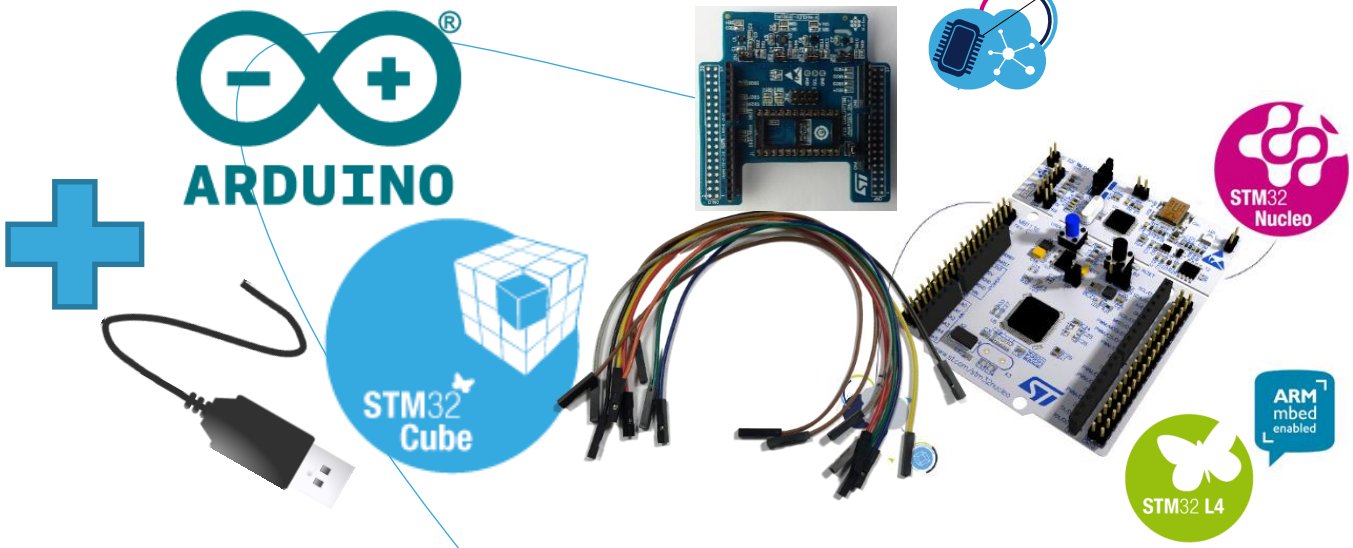
STM32
Arduino
Ecosystem & sensor workshop



life.augmented

Agenda

- Introduction
  - What is Arduino?
  - Arduino IDE downloading & installing
- Getting started with STM32 in Arduino IDE
  - Adding suport for STM32 in Arduino IDE
  - List of STM32 hardwdare platforms supported by Arduino IDE
  - Selecting of STM32 platform in Arduino IDE sketch
- STM32 Arduino hands-ons
  - Guidelines how to develop applications
  - Hands-on 1: GPIOs
  - Hands-on 2: serial communication
  - Hands-on 3: ADC
  - Hands-in 4 : use of a X-nucleo board
- Summary
  - Benefits of Arduino
  - Drawbacks of Arduino

Agenda

- Introduction
  - What is CubeMX ?
  - What is Atollic ?
- How to select the right product
  - MCU finder presentation
- STM32CubeMx hands-ons
  - Hands-on 1: Blinking a led
  - Hands-on 2: serial communication
    - Use of plotter through SWO
  - Hands-on 3: serial communication with DMA
- Summary
  - Arduino vs CubeMX
- Q&A

Instructions

- Put your cell phone in silent mode
- Be respectful of the others
- Follow the signs
  - Your turn to work when the following sign appears

  - When the following sign appears, you don’t neef to follow the instruction

- Don’t hesitate to ask questions

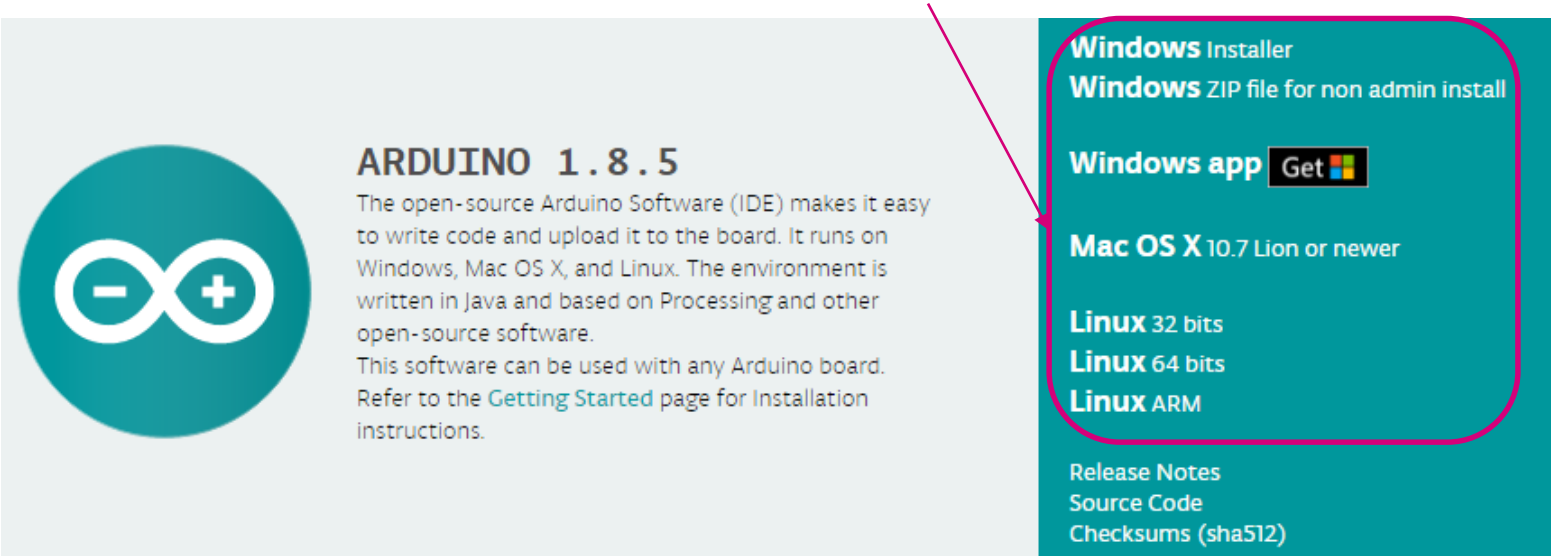
What is Arduino?

A development platform based on easy-to-use hardware and software.

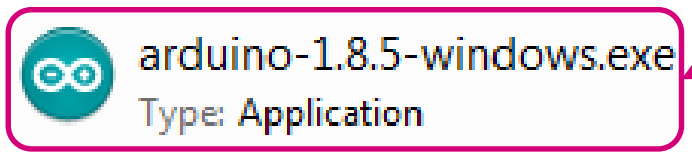
- Hardware
  - **Evaluation kits with a MCUs** from different vendors
  - **Extension shields** with sensors, connectivity and more
- Software
  - **Arduino IDE** – software development platform, which allows to write, code, compile it and upload to the board. It runs on computer (Windows, Linux and Mac OS X are supported)
  - **Arduino Web Editor** – online editor, similar to Arduino IDE, giving the advantage of saving sketches in the cloud and having almost the most up-to-date version of IDE
  - **Arduino libraries** – libraries with API, used by both Arduino IDE and Arduino Web Editor for purpose of application development

Arduino IDE – downloading & installing

- Downloading Arduino IDE from Arduino website
  a) Go to https://www.arduino.cc/
  b) Open tab **SOFTWARE**
  c) Navigate to part of the website related to Arduino IDE
  d) Click on download link for your operating system



- Installing Arduino IDE
  - Double click on downloaded file to start installation



life.augmented

Arduino IDE – user interface

- View of Arduino IDE with description of user interface

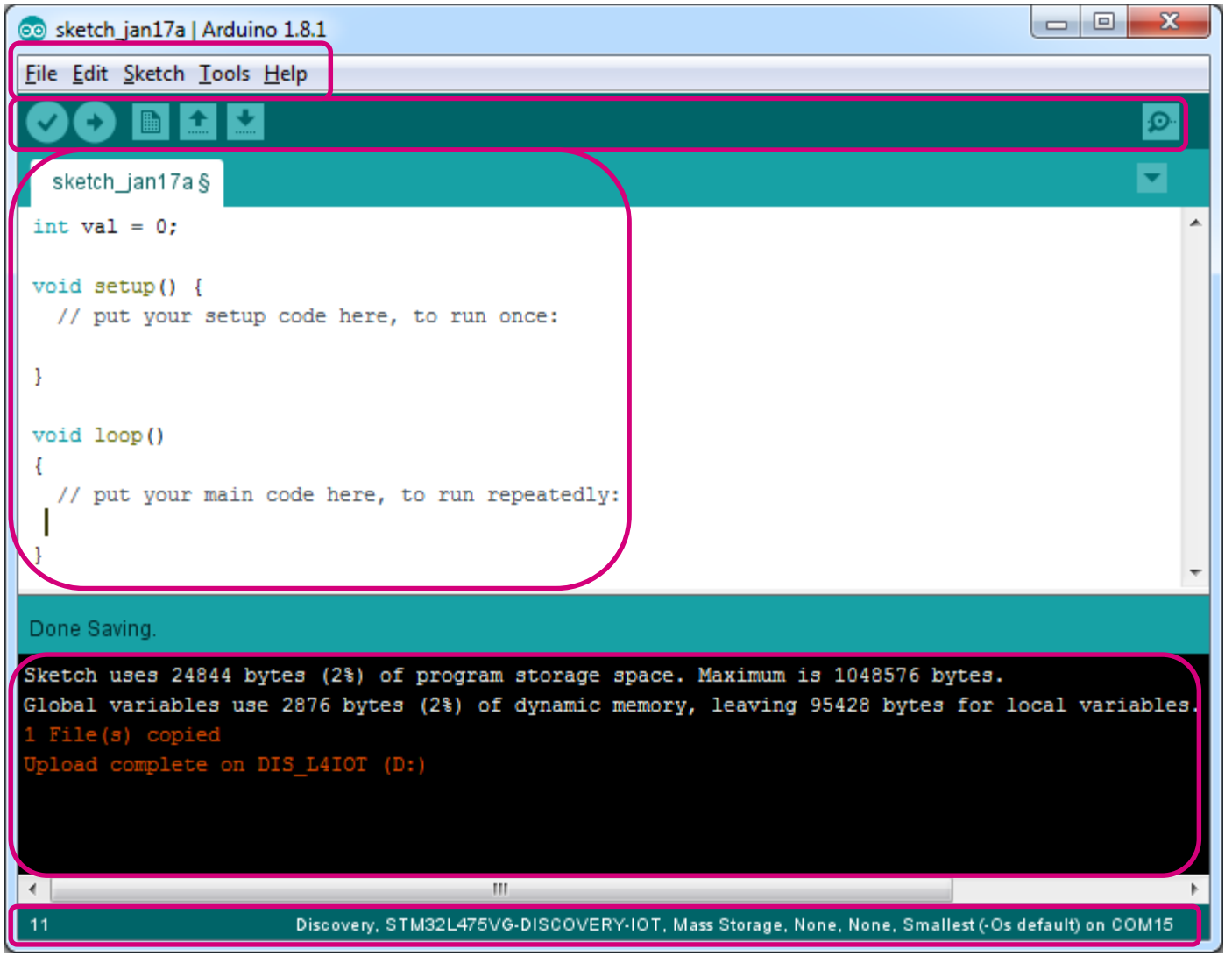
Main menu

Icons for frequently used operations

Editor

Console

Summary of HW/SW settings

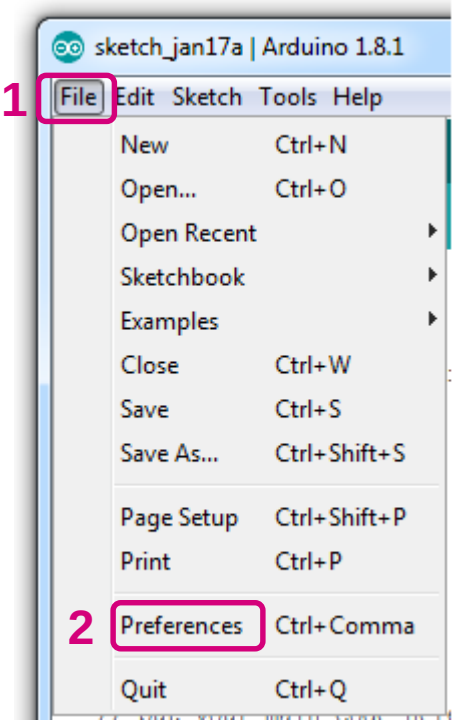


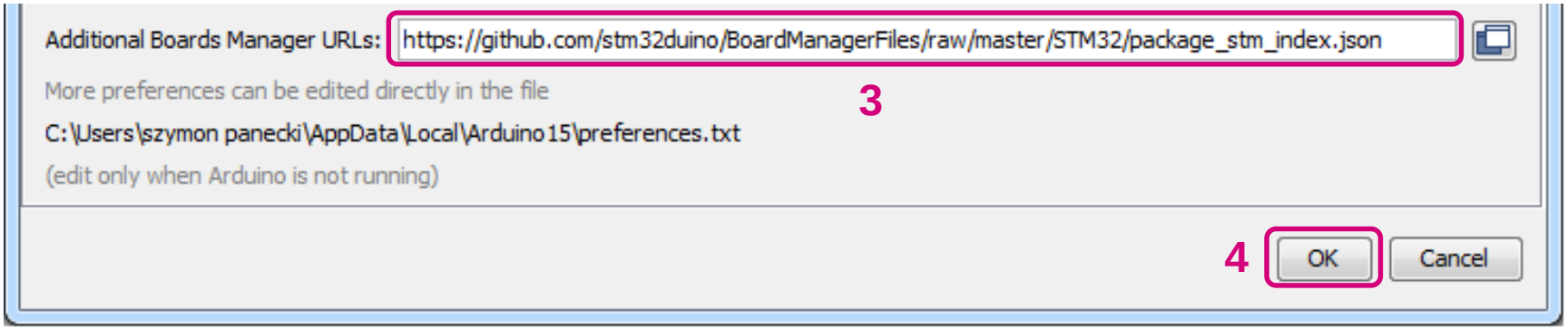
Arduino IDE – adding suport for STM32

- From main menu select **File -> Preferences**

- In **Preferences** window fill in **Additional Boards Manager URL** with link:

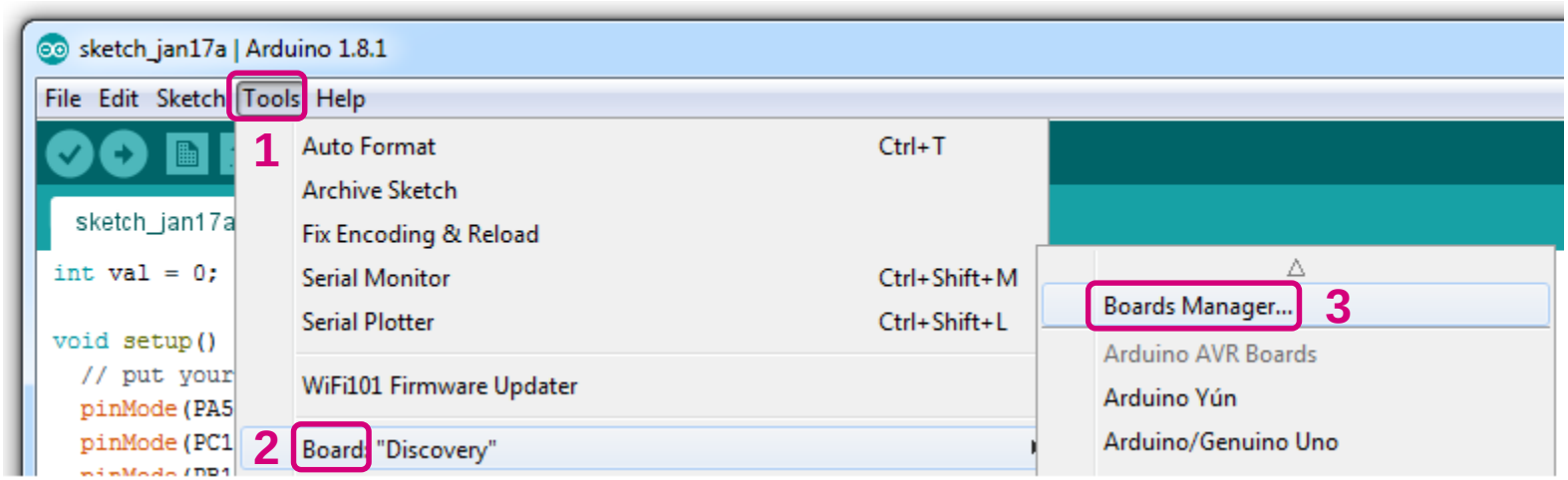
https://github.com/stm32duino/BoardManagerFiles/raw/master/STM32/package_stm_index.json





