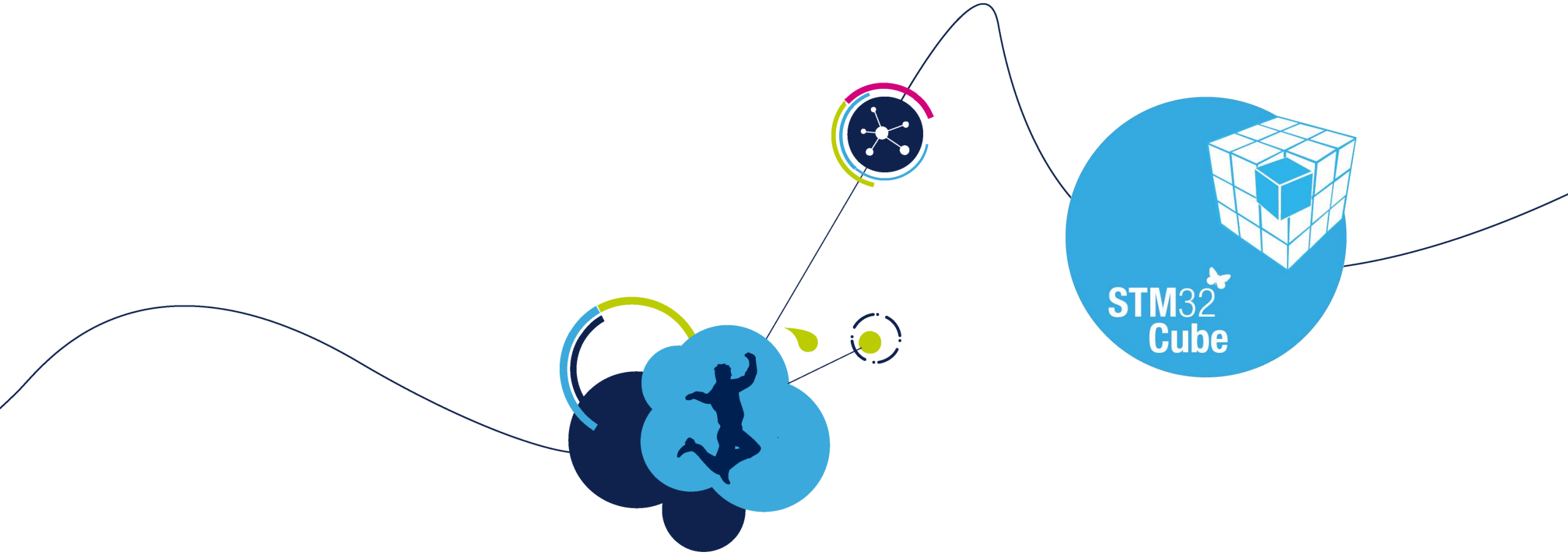
STM32 Ecosystem workshop

T.O.M.A.S Team

life.augmented

- Now it is a right time for some slides
- We will present briefly what is STM32CubeMX and the structure of the Cube library



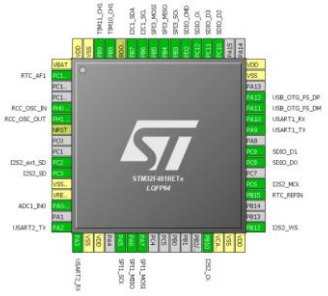
What it is STM32CubeMX ?

life.augmented

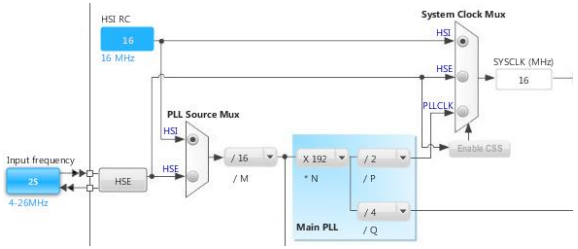
STM32CubeMX application

- STM32CubeMx **do** allow to configure peripherals using GUI, generate project framework and peripheral initialization code.
- STM32CubeMx **do not** allow to create algorithms using GUI and generate algorithms code yet – there is still some space for programmers left ;-)

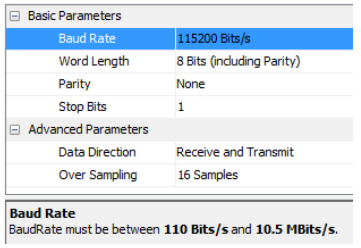
What it is STM32CubeMX ?



