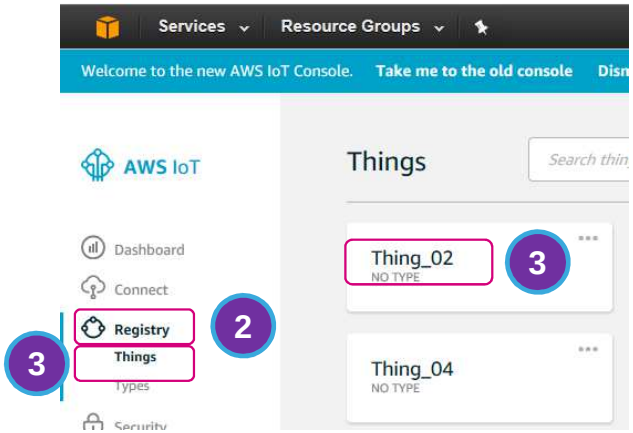


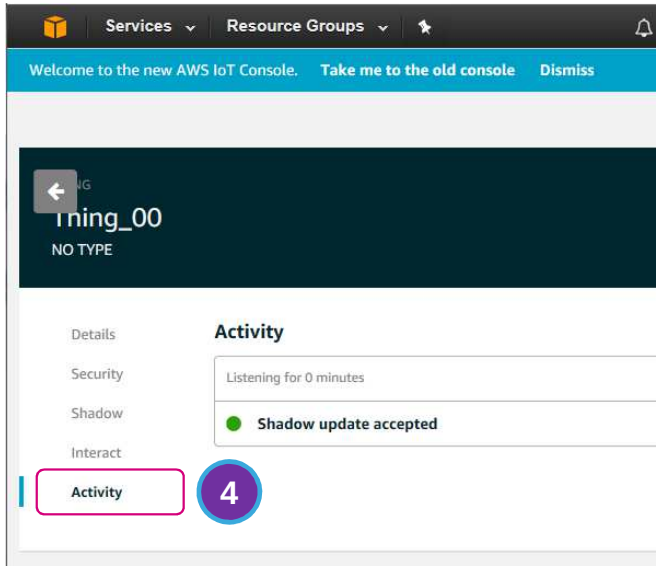


life.augmented

Monitor your Thing Activity

1. Select the AWS Console on your browser
2. Select Registry
3. Select Things, then select your Thing
4. Click on Activity

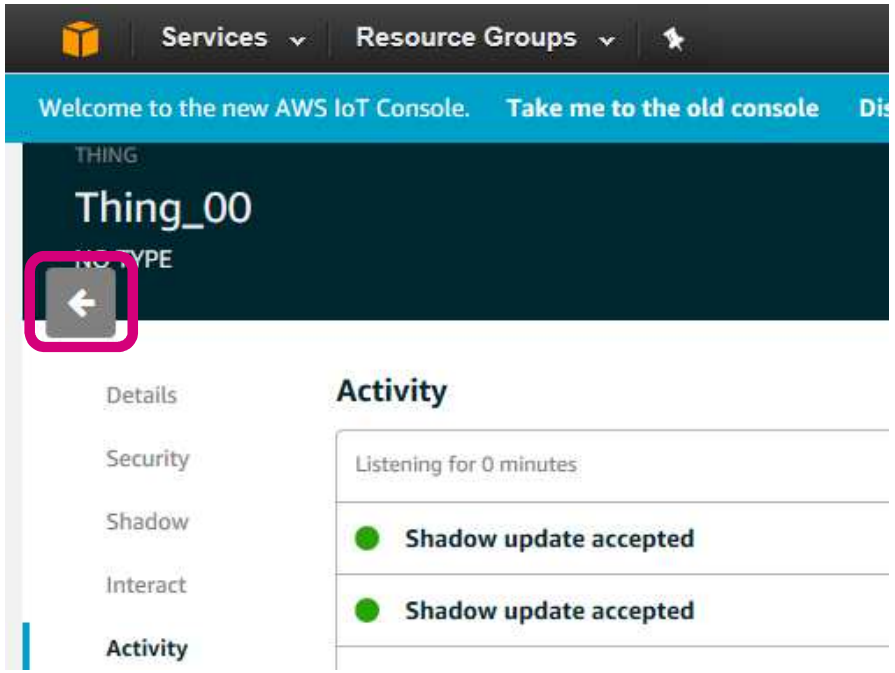




life.augmented

Interact with your Thing (1/9)