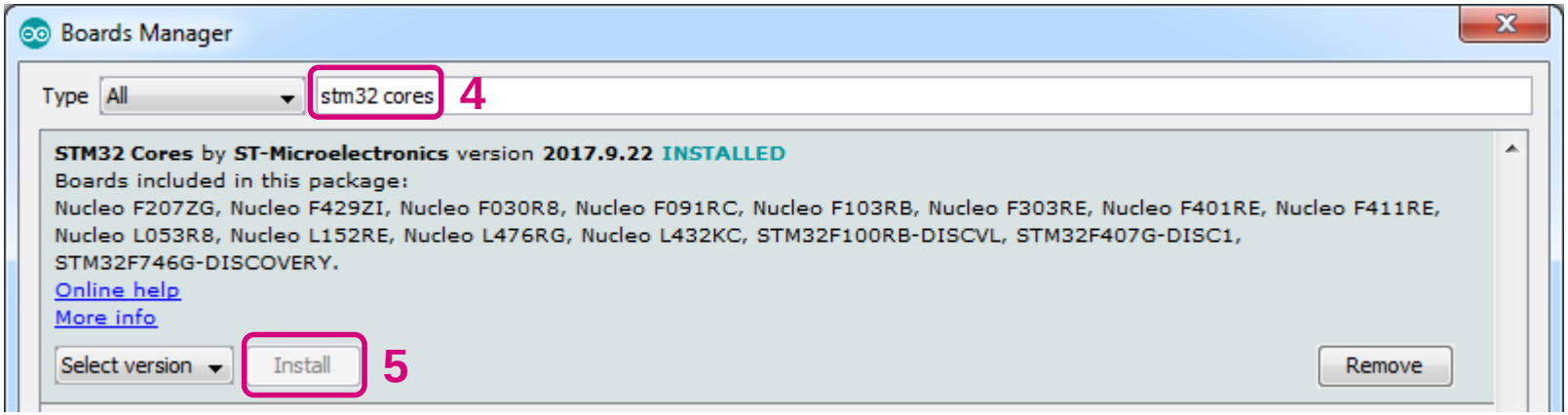
Arduino IDE – adding suport for STM32

- From main menu select **Tools** -> **Board** -> **Boards Manager...**



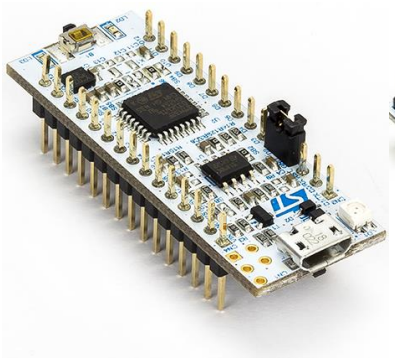
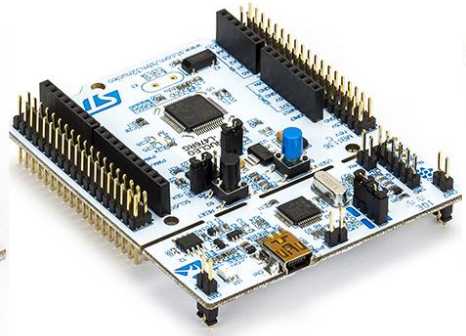
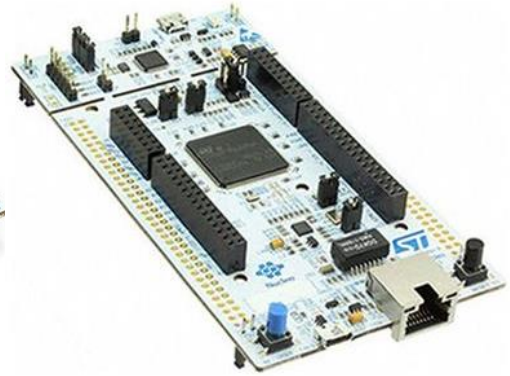
- In the filter item inside **Boards Manager** window type **STM32 cores**

- Select **STM32 cores** package and click on **Install** button



Arduino for STM32: hardware

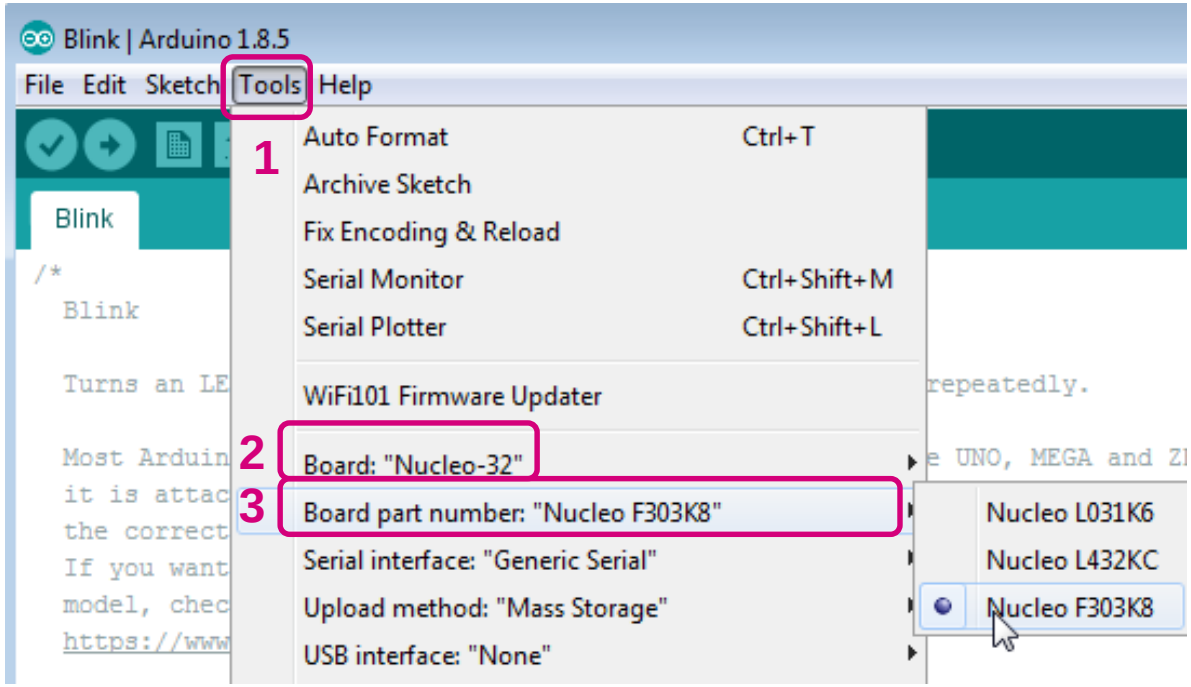
- **STM32 cores** package provides Arduino support for these boards:
  - **STM32F0**: Nucleo F030R8, Nucleo F091RC
  - **STM32F1**: Nucleo F103RB, STM32VLDISCOVERY
  - **STM32F2**: Nucleo-F207ZG
  - **STM32F3**: Nucleo F303RE
  - **STM32F4**: Nucleo F401RE, Nucleo F411RE, Nucleo F429ZI, STM32F407G-DISC1
  - **STM32F7**: STM32F746G-DISCOVERY
  - **STM32L0**: Nucleo L053R8
  - **STM32L1**: Nucleo L152RE
  - **STM32L4**: Nucleo L432KC, Nucleo L476RG, STM32L4 Discovery kit IoT node


Arduino IDE – selecting of STM32 platform

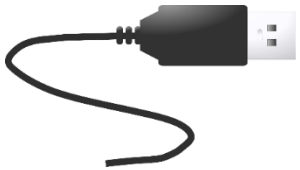
- From main menu select **Tools** -> **Board** and from the list click on one board type, for example **Nucleo-32**

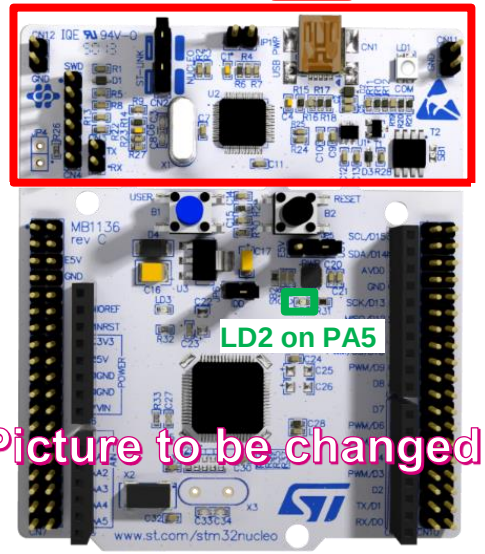
- From main menu click on Tools -> Board part numer and from the list click on one board, for example **Nucleo F303K8**

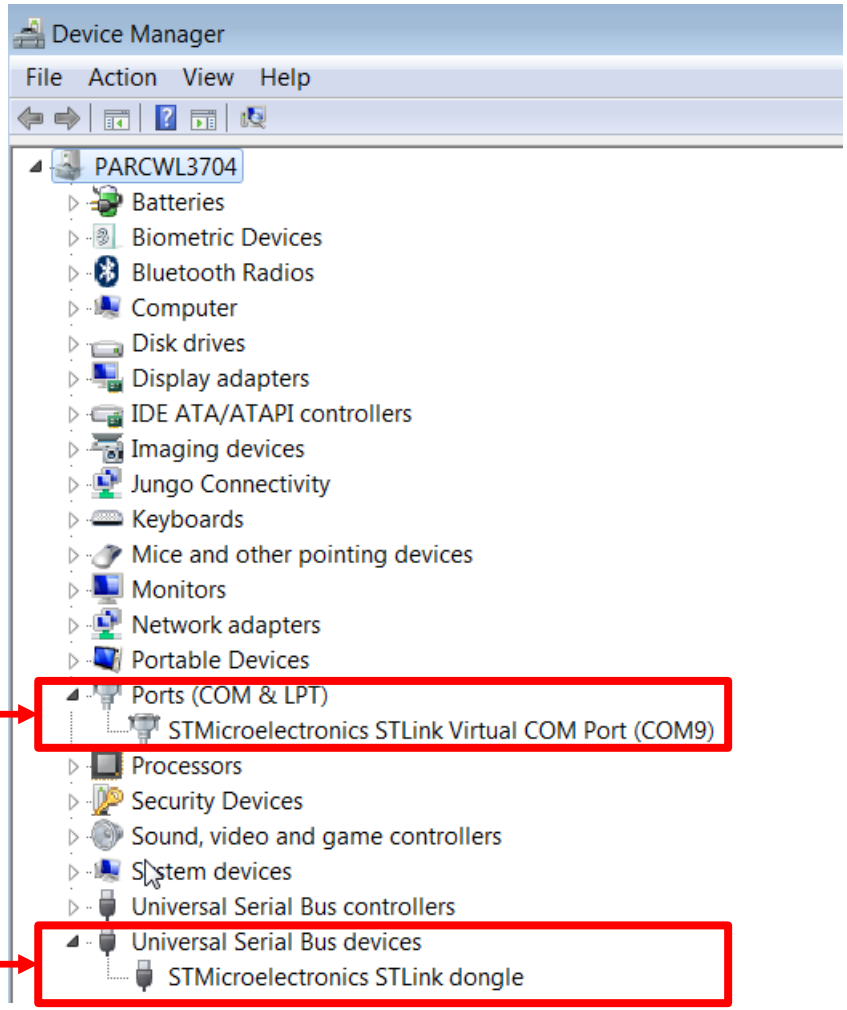


The board we will use

- NUCLEO-F303K8 board
- Mini-USB cable





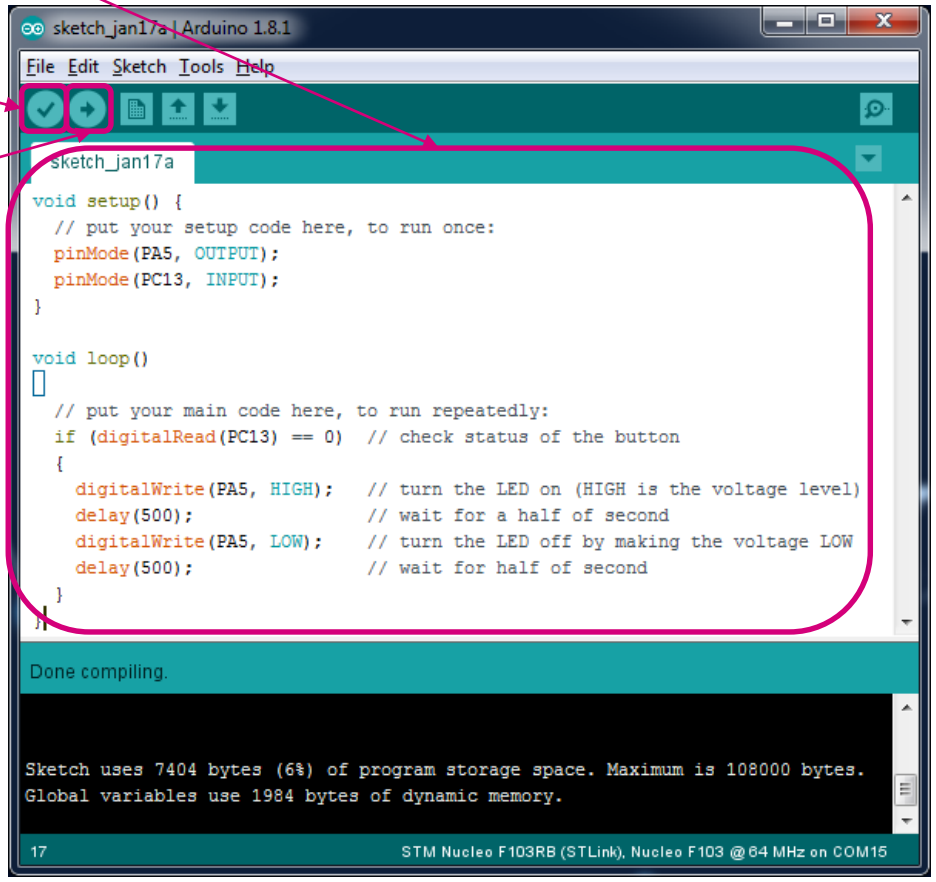


In Windows : 2 USB profile are created
STLINK & Serial comm (over USB)

life.augmented

Arduino IDE – guidelines how to develop applications

- Create a new sketch: from main menu click on **File** -> **New**
- Write user code in editor
- Compile an application by clicking on a ***Verify/Compile*** button
- Upload an application by clicking on an ***Upload*** button
- Watch the application running on hardware platform



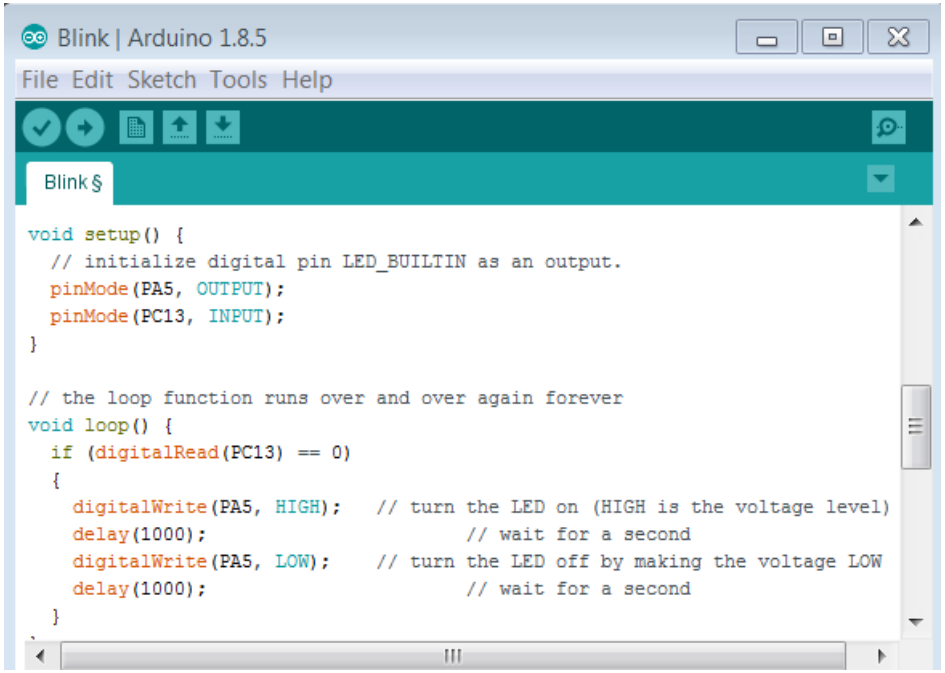
STM32 Arduino application 1: GPIOs
Introduction to functions