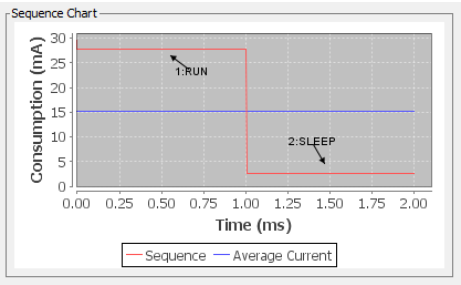
Pinout Wizard



Clock Tree Wizard



Peripherals & Middleware Wizard



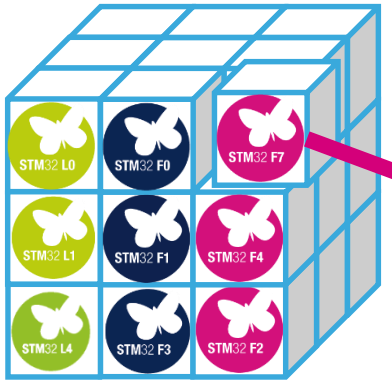
Power Consumption Wizard

STM32CubeMX

Generates Initialization C Code based on user choices !

life.augmented

STM32Cube FW Package Organization



Can be found in STM32CubeMX repository

STM32xxCube HAL package

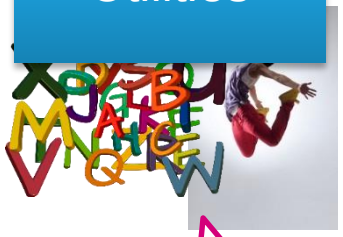
Documents

Drivers

Middleware

Projects

Utilities

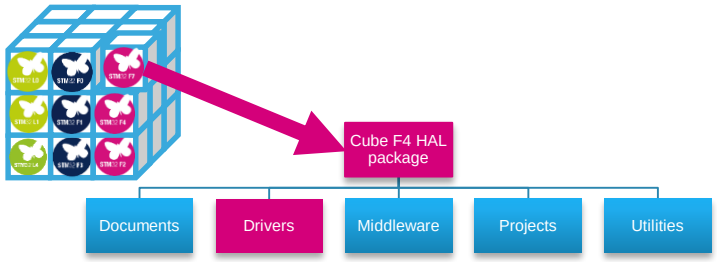


Getting started with STM32F4Cube document

Supporting files like fonts, pictures for graphic examples, …

life.augmented

STM32Cube FW Package Drivers



Drivers

CMSIS

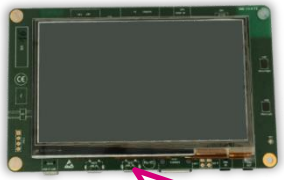
CMSIS COMPLIANT
ARM® Cortex™ Microcontroller Software Interface Standard

Register definitions for Core, startup files, ARM cortex libraries

STM32Fxxx_HAL_Driver

HAL drivers for each periphery in STM32

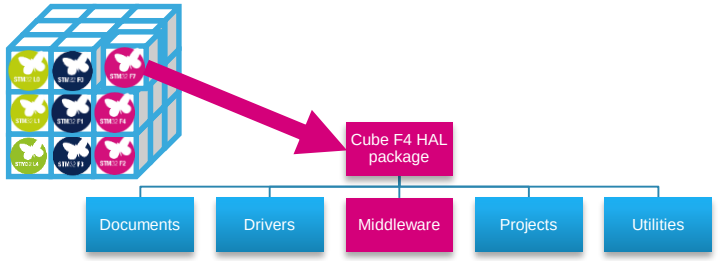
BSP



Functions using HAL drivers to communicate with other components present on EVAL/Discovery boards

life.augmented

STM32Cube FW Package Middleware



Advanced set of libraries

Developed/Owned by ST

Middleware

Third Party libraries

ST

Third Party

STemWin

STM32_Audio

FatFS

FreeRTOS

STM32_USB_Device_Library

STM32_USB_Host_Library

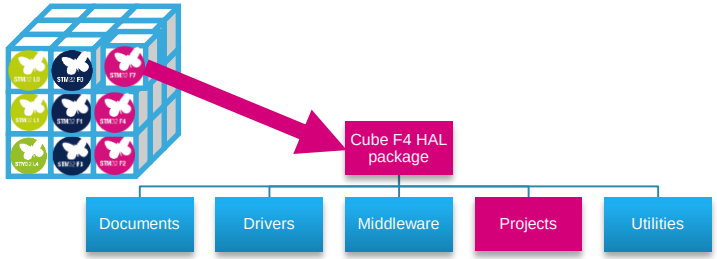
LibJPEG

LwIP

PolarSSL

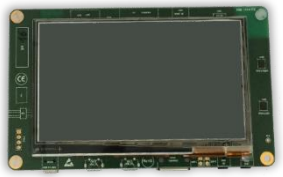
life.augmented

STM32Cube FW Package Projects



Complete projects for STM32 boards

Projects



STM32F-Discovery

STM32Nucleo, Discovery kits, Eval Boards

....