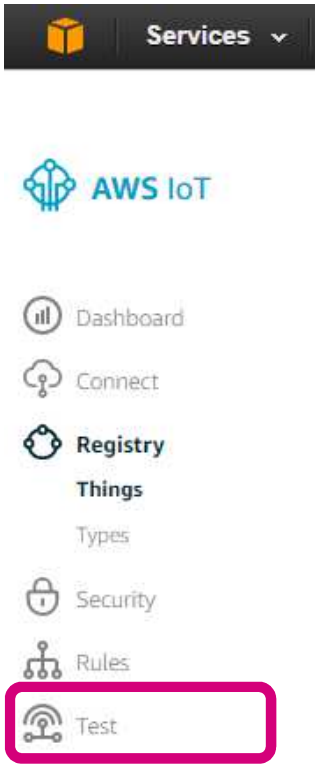
Click the gray back arrow.



Interact with your Thing (2/9)

Select “Test” on the left panel.

This will open a web based MQTT Client.

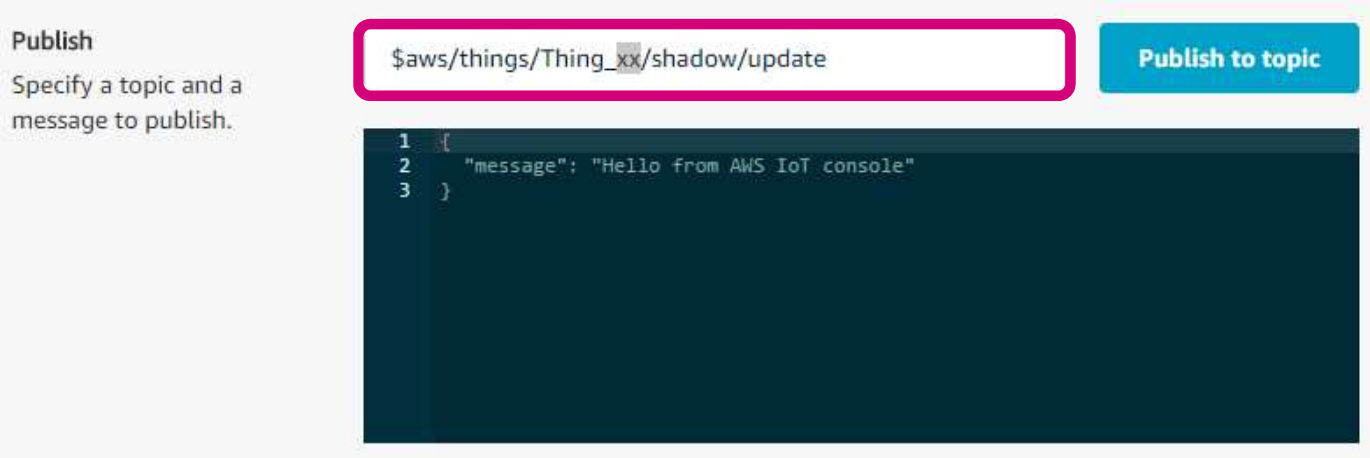


Interact with your Thing (3/9)