- Function **pinMode**
  - Purpose of usage: to configure selected GPIO
  - Arguments: port name and pin number, configuration option
  - Example of usage: pinMode(PA5, OUTPUT);
- Function **digitalWrite**
  - Purpose of usage: to set logical state on output pin
  - Arguments: port name and pin numer, logical state
  - Example of usage: digitalWrite(PA5, HIGH);
- Function **digitalRead**
  - Purpose of usage: to read logical state on input pin
  - Arguments: port name and pin numer
  - Example of usage: digitalRead(PC13);

STM32 Arduino application 1: GPIOs Implementation

- Application: reading of user button (pin PC13) and toggling of LED (pin PA5) when button is pressed

- Code:
  - Call of **pinMode** function to configure PC13 and PA5
  - Call of **digitalRead** function to read state of PC13
  - Call of **digitalWrite** function to toggle PA5
  - Call of **delay** function in milliseconds to make PA5 state visible



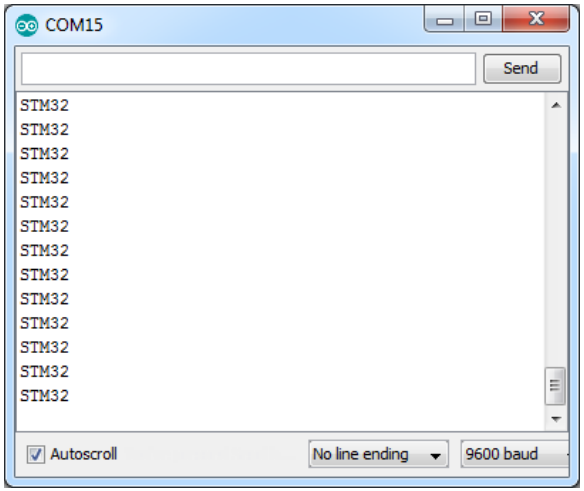
STM32 Arduino application 2: serial comm. Introduction to functions

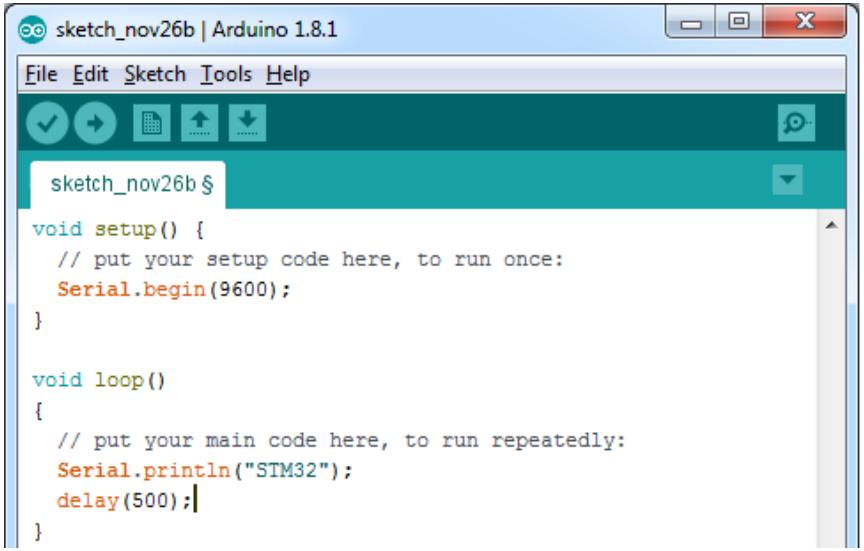
- Function **Serial.begin**
  - Purpose of usage: to configure baudrate of serial interface
  - Arguments: baudrate value
  - Example of usage: Serial.begin(9600);
- Function **Serial.println**
  - Purpose of usage: to send data over serial interface
  - Arguments: data to be sent
  - Example of usage: Serial.println(„Hello world");

STM32 Arduino application 2: serial comm. Implementation

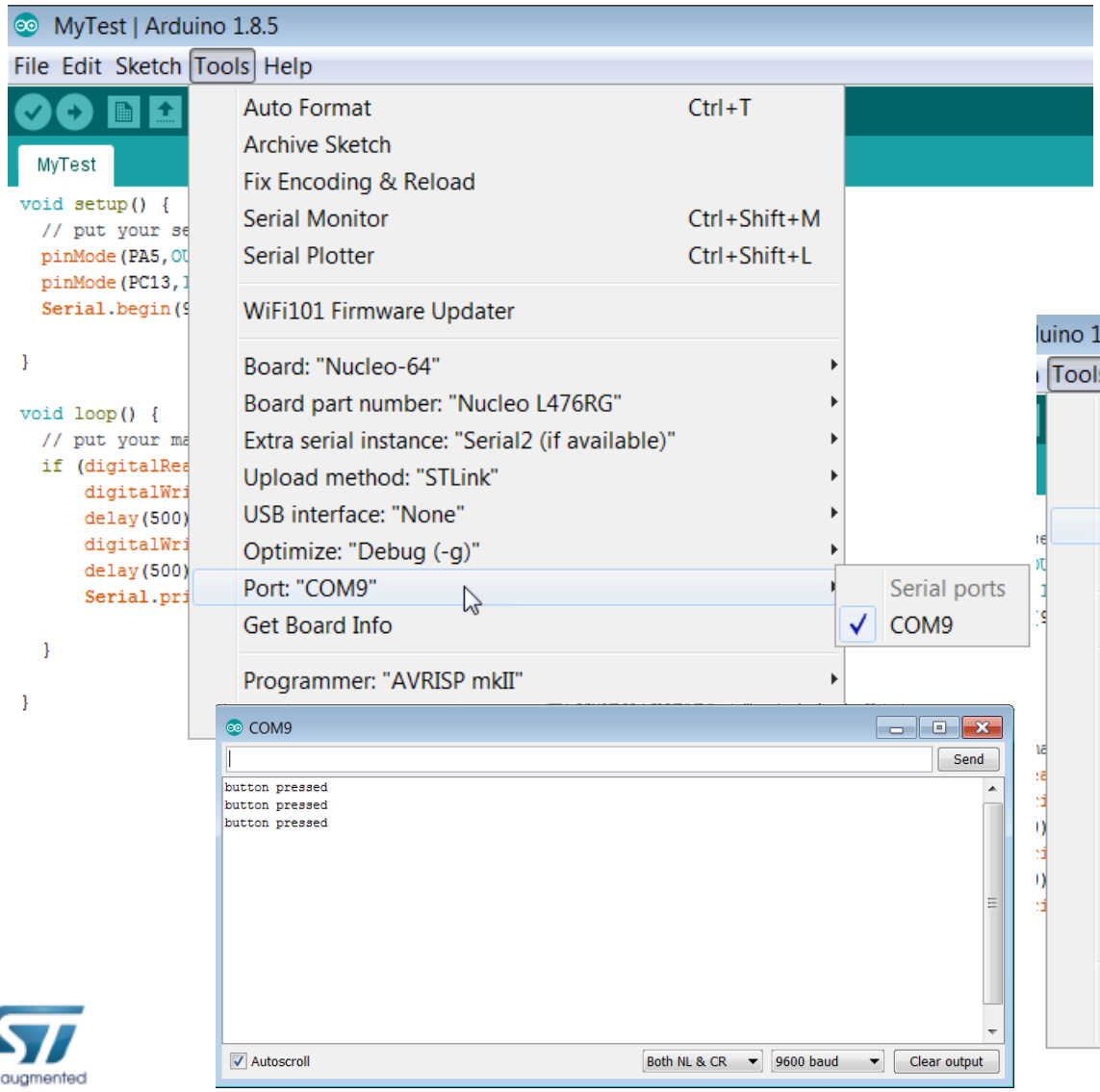
- Application: sending data over serial interface
- Code:
  - Call of **Serial.begin** function to configure baudrate
  - Call of **Serial.println** function to send data
  - Call of **delay** function to make a short brake before sending again the data

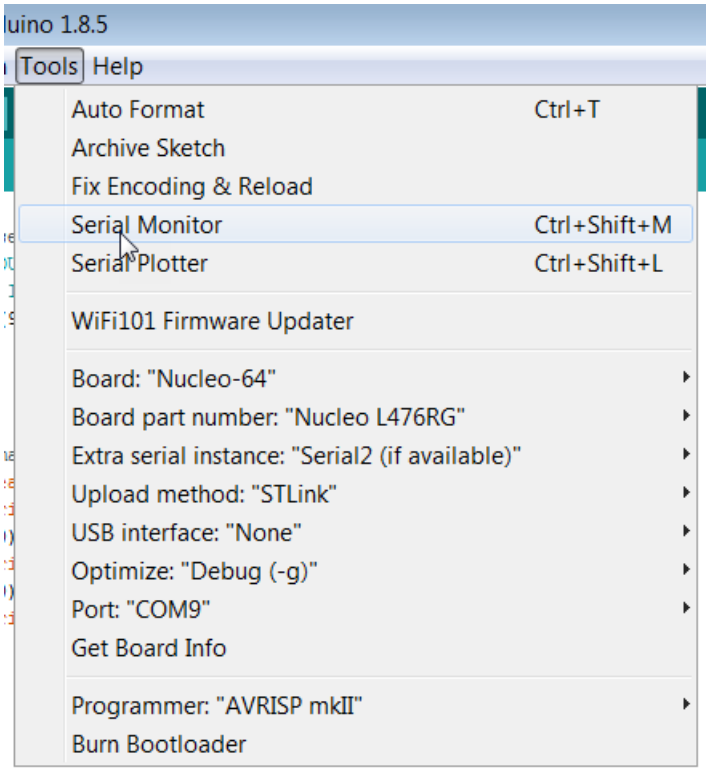
- Additional steps
  - Open **Serial Monitor** (CTRL+Shift+M), which will present data received from the board





Using serial monitor of arduino (CTRL+SHIFT+M)





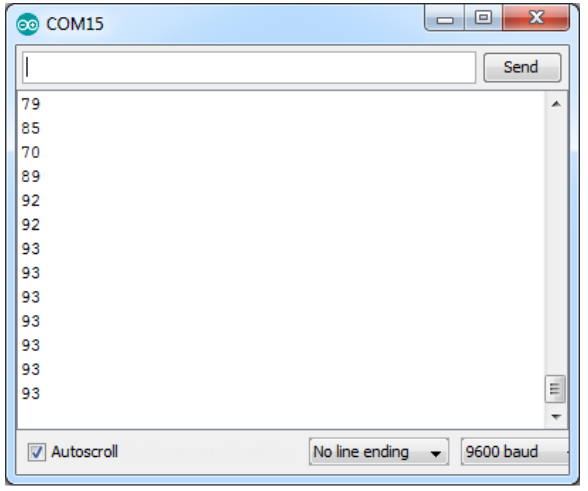
STM32 Arduino application 3: ADC
Introduction to functions

- Function **analogRead**

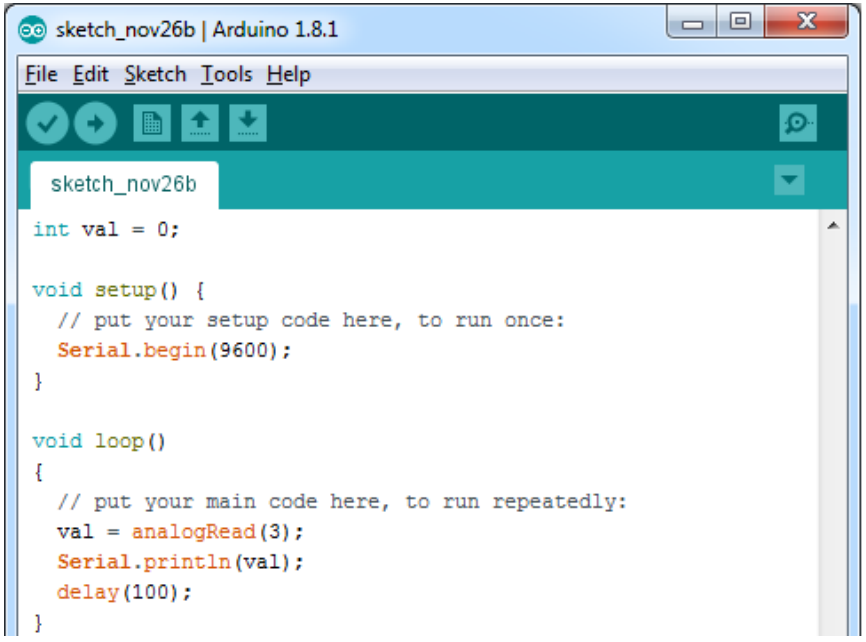
- Purpose of usage: Read voltage value
  - Arguments: analog pin number
  - Example of usage: analogRead(3)

STM32 Arduino application 3: ADC Implementation

- Application: making A-D conversion from the analog pin 3 and sending conversion results over serial interface

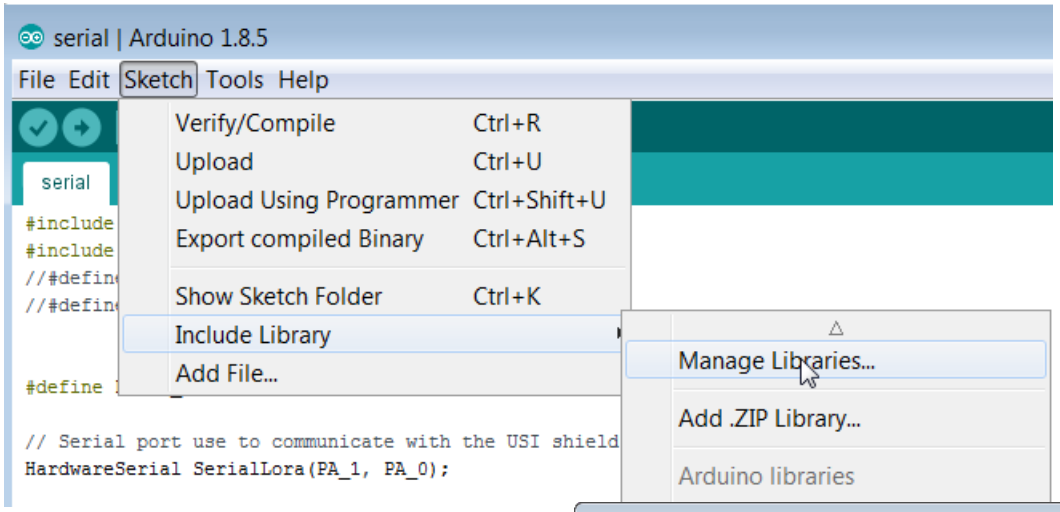


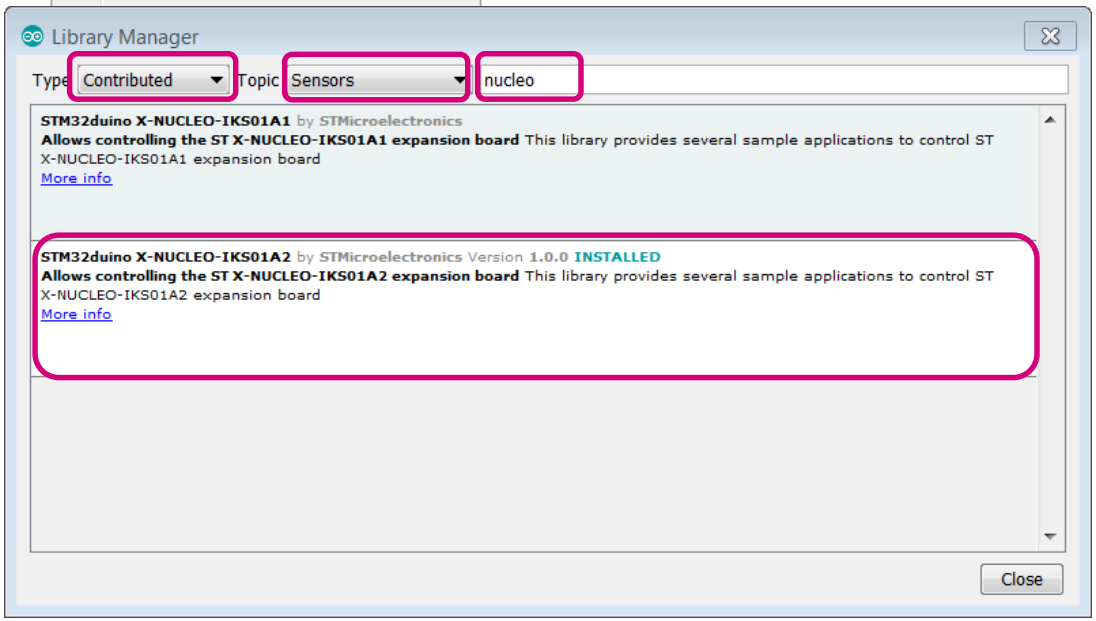
- Code:
  - Define variable, which will store A-D result
  - Call of **Serial.begin** function to configure baudrate
  - Call of **analogRead** function to get value from analog pin 3
  - Call of **Serial.println** function to send data
  - Call of **delay** function to make a short brake before sending again the data