Templates

Examples

Applications

Demonstrations

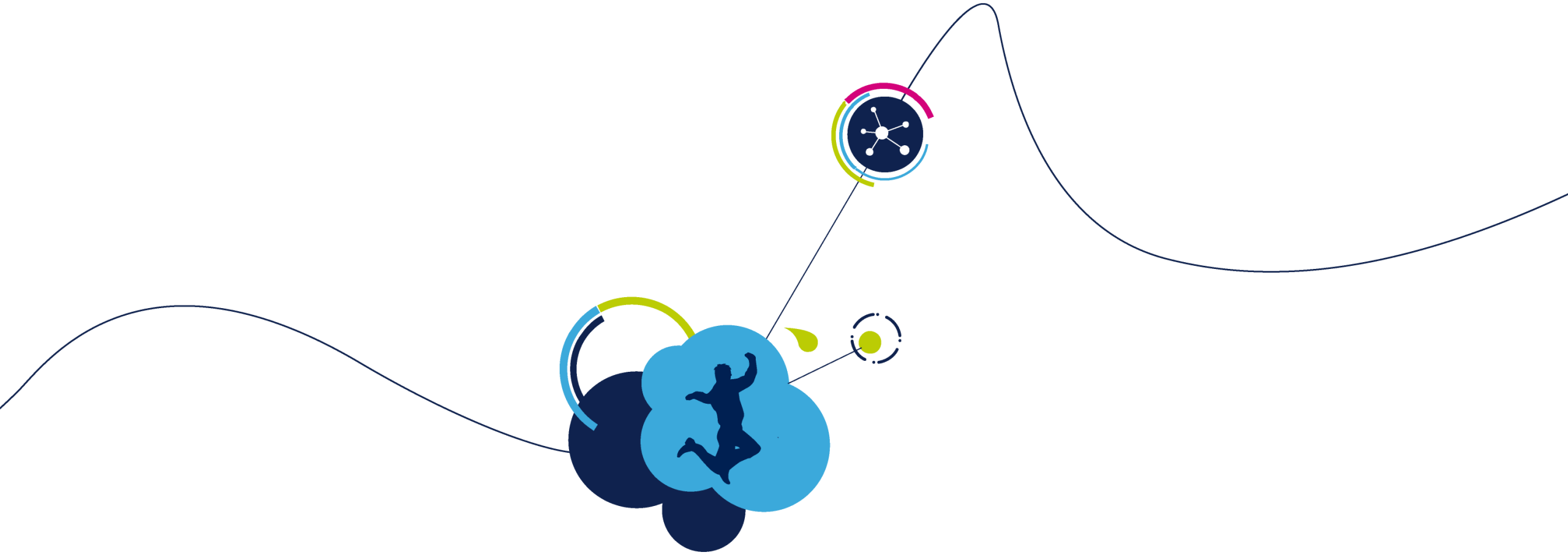
Empty project only with startup file prepared for modification

Simple examples for STM32 Peripheries (GPIO, USART, …)

Advanced examples using Middleware (USB virtual com port)

Demonstration project combining multiple Middleware together

life.augmented

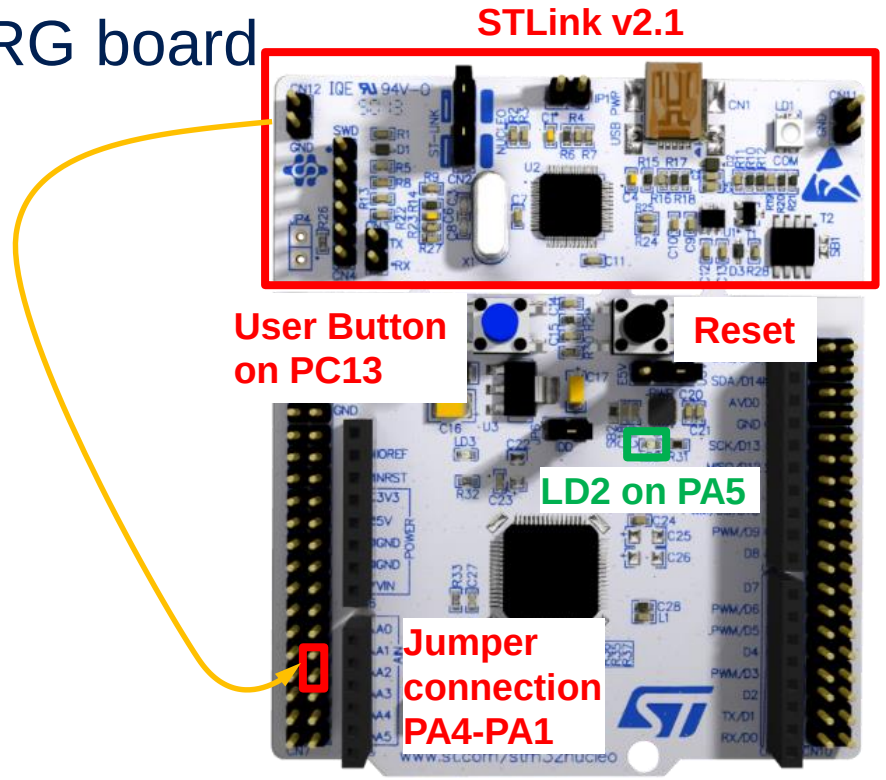


Preparation of the hardware

life.augmented

What we would need ?