Under Publish, type

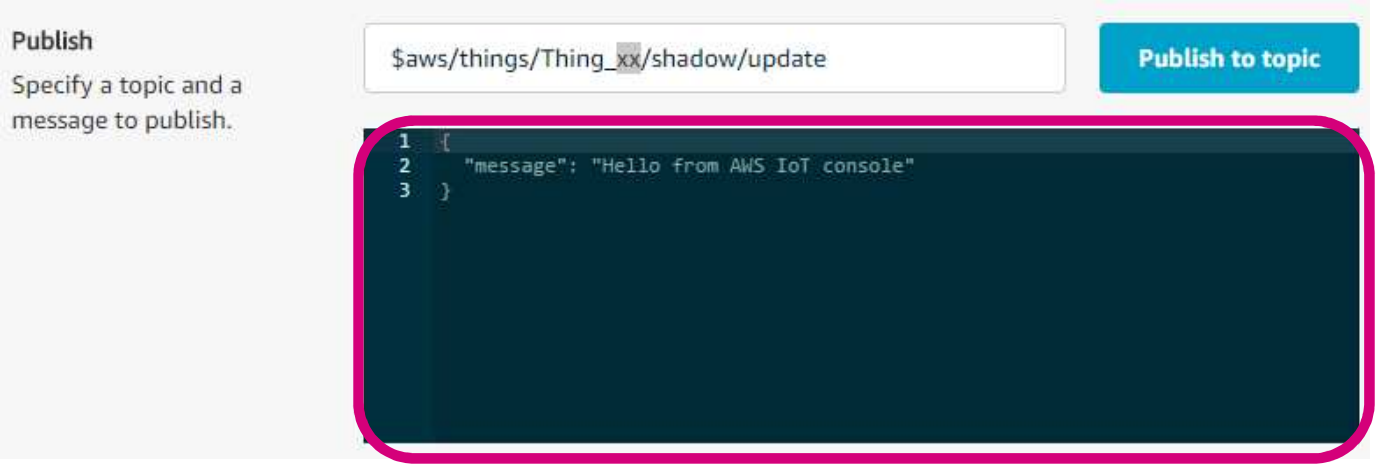
`$aws/things/Thing_xx/shadow/update`

with **`xx`** being your participant number



Interact with your Thing (4/9)

Remove the default message in the dark area.



Interact with your Thing (5/9)

Type the following message:

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

Interact with your Thing (6/9)

Click “Publish to topic” and watch your board’s LED turn ON

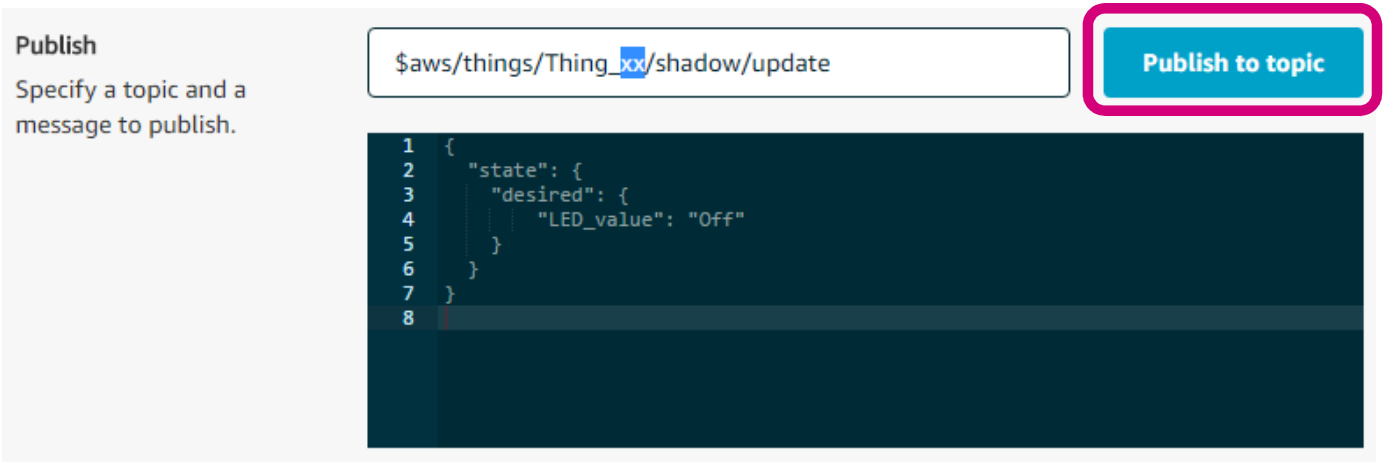
Interact with your Thing (7/9)