- Additional steps
  - Open **Serial Monitor** (CTRL+Shift+M), which will present data received from the board

STM32 Arduino librairies for **X-NUCLEO-IKS01A2**





- The keyword can be:
  - nucleo
  - STM32duino
  - The name of a sensor
    - HTS221
  - etc…

life.augmented

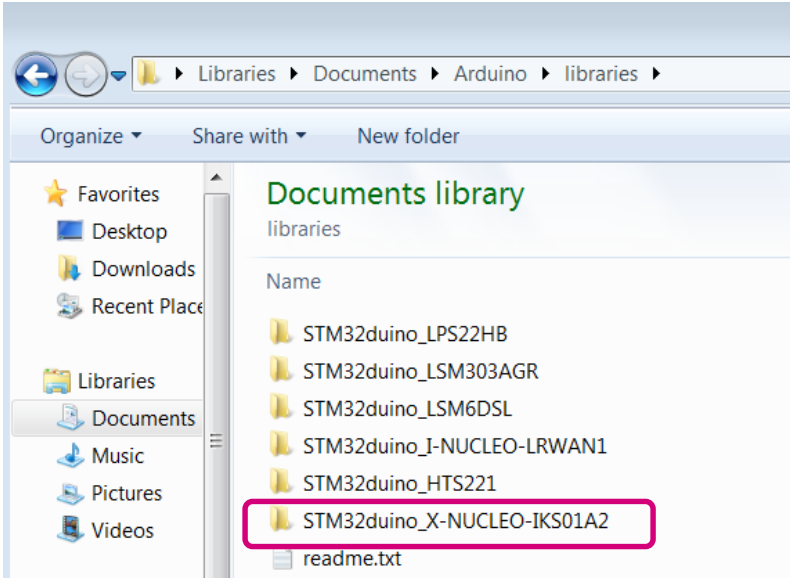
STM32 Arduino extensions for Sensors

1. Open a website with Arduino libraries for STM32:

   http://www.arduinolibraries.info/architectures/stm32

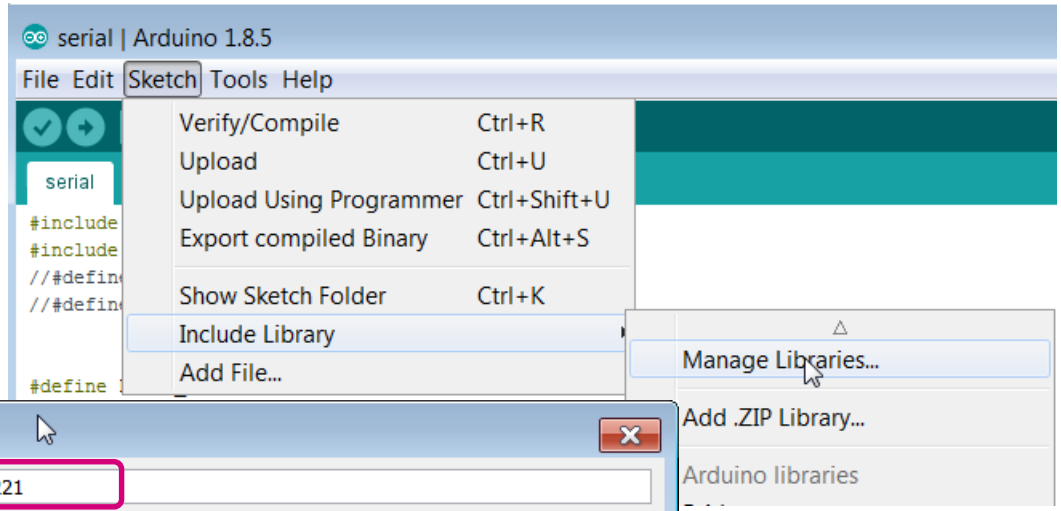
2. Click on **STM32duino X-NUCLEO-IKS01A2** to download library for the **X-NUCLEO-IKS01A2** board

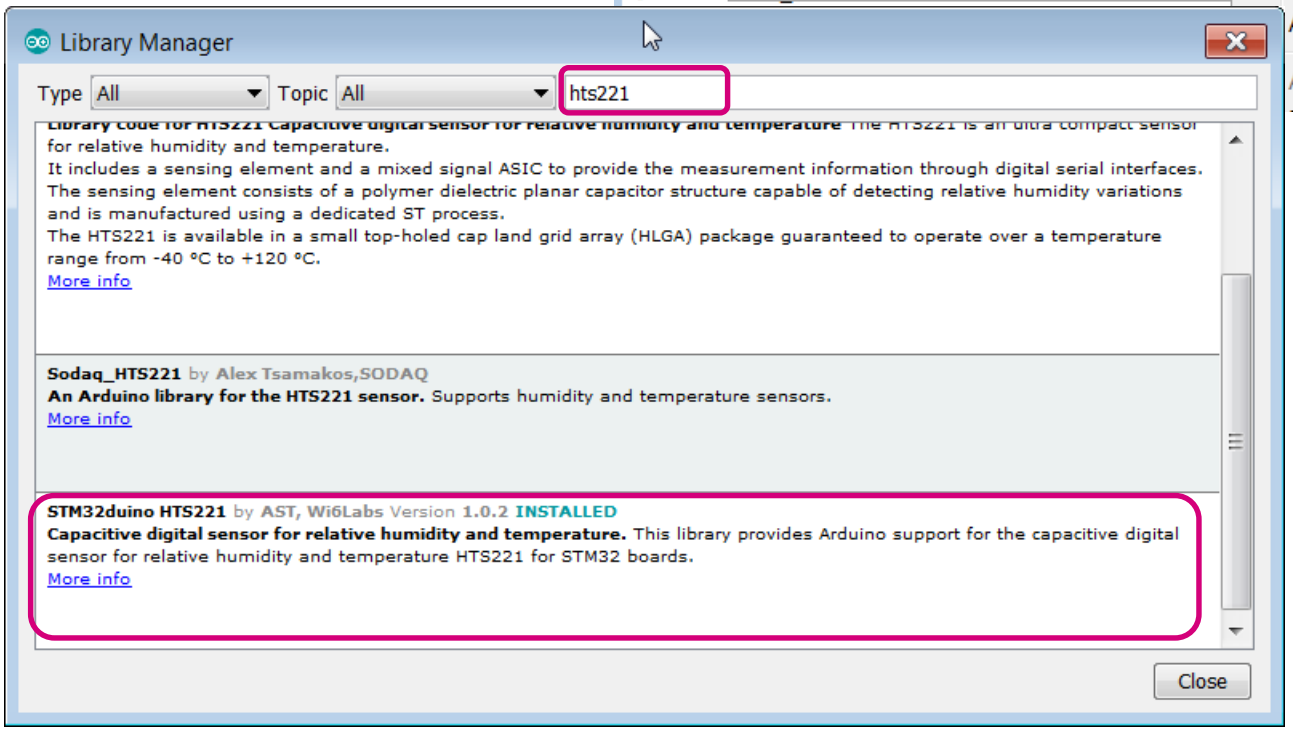
3. In Arduino IDE use main menu and click on **Sketch -> Include Library -> Add .ZIP library…**, then select downloaded library



STM32 Arduino librairies for **HTS221**

- Install **STM32duino HTS221**





Send Humidity and Temperature by UART

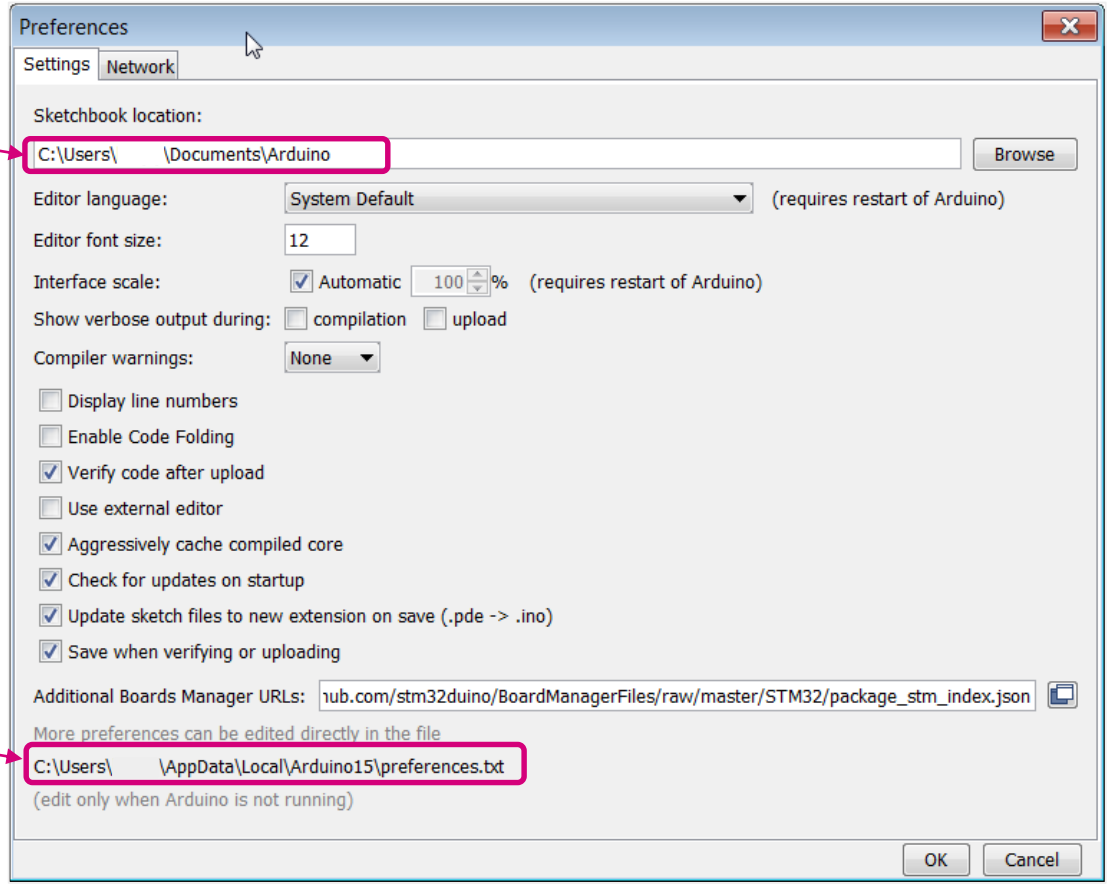
- Select the example in STM32duino HTS221
  - You will have to adapt it as it is developed for a different board
- Where to find the definitions ?
  - Refer to https://github.com/stm32duino/wiki/wiki/Where-are-sources
  - Locate « variant.h » in variants folder
- What you need to look for ?
  - LED_BUILTIN
  - SDA/SCL : they are renamed on the board itself D14/D15
  - You can also look at or double-check with the schematics

Quickly access to the package and libraries

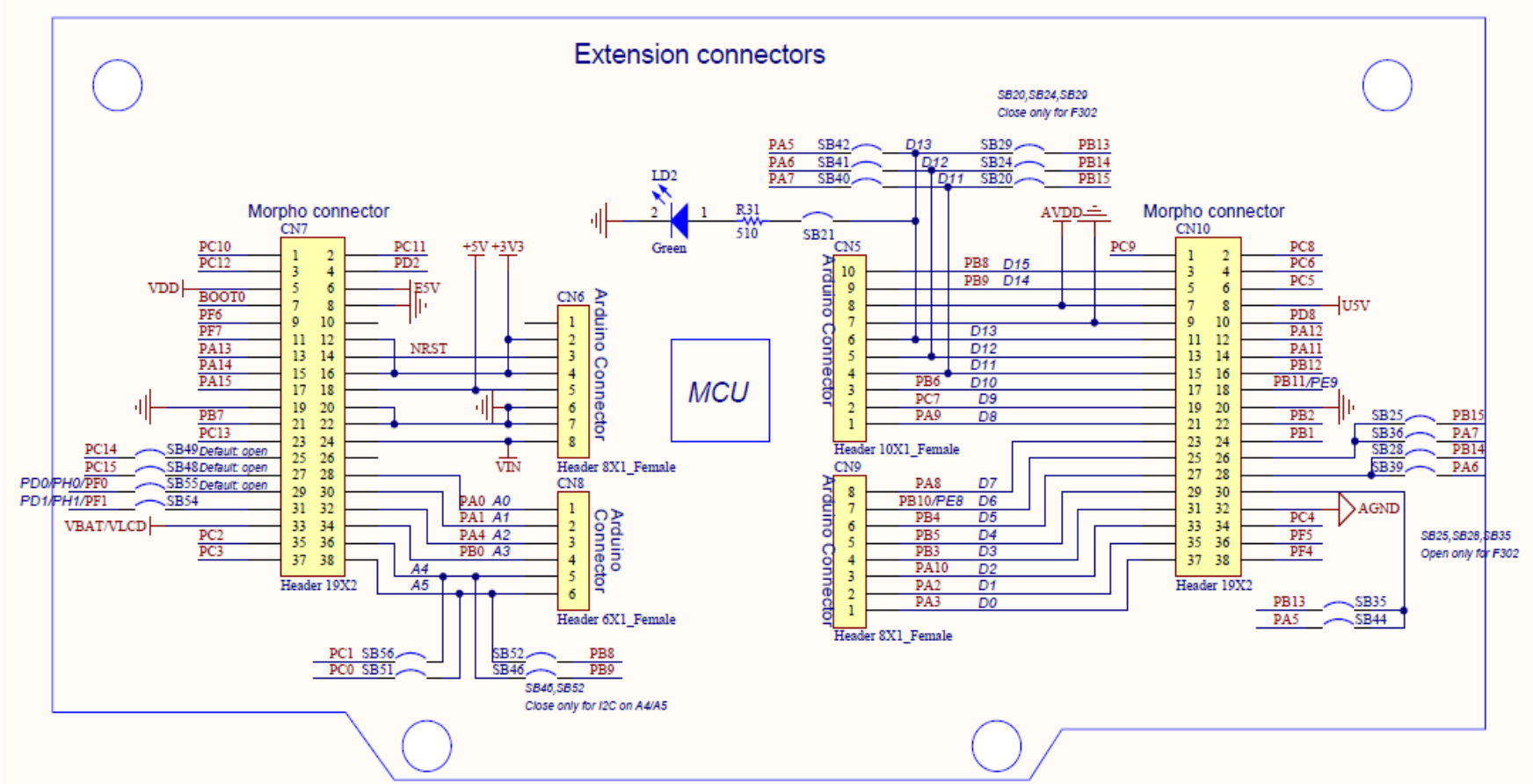
- Open menu File/Preferences

Location of the installed Libraries

Location of the core packages, you can click on the link



Arduino pin out & Morpho connectors



Customize the example

- Simply replace the initialization of the I2C bus with :

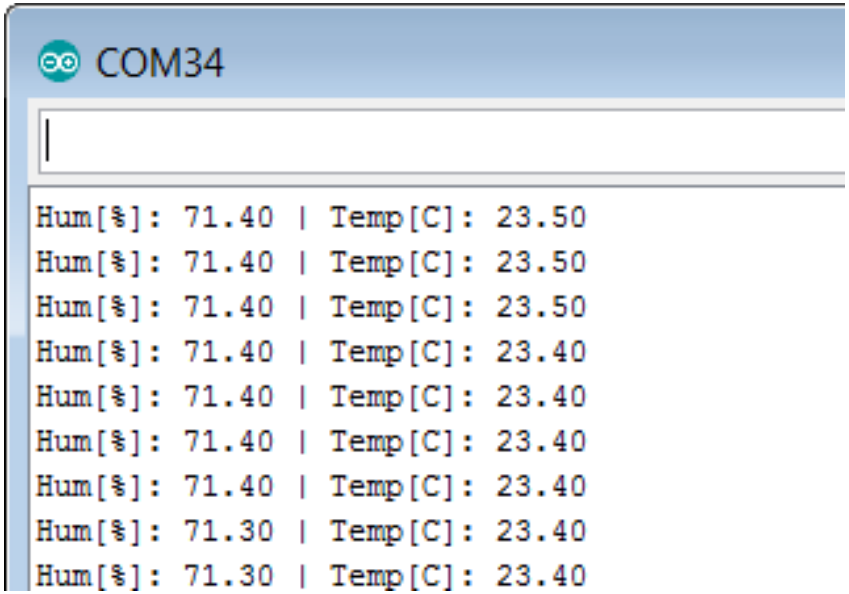
  dev_i2c = new TwoWire(**SDA, SCL**);