- NUCLEO-L476RG board

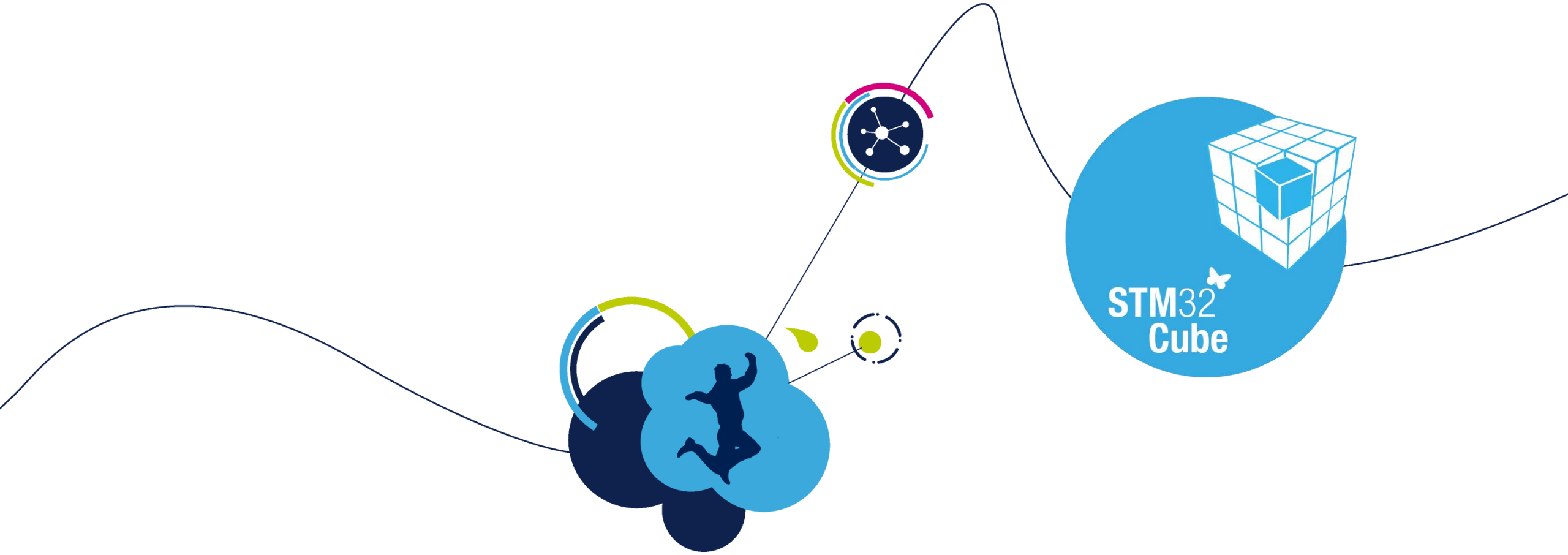


- Mini-USB cable
- PC with preinstalled software (STM32CubeMX, SW4STM32, STLinkv2.1 drivers)

- Now it is a right time for some software activity
- Our first task is to create LED blinking application – just to check whether all the software packs and drivers are installed correctly and whether the hardware is ready for more challenging job

Goal of this part

- To practice a little bit with STM32CubeMX by:
  - MCU selection
  - Play a bit with clock configuration for STM32L4 device
  - Create a skeleton of simple LED blinking application
- Have some fun!