Modify the message as follows

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

Interact with your Thing (8/9)

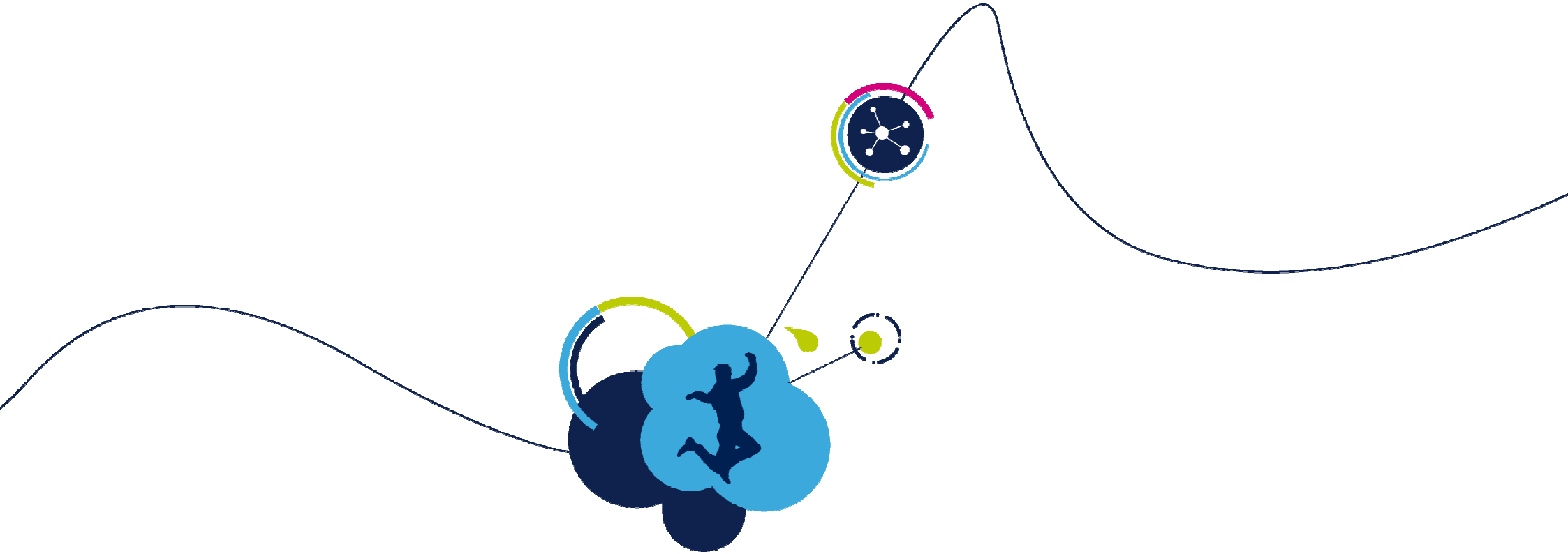
Click “Publish to topic” and watch your board’s LED turn OFF



Interact with your Thing (9/9)