- You can comment the definitions:

  // #define I2C2_SCL PB10

  // #define I2C2_SDA PB11

- Upload and check the monitor



Summary
Benefits of using Arduino

- Rapid development of application
  - API is very easy to use:
    - it contains mainly basic C forms (functions)
    - one or few lines of code is/are enough to make peripherals work
    - common API for all supported MCUs (easy porting/migrating)
  - API covers not only MCU's resources, but also external components (sensors, wireless communication, …)
- Multiple MCUs platforms
  - Supports ARM and non-ARM MCU platforms from different vendors
- Supports many operating systems
  - Version for Windows, Linux and MaxOS

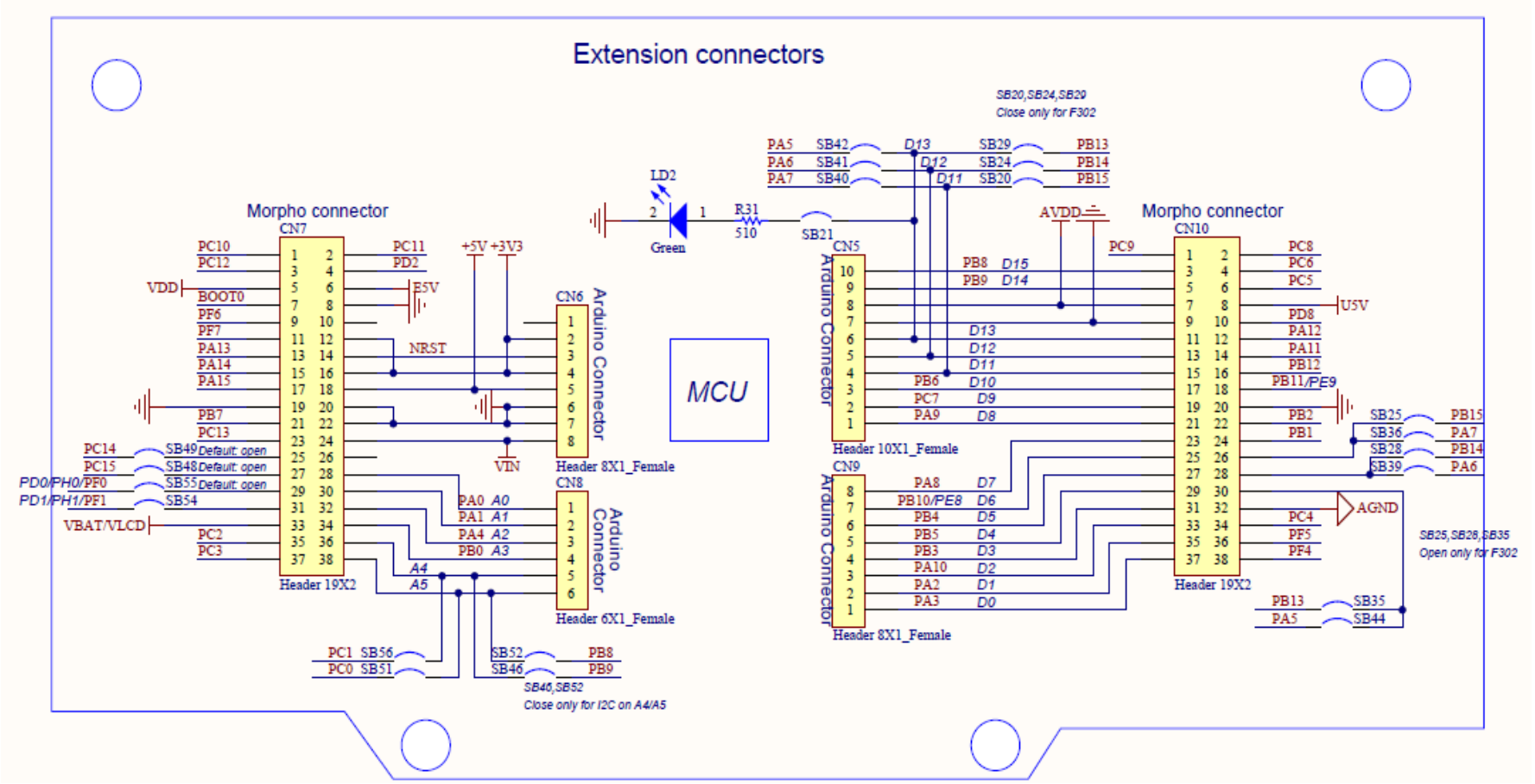
- Free of charge
  - IDE is free of charge and doesn't have any time or code size limitations
- Open source
  - Libraries for MCUs and companion devices are developed by community

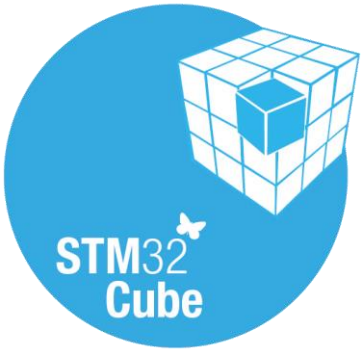
Summary

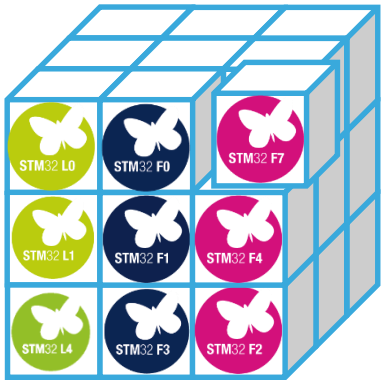
Drawbacks of using Arduino

- Limited selection of MCUs within family
  - Only part of MCUs from selected family are supported by Arduino
- Limited control of MCU
  - Detailed configuration of MCU is not possible like system clock configuration, low-power modes and more
- Limited debugging posibilites
  - There is no real debugger, which allows to run the code step by step, analyze MCU's registers, place breakpoints and more. Main debugging tool is serial monitor
- Limited number of pins for interconnect
  - Arduino standard connect allows a few IO pins (PWM, UART, I2C, SPI…)
  - Which had been extended with the MEGA

Arduino pin out & Morpho connectors



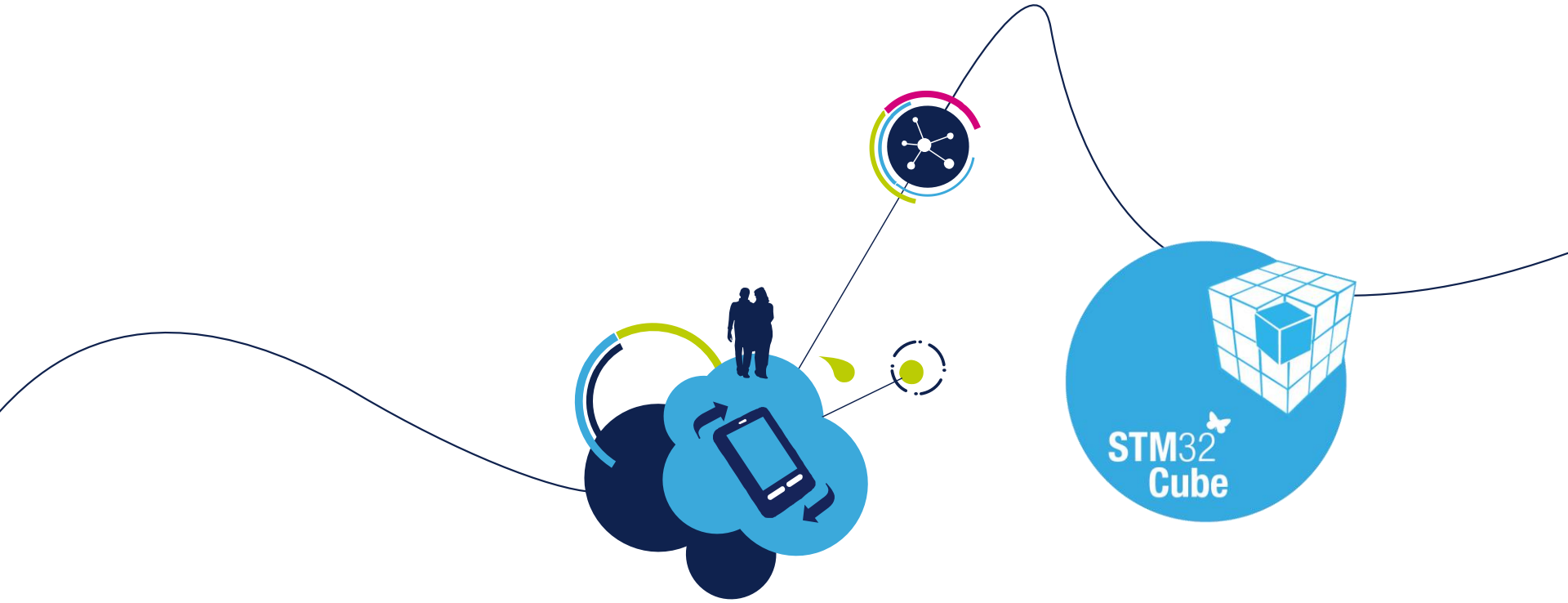




- Now it is a right time for some software activity with STM32 cube
- STM32 cube allows you fast prototyping like Arduino while offering full/precise configuration & access to power of STM32 devices

Let’s discover with an example

- 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



Creating the ‘L4_Blinky’ example in STM32CubeMX

life.augmented

Goal of this part

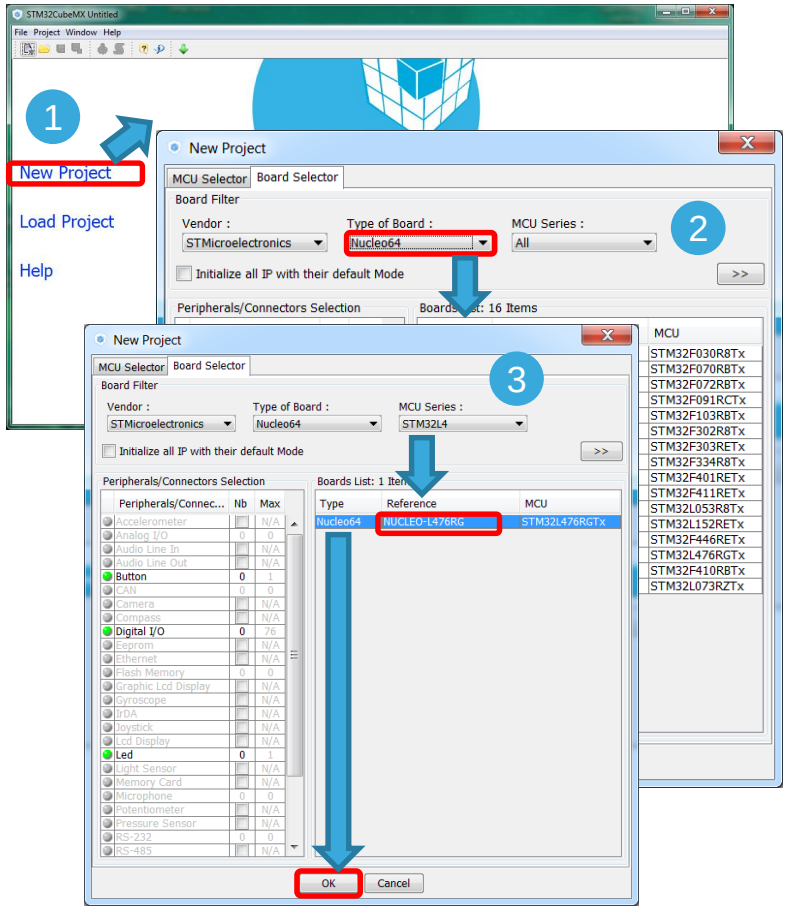
- To practice a little bit with STM32CubeMX by:
  - MCU selection
  - Play a bit with clock configuration for STM32L4 device
  - Create a skeleton of simple LED blinking application
- Have some fun!

Creating a New Project

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

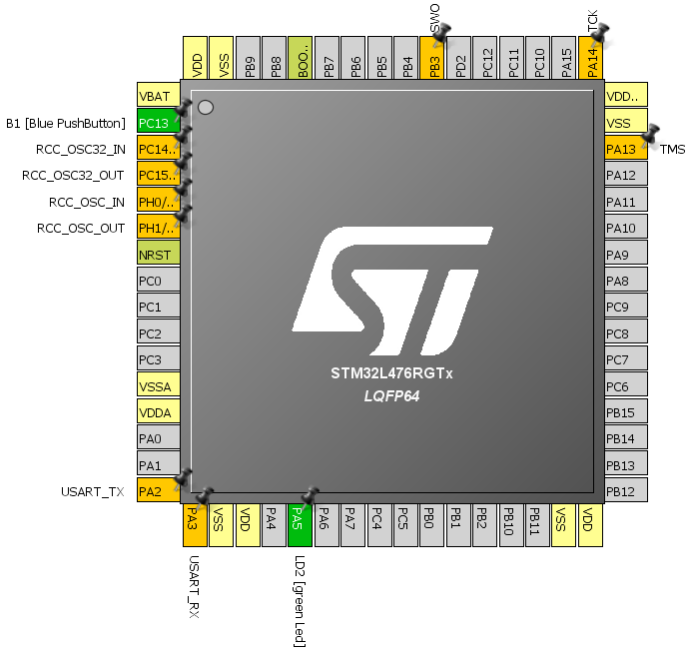
2. There are 3 ways to create a New Project
   - By STM32 Series and Product Line
   - By Peripheral Mix
   - By Board

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

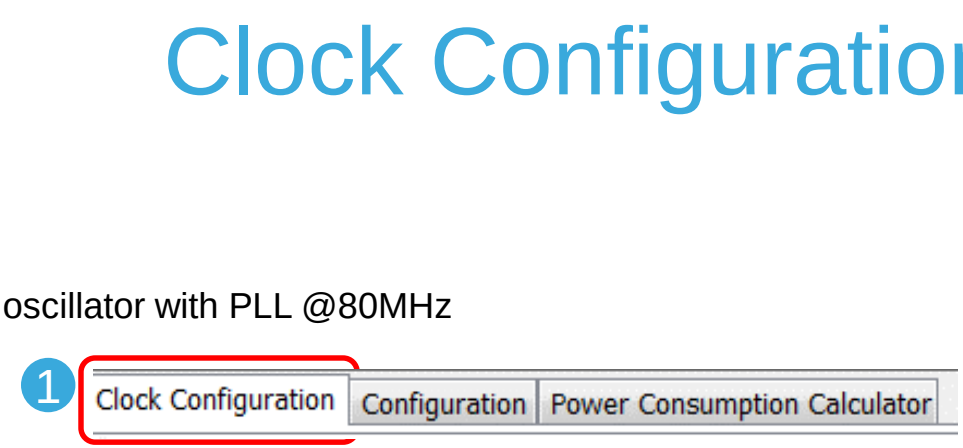
- You will be presented with the pinout of the **NUCLEO–L476RG**
- The debug pins, Push Button and LED are already highlighted in **green**, to say they are connected to the hardware on the board.
- System Pins are highlighted in **yellow**/**yellow-grey**
- Optional Hardware, like crystals and USART 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.