Creating the ‘L4_Blinky’ example in STM32CubeMX

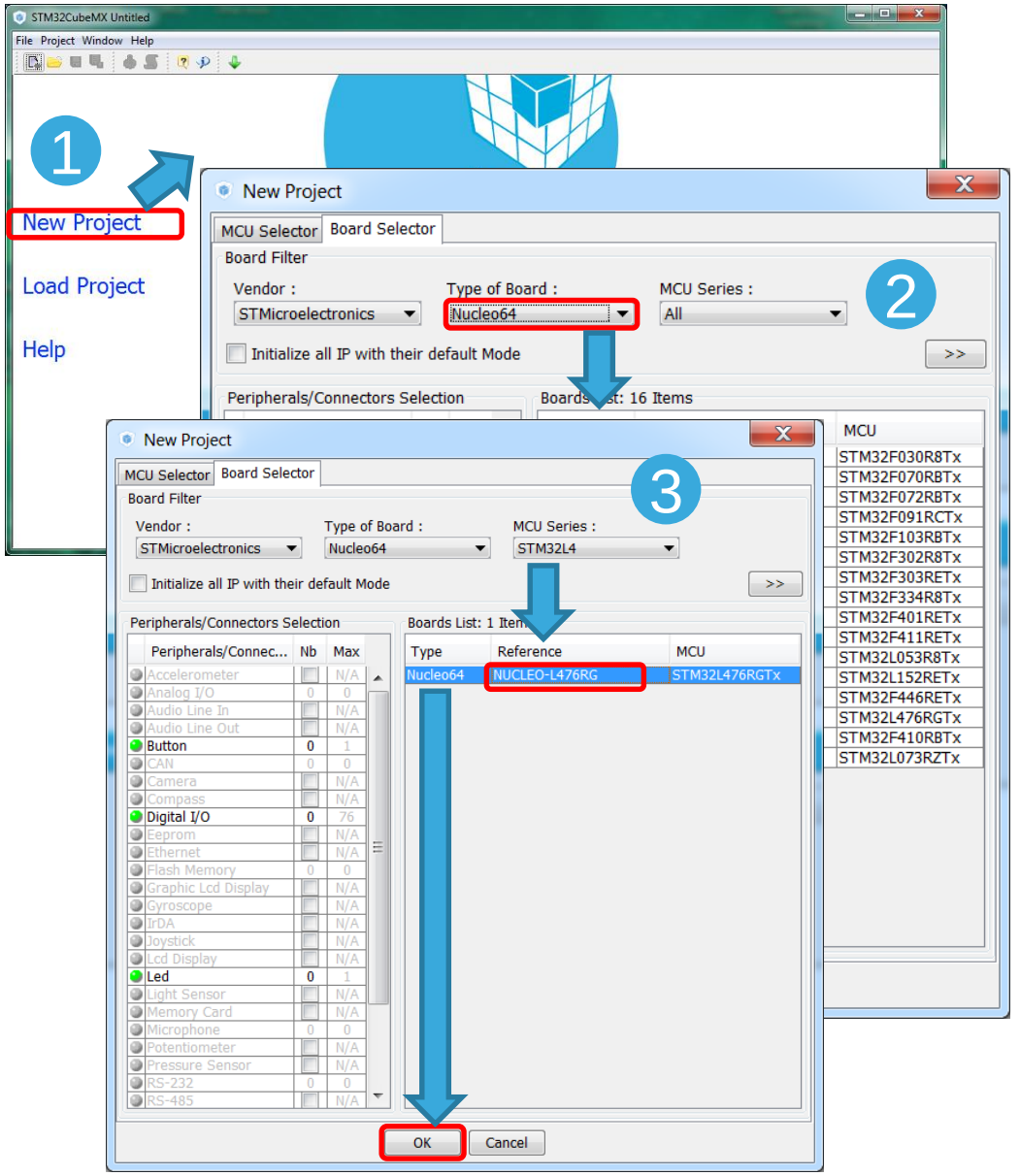
life.augmented

Creating a New Project

1. From the STM32Cube Home Page or **Menu → File** select **New Project**

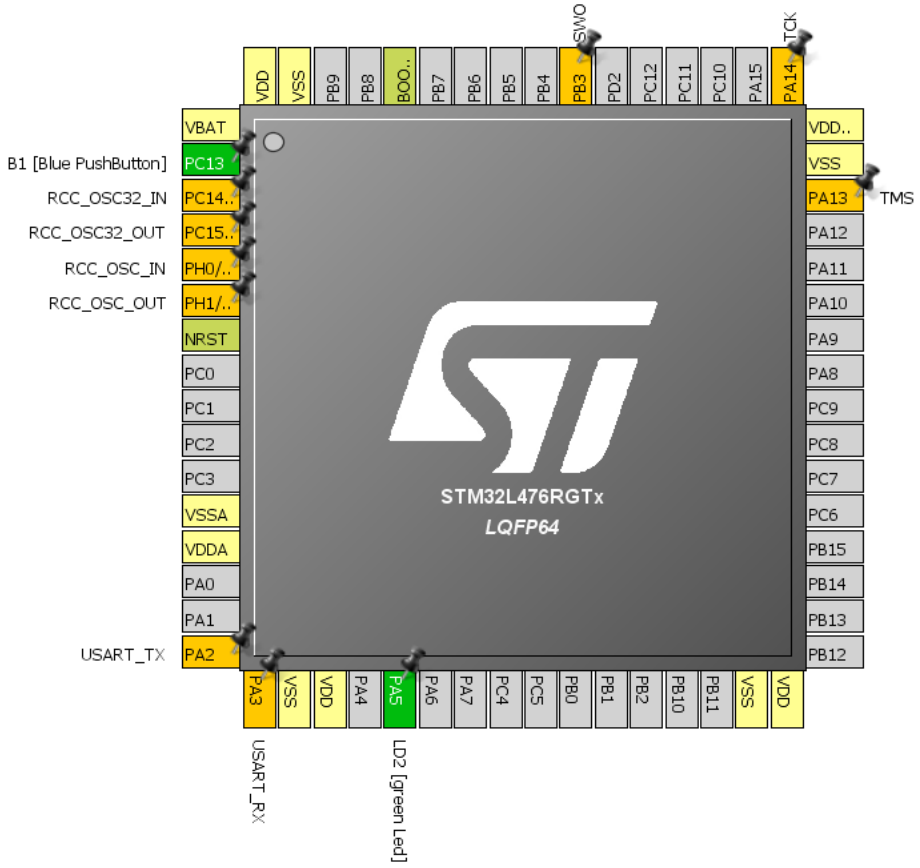
2. There are 3 ways to create a New Project
   - By STM32 Series and Product Line
   - By Peripheral Mix
   - By Board

3. For this example we will use NUCLEO–L476RG
   - Select the **Board Selector** Tab from the top left
   - Select **Type of Board** to be **Nucleo64**
   - Select **STM32L4** in **MCU Series**
   - Select the **NUCLEO–L476RG** from the list on the right
   - Click ‘OK’ to continue



Peripheral and Pin Configuration

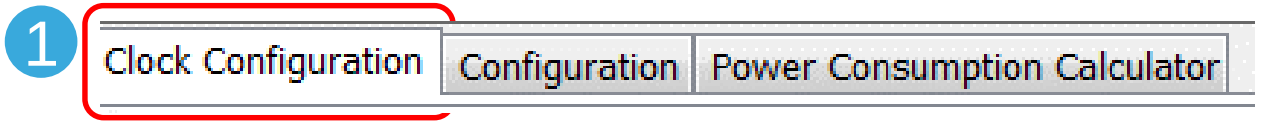
- You will be presented with the pinout of the **NUCLEO–L476RG**
- The debug pins, Push Button and LED are already highlighted in **green**, to say they are connected to the hardware on the board.
- System Pins are highlighted in **yellow/yellow-grey**
- Optional Hardware, like crystals and USART are highlighted in **orange**. This means there are PCB connections but not necessarily any hardware connected by default.
- For the “L4_Blinky” example all relevant peripherals are already connected, so no modifications are needed.