Change the publish topic to interact with another participant's board

`$aws/things/Thing_yy/shadow/update`



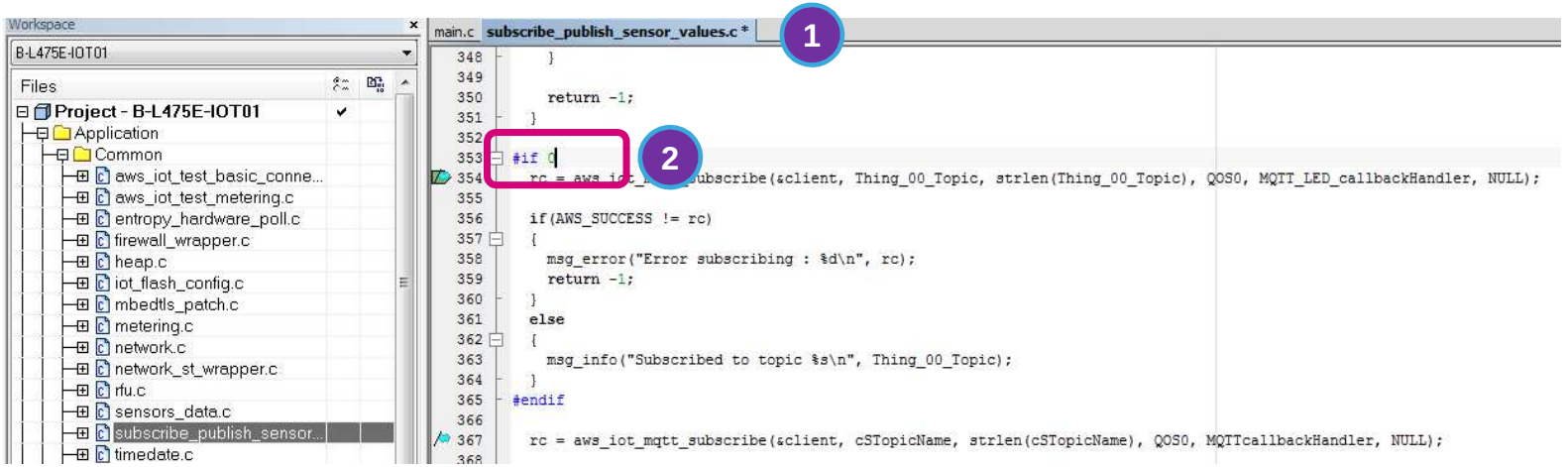
Lab 8 : Connect to a Different MQTT Topic

life.augmented

Change the Subscription Topic

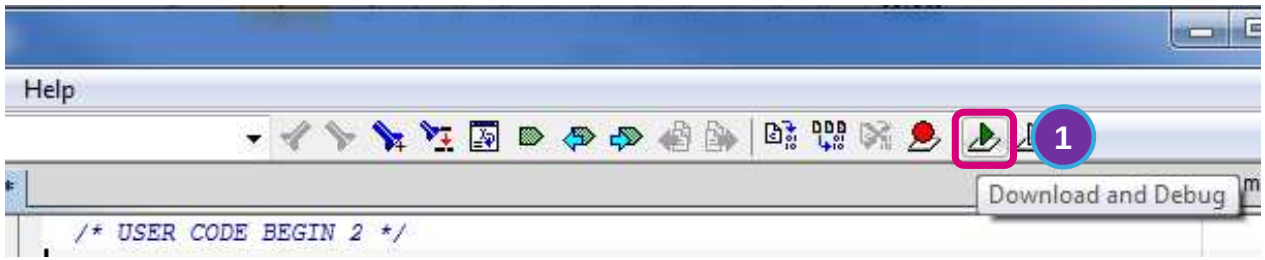
- Now we are going to subscribe to the presenters' publishing topic. The LED on your board will toggle every time the presenter presses the button on his board.