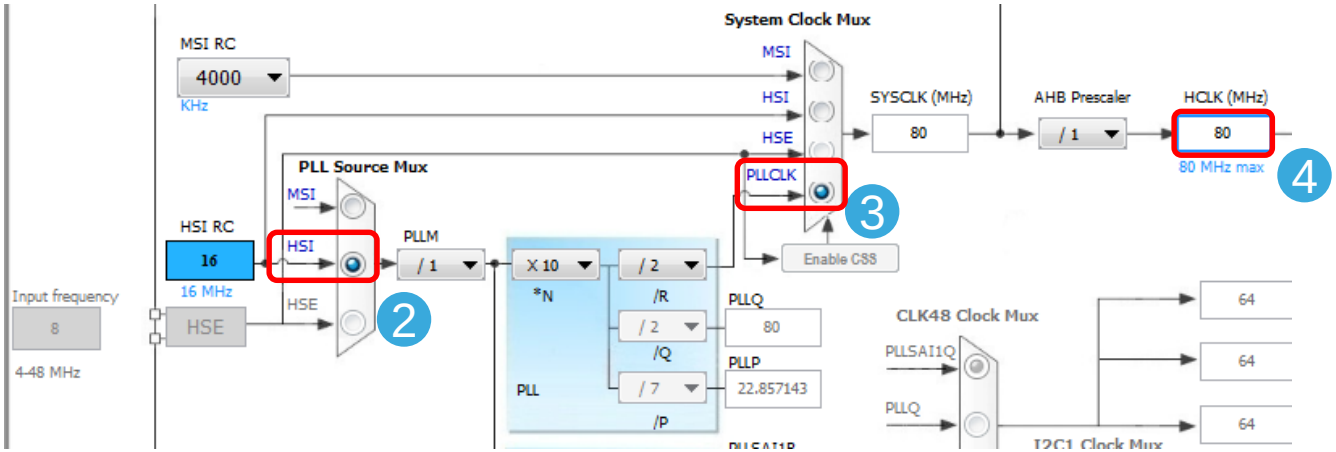


Clock Configuration

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

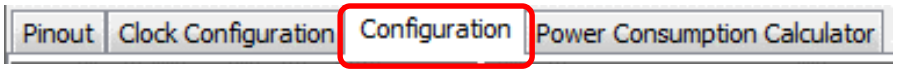
1. Select '**Clock Configuration**' tab
2. Select **HSI** in PLL Source Mux (HSI – High Speed Internal clock)
3. Select **PLLCLK** in the System Clock Mux
4. Set HCLK to **80** and press ENTER – application will propose PLL configuration to match this requirement



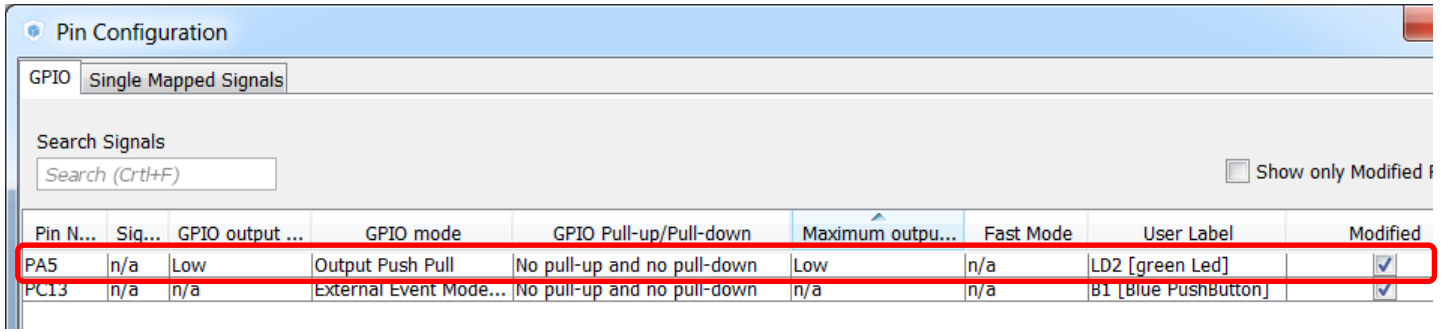


Peripheral Configuration

- Select '**Configuration**' tab

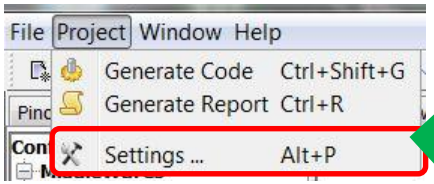


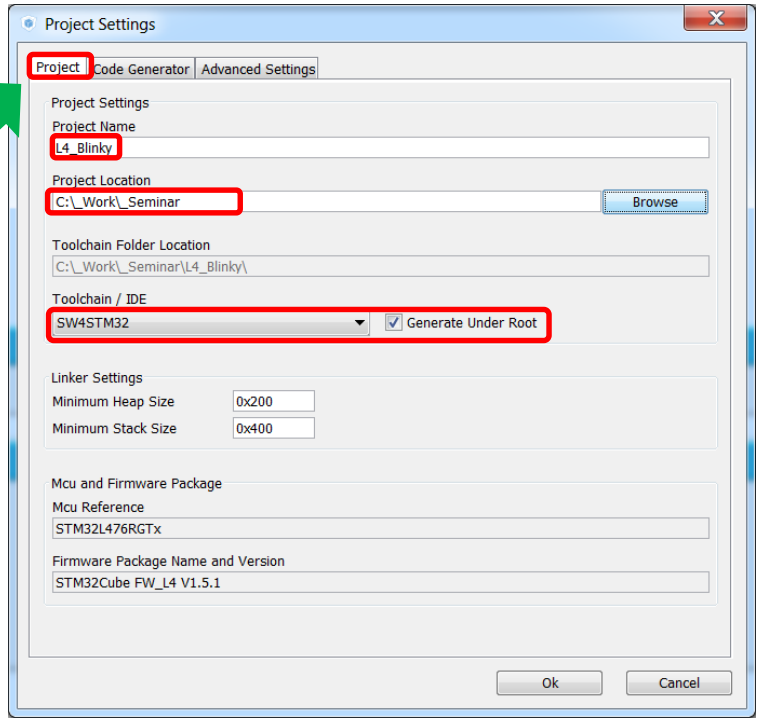
- In this section peripherals with no physical pins or middleware can be added to the project
- For the 'L4_Blinky' example no additional configuration is required as LED is already configured in **GPIO** link as **Output Push-Pull**.



Configure the code generator 1/2

- Open project setings:
  - Menu → Project → Settings



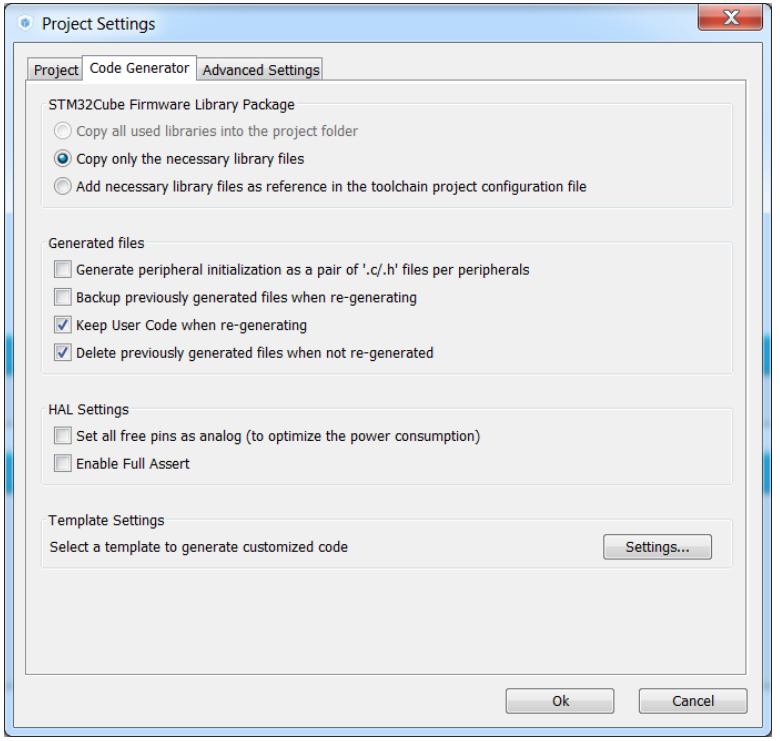


- Under **Project** tab:
  - Give the project a name and location (i.e. **L4_Blinky**)
  - *We strongly recommend to place this folder on the root of 'C:' as some C-compilers show issues when the build path contains too many characters*
  - Select the toolchain to be SW4STM32

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

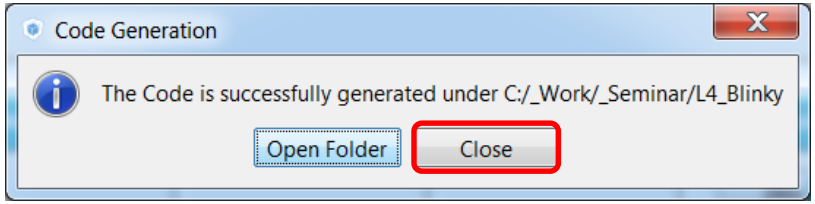
Configure the code generator 2/2

- 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

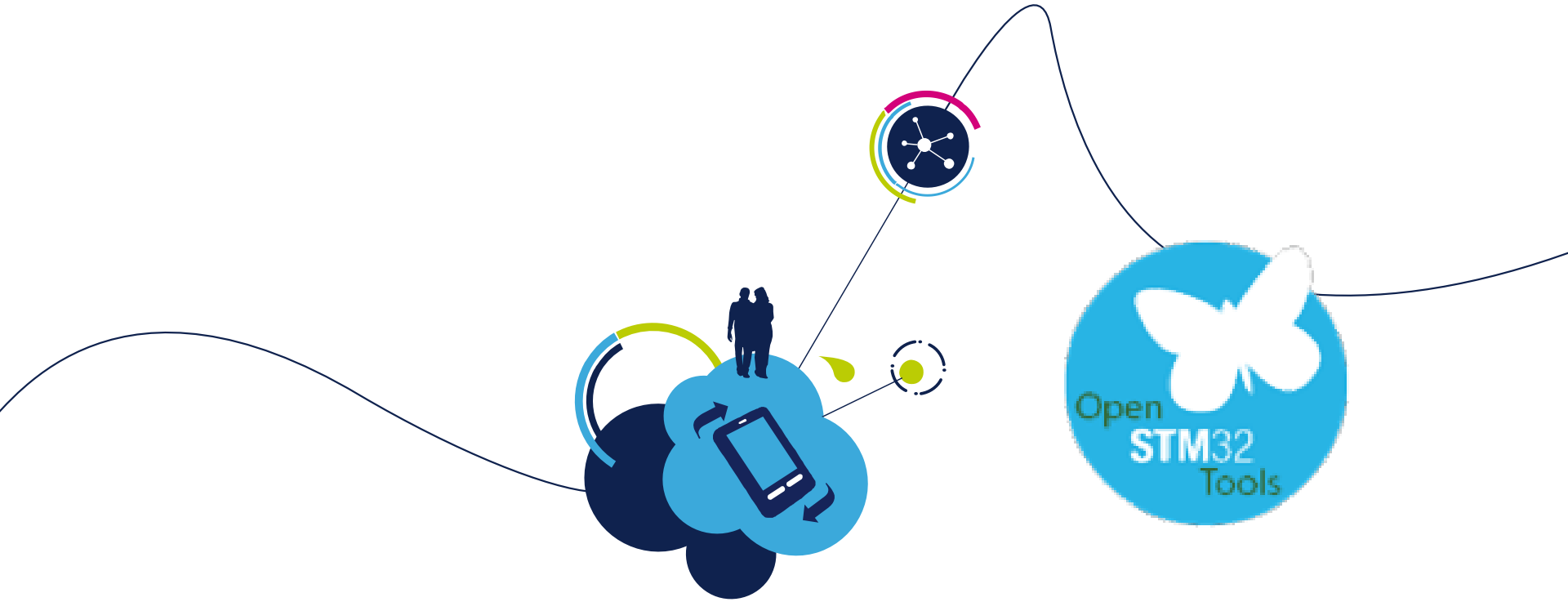
- Once we have configured the code generator, we can generate code for selected toolchain.
- There are 3 ways to do it, namely:
  - Clicking  icon
  - Pressing Ctrl+Shift+G keys combination
  - Selecting Project→Generate Code option from menu
- When prompted, click ‘Close’ (we will import this project 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!

- After successful code generation by STM32CubeMX this is the right time to import it into SW4STM32 toolchain for further processing



Handling the project in SW4STM32

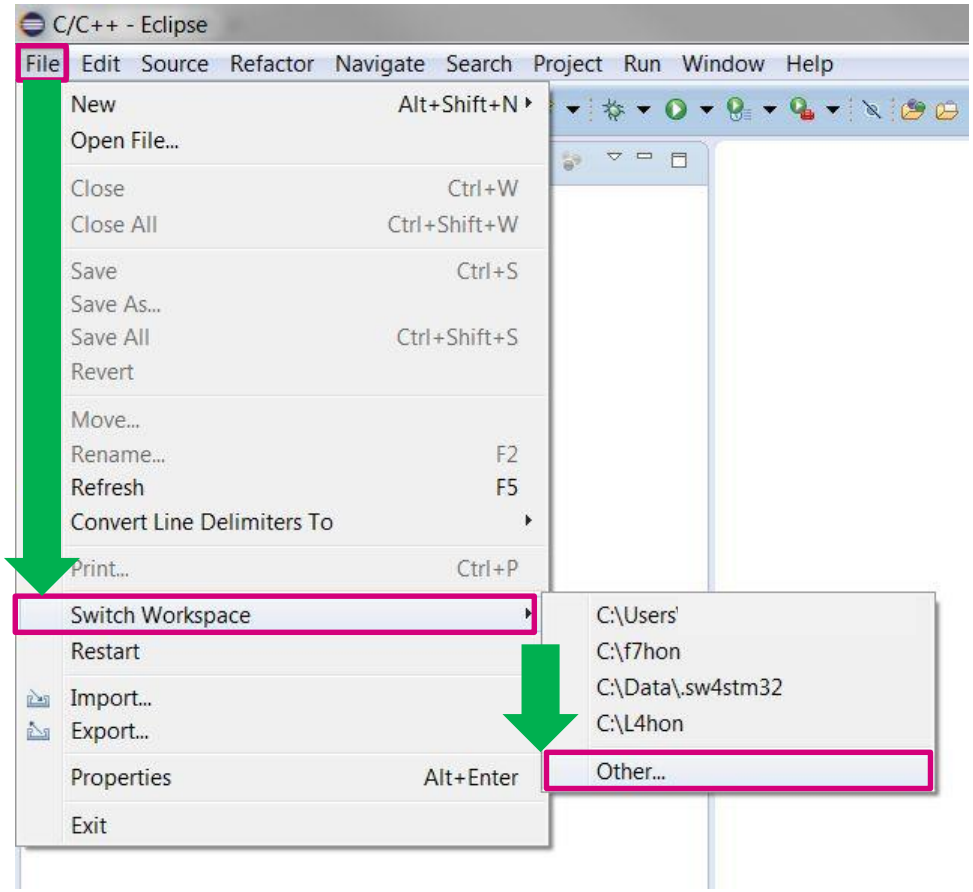
life.augmented

Our goals for this session

- Handling the projects generated by STM32CubeMX in SW4STM32
  - Import project generated by STM32CubeMX
  - Tune sources to run selected peripherals in desired algorithm
  - Build project
  - Configure debug session
  - Run debug session
  - Debug perspective
  - Watching the variables and registers content
  - Handling errors

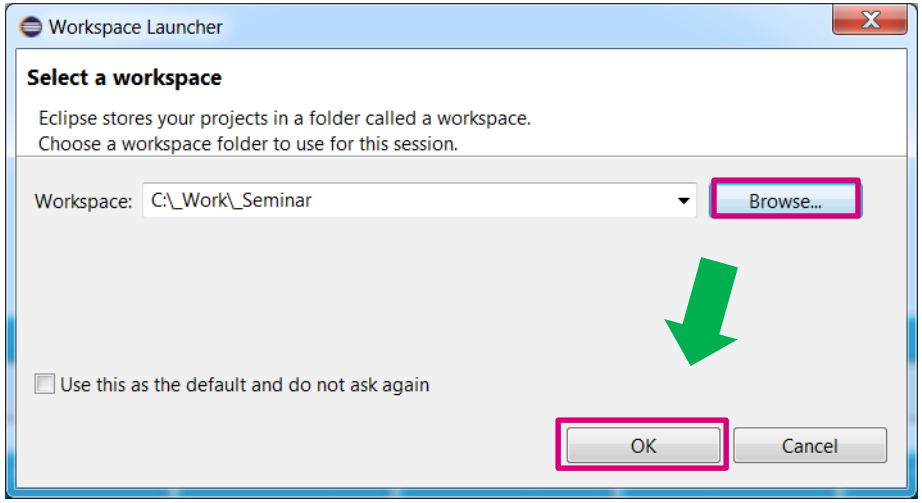
Create a new workspace
SW4STM32

- Start Workspace launcher if not done automatically by Eclipse.