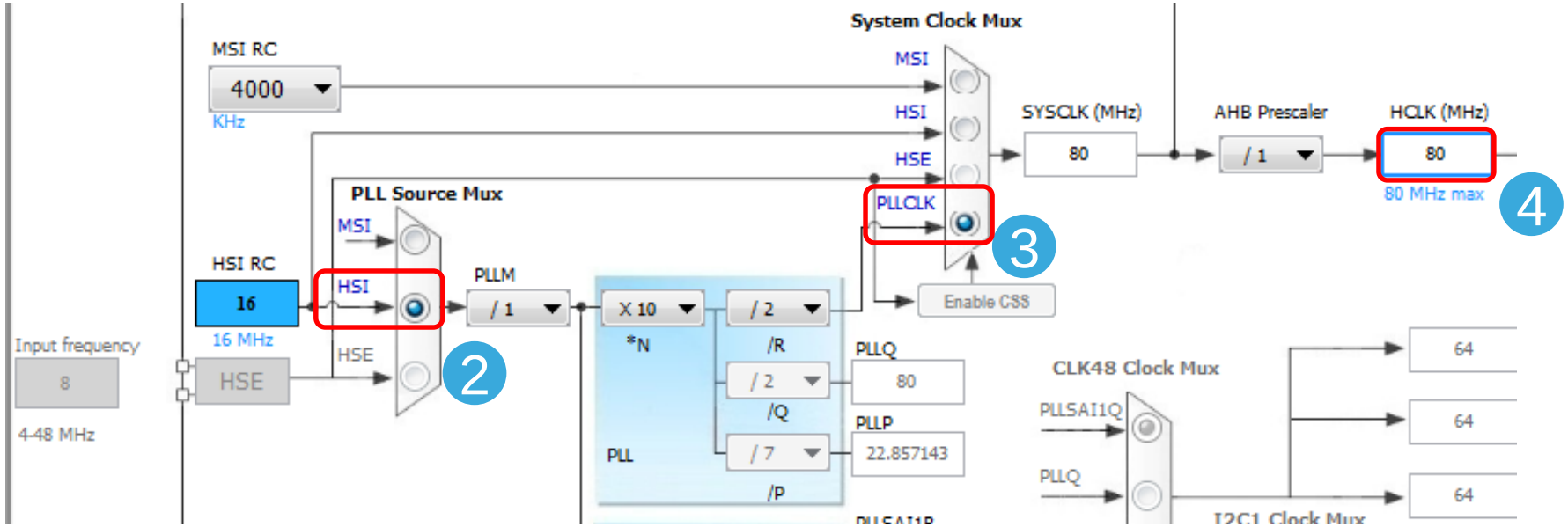
life.augmented

Clock Configuration

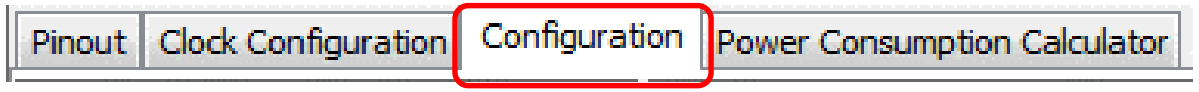
Task: Configure clock system to use internal oscillator with PLL @80MHz



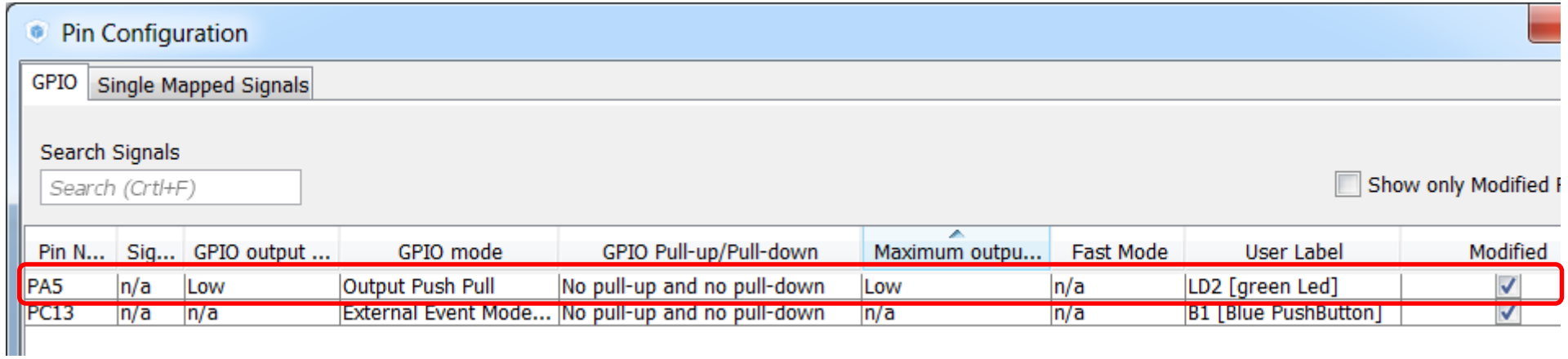
1. Select '**Clock Configuration**' tab
2. Select **HSI** in PLL Source Mux (HSI – High Speed Internal clock)
3. Select **PLLCLK** in the System Clock Mux
4. Set HCLK to **80** and press ENTER – application will propose PLL configuration to match this requirement