1. Open the **`subscribe_publish_sensor_values.c`** file.
2. Change the `#if 0` to `#if 1 (Line 353)`.

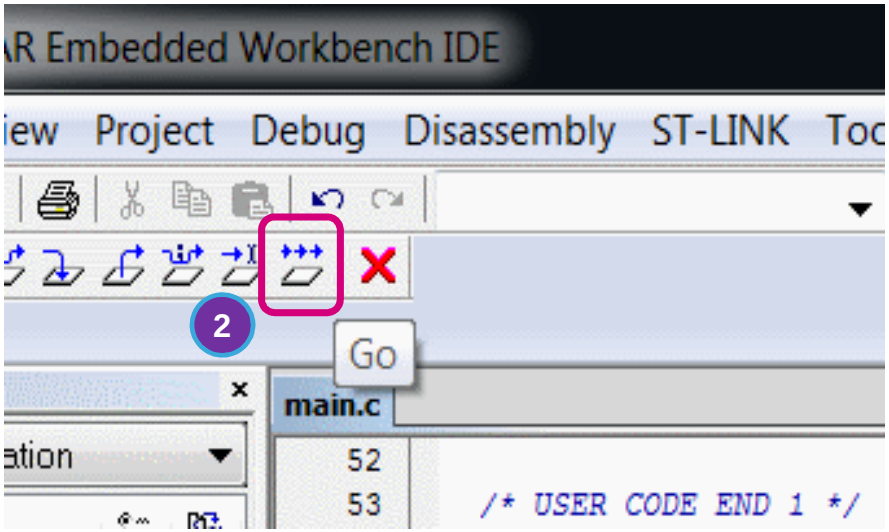


Load and Run

1. Click the GREEN ARROW to Build the Project, **Download** and start the **debugger**. (Ctrl + D)

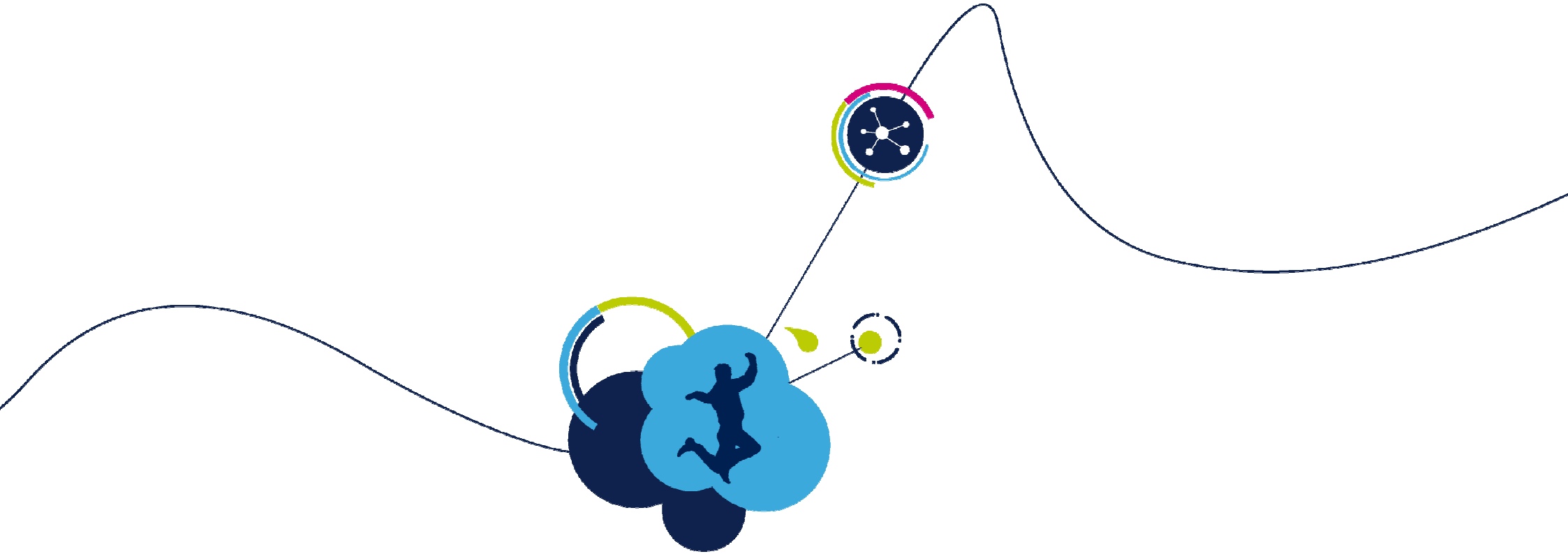


2. Click the triple-arrow **GO** button! (F5)



Verify your AWS connection

1. Open the TeraTerm console.
2. Wait until the DK IoT board **connects to AWS**.
3. The LED on your board will toggle every time the presenter presses the button on his board.



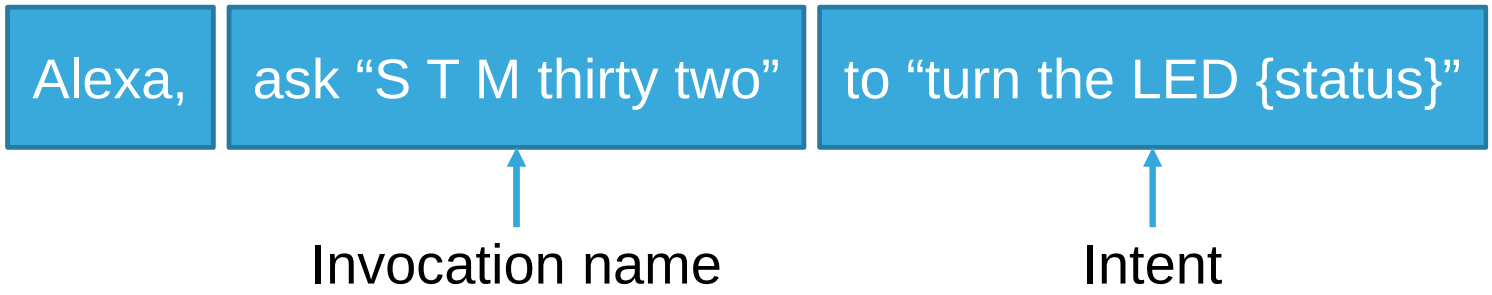
Alexa Voice Demo

life.augmented

What is an Alexa Skill?

- The Alexa skill is based on a speech recognition service that allows you to invoke a pre-specified lambda function.

- The Alexa skill uses an invocation name and intent to call the Lambda function.



What is an AWS Lambda Function

- AWS Lambda is a compute service that lets you run code (function) on AWS servers.

- AWS Lambda executes your code in response to events such as:
  - AWS IoT update to MQTT Topic
  - Alexa skill invocation
  - Other AWS services

Using Amazon Alexa (1/2)

- We will now use Amazon Alexa to voice control the LED on the DK IoT board.

- The presenter will ask Alexa to turn ON/OFF the LED and you will see your board responding to the command:
  - “Alexa, ask STM32 to turn the LED **ON**”
  - “Alexa, ask STM32 to turn the LED **OFF**”
  - This will trigger a Lambda function

Using Amazon Alexa (2/2)

- The Lambda function will send the following JSON message to Thing_00's topic:

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

- Since all the boards are connected to Thing_00's topic (from the previous lab), the message will be sent to your board and toggle the LED.

More Info On Alexa Skills

- Please refer to the following links to get more info on how to use Alexa to interact with IoT devices:
  - https://developer.amazon.com/blogs/post/Tx3828JHC7O9GZ9/Using-Alexa-Skills-Kit-and-AWS-IoT-to-Voice-Control-Connected-Devices
  - https://developer.amazon.com/alexa-skills-kit/alexa-skill-quick-start-tutorial
  - https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/developing-an-alexa-skill-as-a-lambda-function

- Also we have included the Alexa skill code, the Lambda code and a document in the following directory:

C:\STM32L4_DK_IoT_Node_Seminar\Alexa

Releasing Your Creativity

life.augmented

www.st.com/stm32