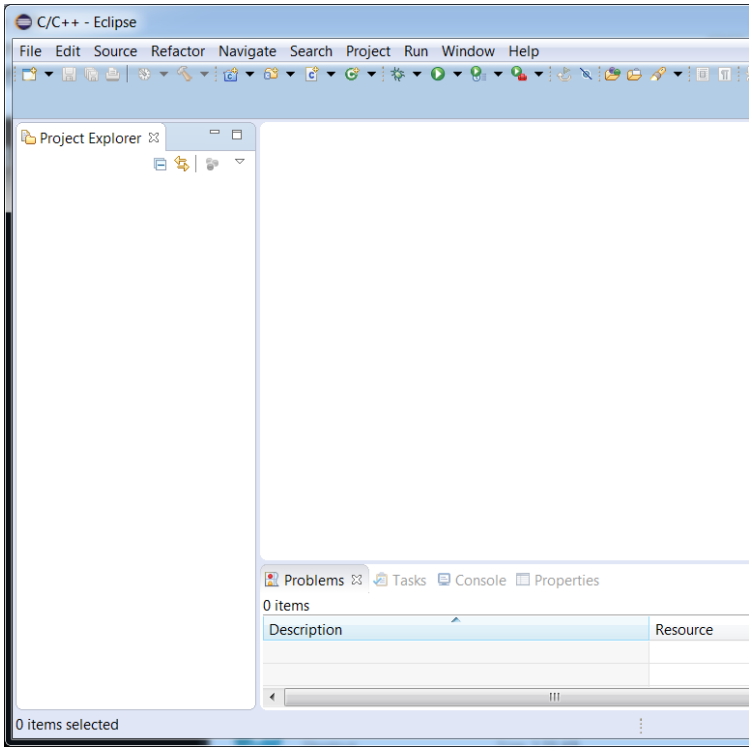


Create a new workspace
SW4STM32

- Create new workspace in the desired location – but not in the same folder where the project which will be imported is located (it must be one level above the project)



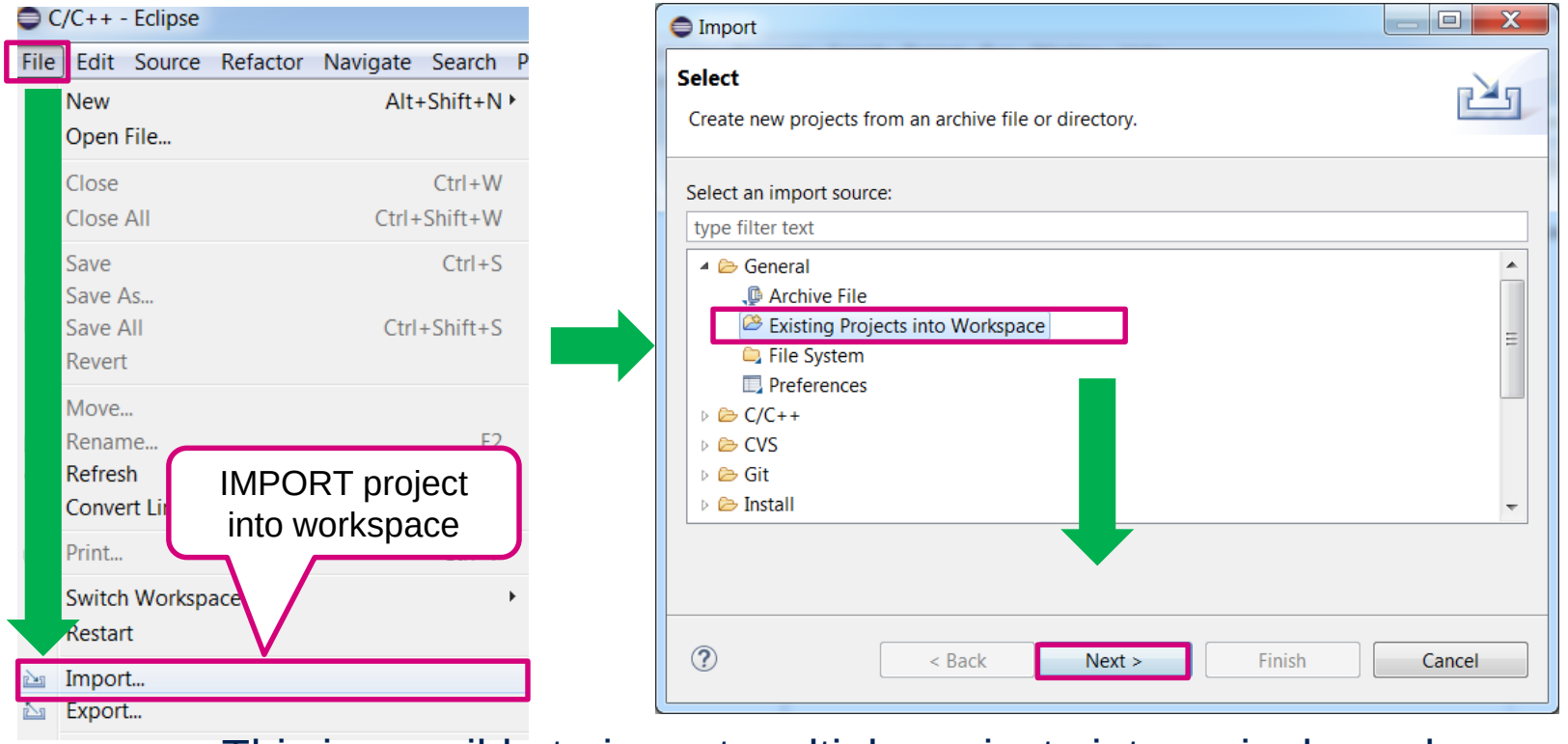
- An empty workspace will be generated



Import the project into the workspace 1/3

SW4STM32

Import “L4_Blinky” project into empty workspace following below steps:

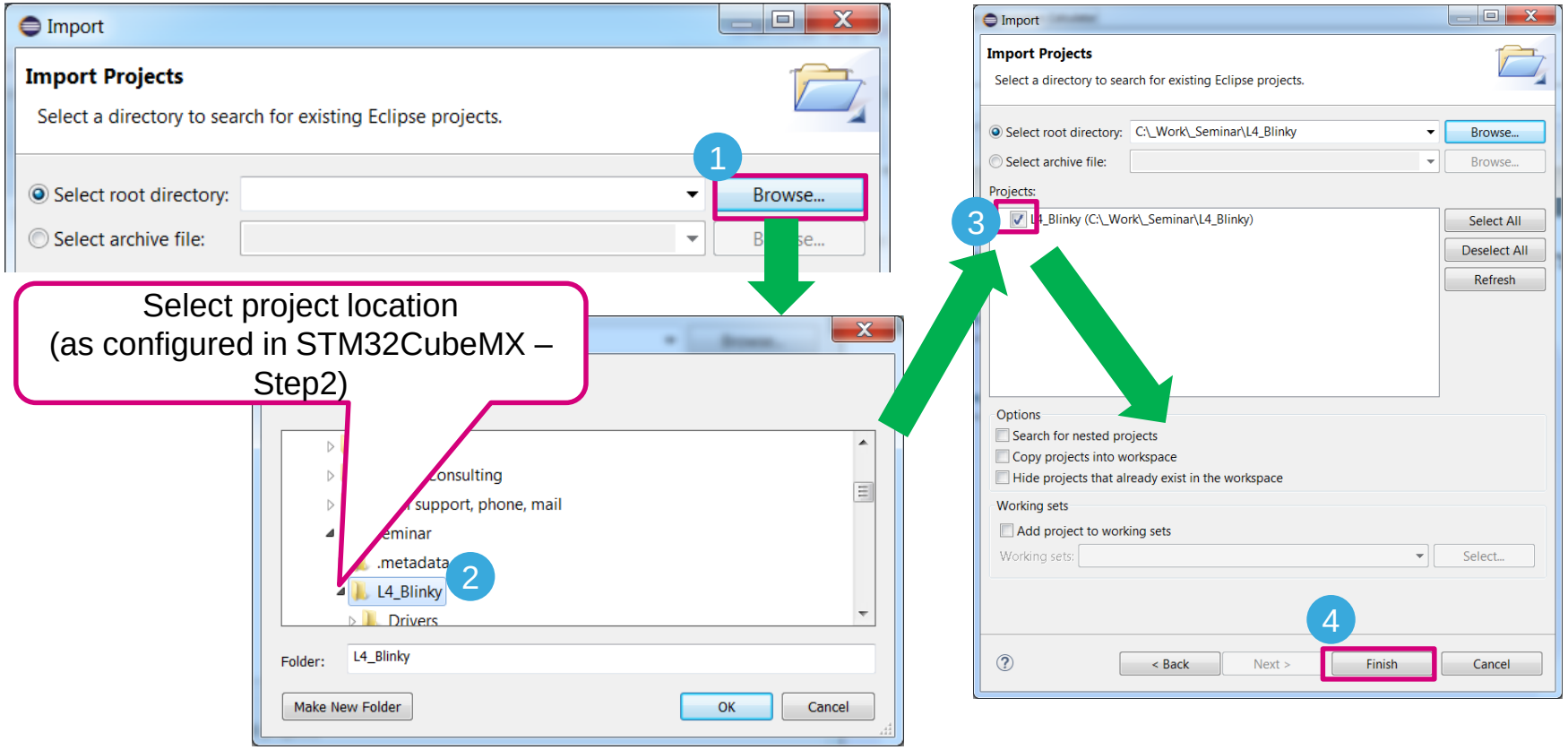


This is possible to import multiple projects into a single workspace

Import the project into the workspace 2/3

SW4STM32

In this example L4_Blinky project will be processed.

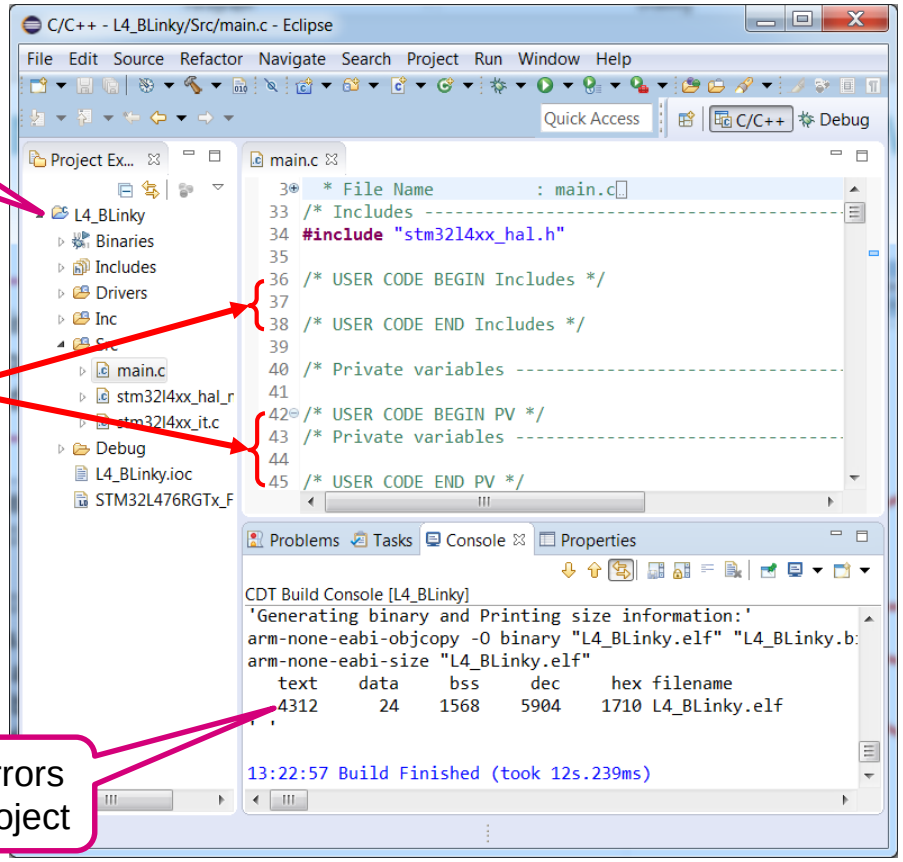


Import the project into the workspace 3/3

SW4STM32

Once project is included into the workspace, its folder structure becomes visible in Project Explorer

Places dedicated for user code are marked by
`/* USER CODE … BEGIN*/`
and
`/* USER CODE … END*/`
comment lines.
These places are protected from being removed during code re-generation by STM32CubeMX.
This is **possible** to define another user code places in **.c** source files but **not possible** in **.h** header files.



Warnings and errors after build the project

- STM32CubeMX generated project is only a skeleton which should be filled with some code from our side
- To make green LED (connected to properly configured PA5 pin) we should continuously invoke GPIO toggle function with the proper delay to make blink visible

Modifying the code
blinking green LED (PA5)

Tasks (within while(1) loop in main.c):

1. Add GPIO pin toggle function for PA5 pin. Which function we can use here?
   ?

2. Add 500ms delay between each change of the GPIO pin state. Which function we can use here?
   ?

Hints:

- All HAL function begins with **HAL_PPP_** prefix (PPP – short name of the peripheral, i.e. GPIO)
- Please try to use Content Assistant (**Ctrl+SPACE**) in Eclipse

Modifying the code

blinking green LED (PA5) - solution

Solutions (within while(1) loop in main.c):

1. Add GPIO pin toggle function for PA5 pin. Which function we can use here?

```
HAL_GPIO_TogglePin();
```

2. Add 500ms delay between each change of the GPIO pin state. Which function we can use here?

```
HAL_Delay();
```

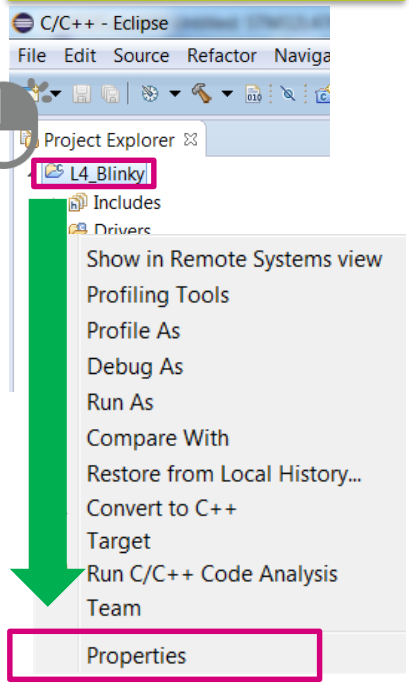
```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
  HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
  HAL_Delay(500);
}
/* USER CODE END 3 */
```

Useful project settings in SW4STM32

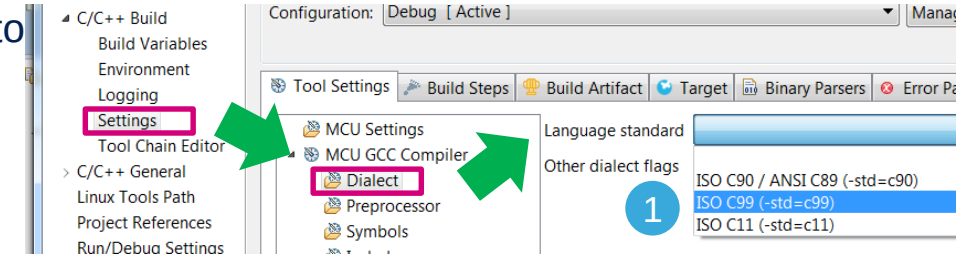
configuring C dialect and parallel build

Project->Properties



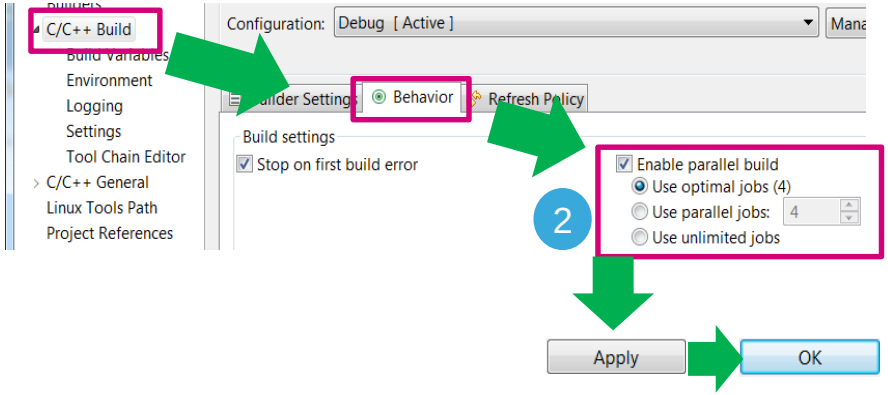
C/C++ Build->Settings->Tools Settings tab->MCU GCC Compiler->Dialect