Peripheral Configuration

- Select '**Configuration**' tab 

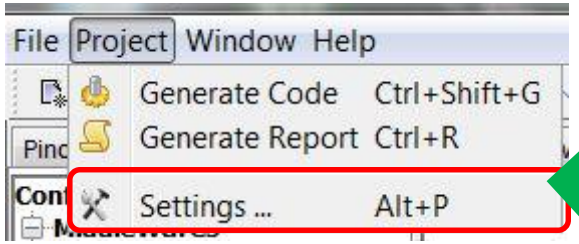
- In this section peripherals with no physical pins or middleware can be added to the project
- For the 'L4_Blinky' example no additional configuration is required as LED is already configured in **GPIO** link as **Output Push-Pull**.



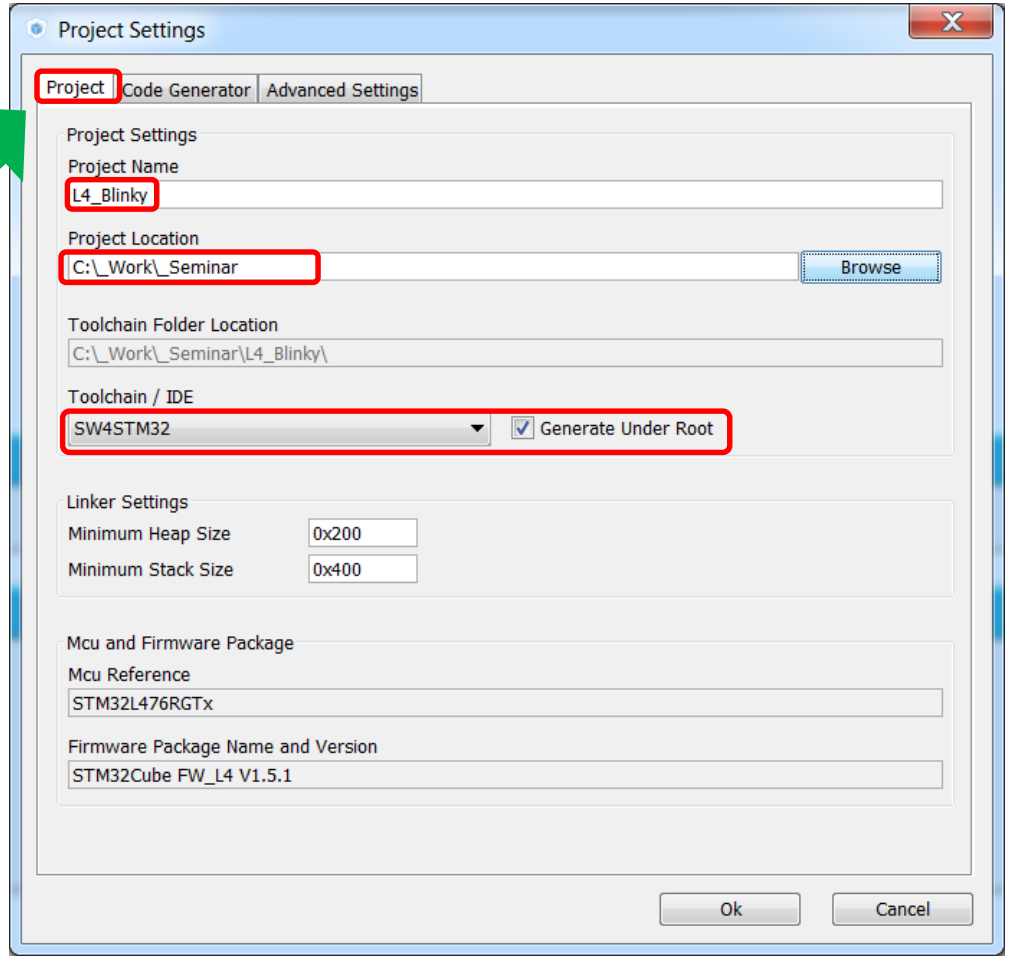
life.augmented

Configure the code generator 1/2

- Open project setings:
  - Menu → Project → Settings



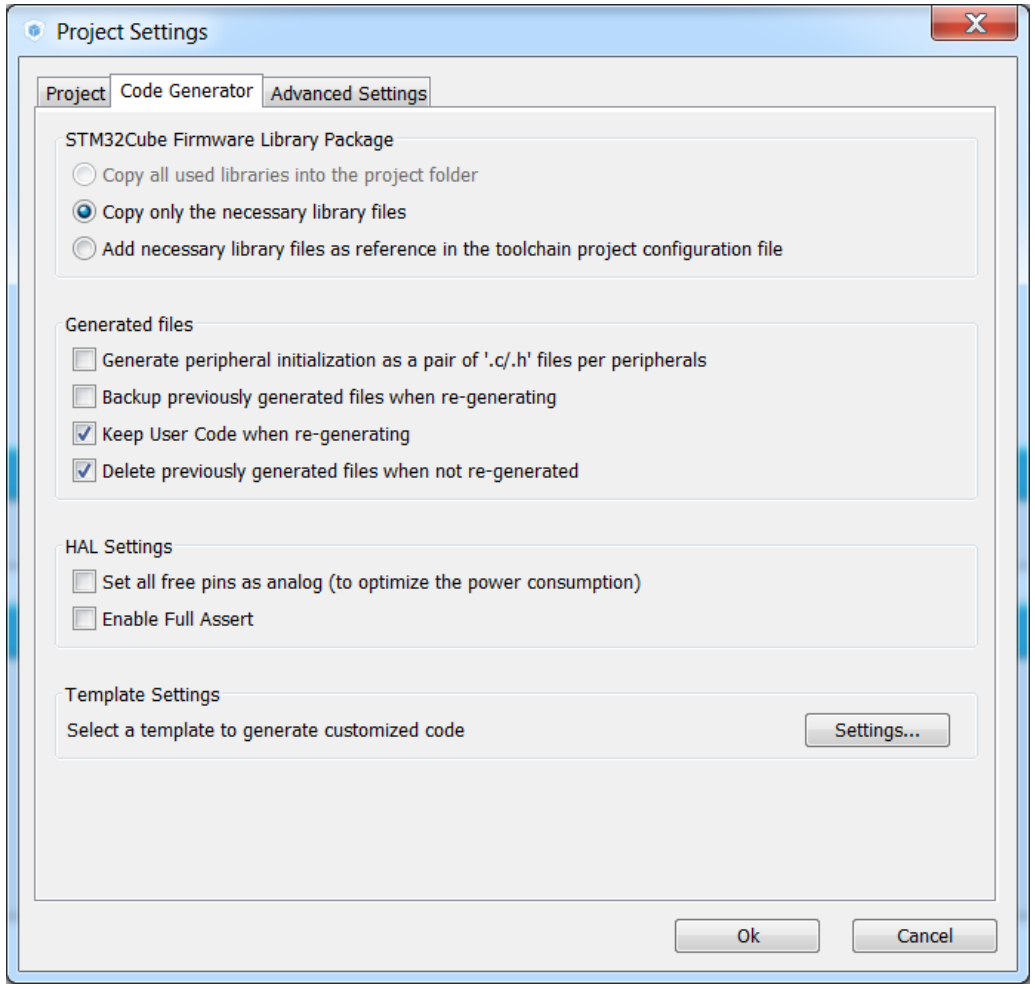
- Under **Project** tab:
  - Give the project a name and location (i.e. **L4_Blinky**)
  - *We strongly recommend to place this folder on the root of 'C:' as some C-compilers show issues when the build path contains too many characters*
  - Select the toolchain to be SW4STM32



- For better understanding let's review code generation options (**Code Generation** tab) first

Configure the code generator 2/2

- Code generator options
  - Copy either the full library or only the necessary files or just link the files from the common repository
  - Place all peripherals initialization in the **`stm32l4xx_hal_msp.c`** file or one file per peripheral
  - Keep user code or overwrite it (refers to code placed between user code comment sections)
  - Delete or keep files that are not used anymore
  - Set unused pins as analog to keep consumption low (**if SWD/JTAG is not selected in pinout, this option will disable it**)
  - Enable full assert in project, this helps to discover incorrect HAL function parameter used in user code

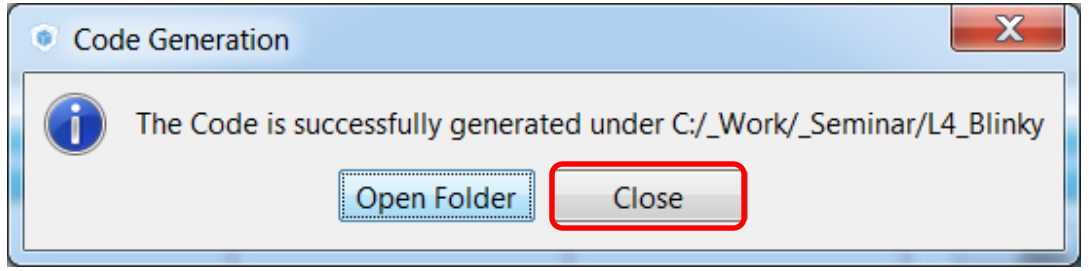


Code generation

- Once we have configured the code generator, we can generate code for selected toolchain.

- There are 3 ways to do it, namely:
  - Clicking  icon
  - Pressing Ctrl+Shift+G keys combination
  - Selecting Project→Generate Code option from menu

- When prompted, click ‘Close’ (we will import this project from SW4STM32 IDE).



What have we gained during this part?

- ✓ Practice a little bit with STM32CubeMX by:
  - ✓ MCU selection
  - ✓ Play a bit with clock configuration for STM32L4 device
  - ✓ Create a skeleton of simple LED blinking application
- ✓ Have some fun!

Further reading

More information can be found in the following document:

- **UM1718 -** STM32CubeMX for STM32 configuration, available on the web:

  http://www.st.com/resource/en/user_manual/dm00104712.pdf

Enjoy!

www.st.com/mcu

life.augmented