1. Configure C standard to **C99** to avoid possible compilation errors



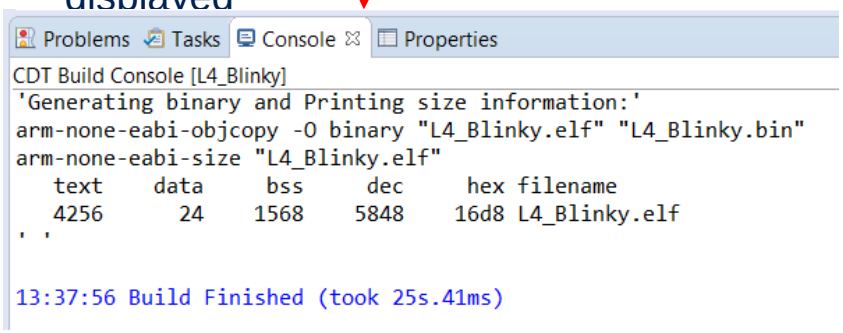
C/C++ Build->Behavior tab

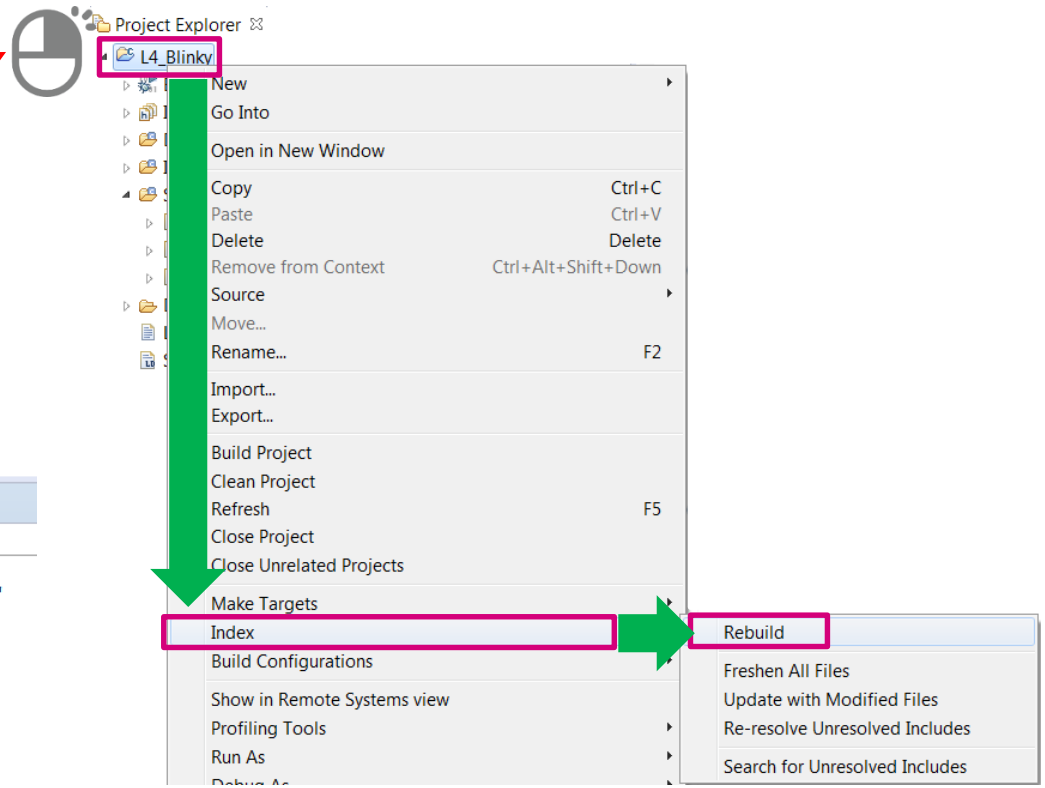
2. Check **Enable parallel build** to make use of your machine potential and to shorten compilation time



Building the project in SW4STM32

- To build the project press **Ctrl+B** or click Make All icon
- In case of multiple compilation errors, re-run Indexing of the project
- After proper build there are information about code/data space usage in Console window displayed

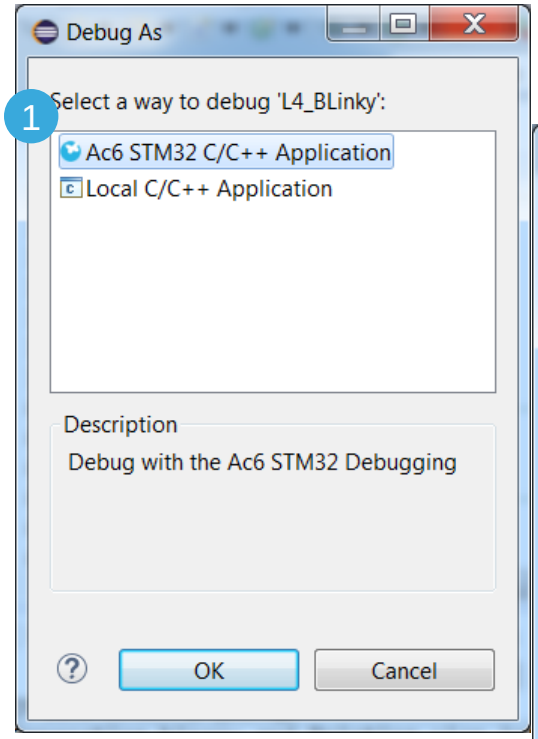


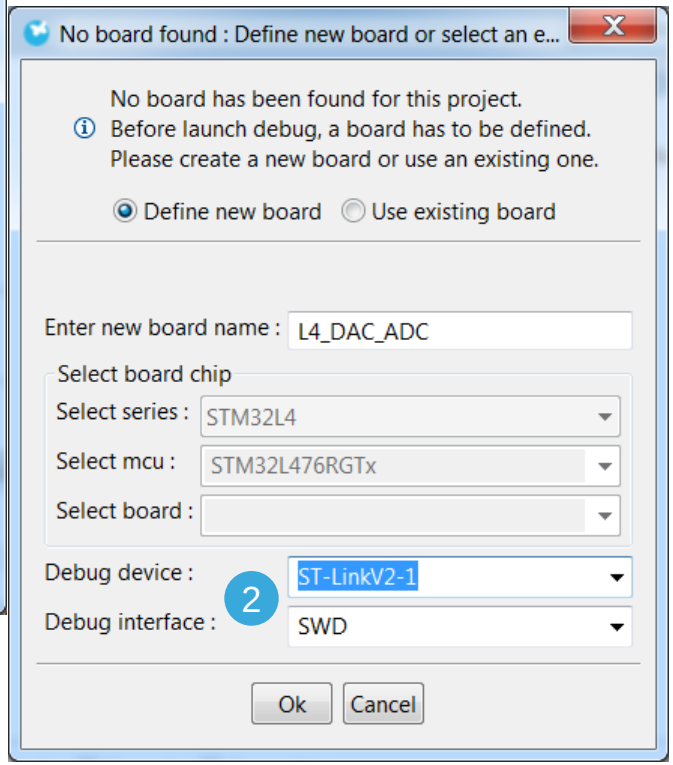


Configure the debug session in SW4STM32

for single project in the workspace

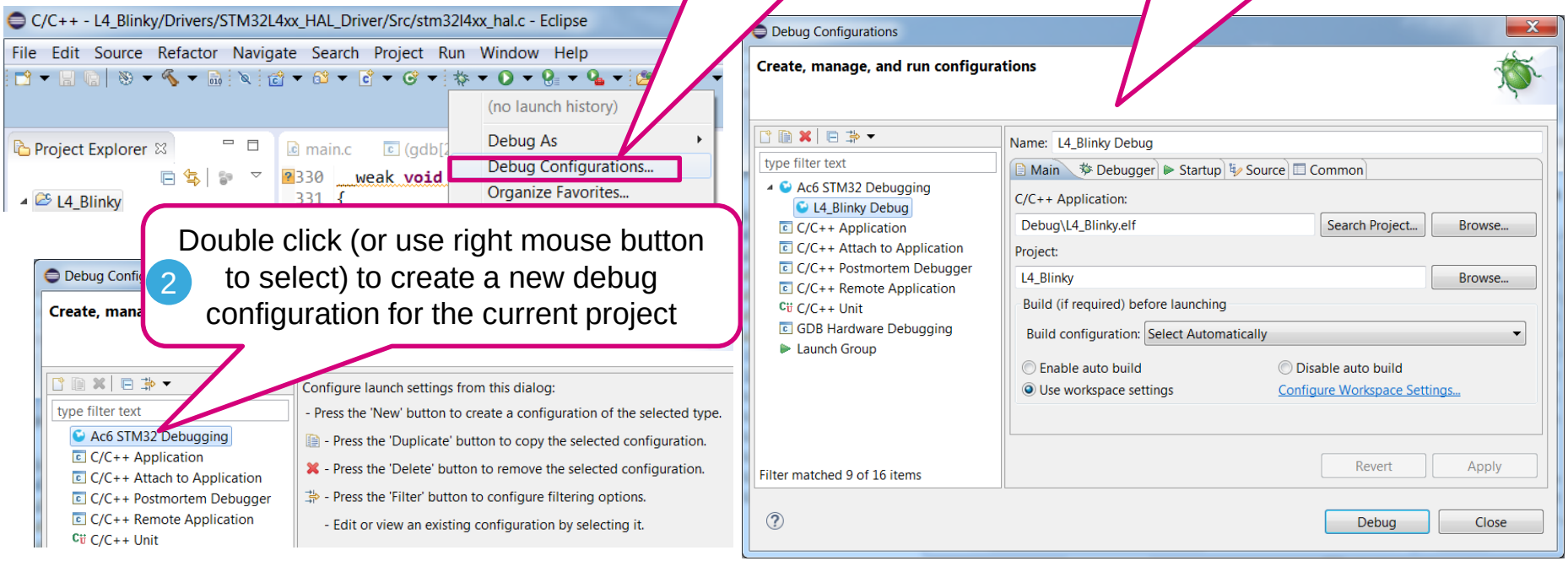
- Before running debug session this is necessary to configure it for current project
- In case there is a single project in the workspace this is enough to click the “bug” icon and:
  1. Select “Ac6 STM32 C/C++ Application” line and click ‘OK’
     In case the project was generated on existing/defined board (like NUCLEO-L476RG in our example) debug will run automatically
  2. Otherwise (we will practice it in L4_DAC_ADC example later) it is necessary to configure debug device (STLinkV2-1 in our case) and debug interface (SWD in our case) and click ‘OK’
- Next step would be to run the debug session (see the next slide)





Configure the debug session in SW4STM32
for multiple projects in the workspace 1/2

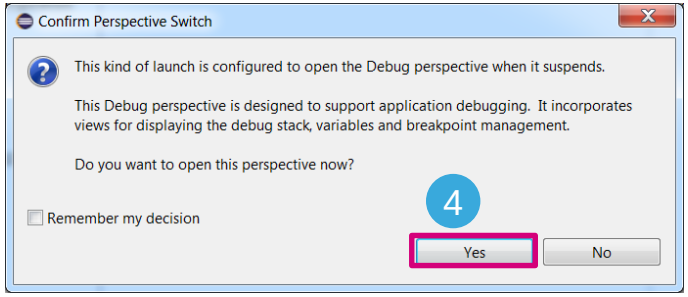
Before debugging current project for the first time, this is necessary to configure its debug session

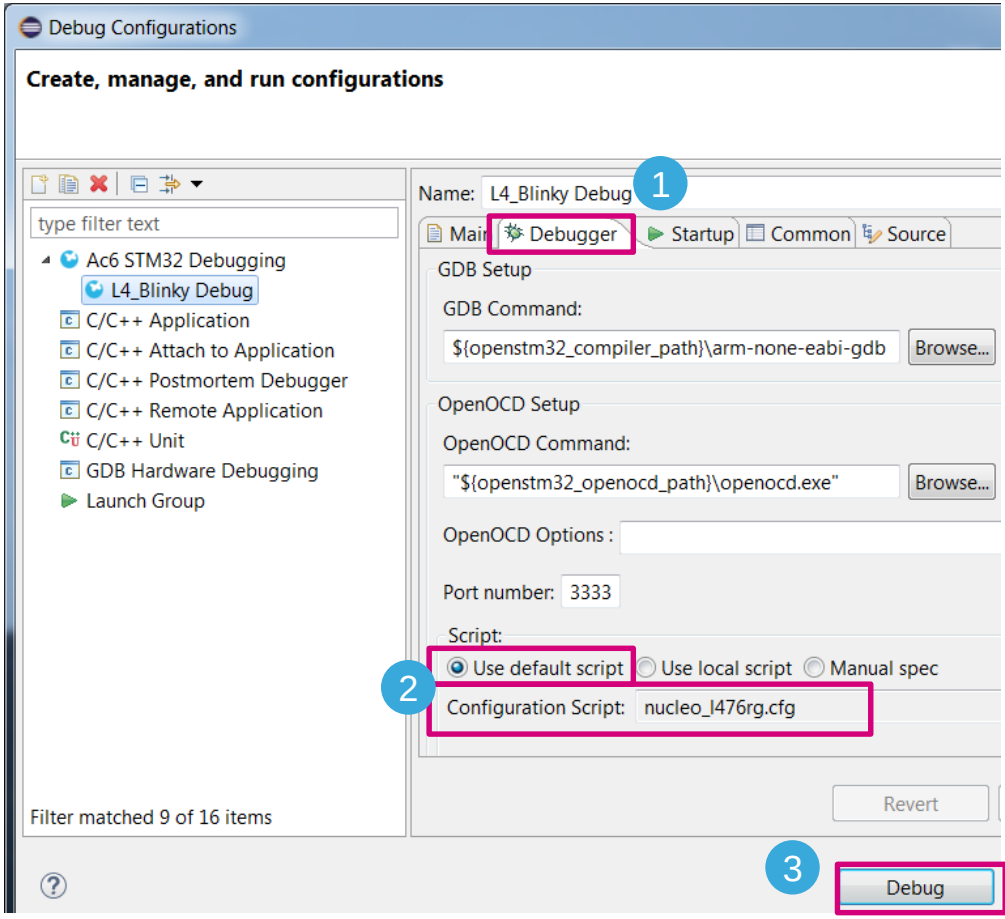


Run the debug session in SW4STM32

for multiple projects in the workspace 2/2

- Connect Nucleo board with miniUSB cable (ST-Link)
- In case of the projects generated for ST board, there should be selected board configuration script which specifies debug device and its interface (you can check it in Debugger tab)
- Debug perspective will be run (please select Yes in the information window)
- This is enough just to click a “bug” icon to enter debug session next time.

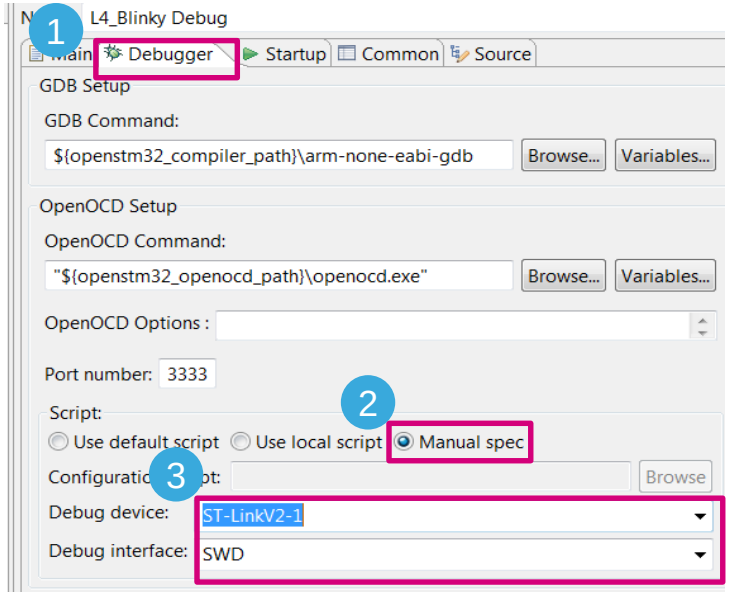


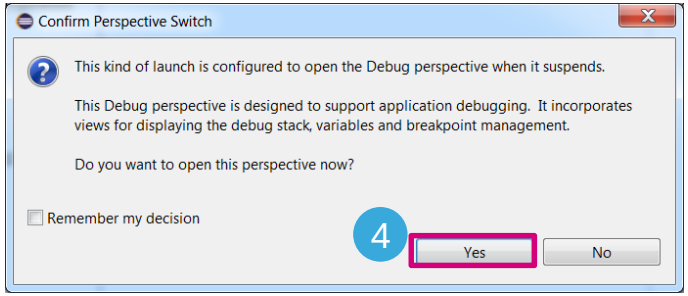


Run the debug session in SW4STM32

for multiple projects in the workspace, but no board specification

- Connect Nucleo board with miniUSB cable (ST-Link)
- Under **Debugger** tab select debug device (ST LinkV2-1 for Nucleo ones) and debug interface (SWD)
- Click **Apply** and then **Debug**
- Debug perspective will be run (please select Yes in the information window)
- This is enough just to click a “bug” icon to enter debug session next time.

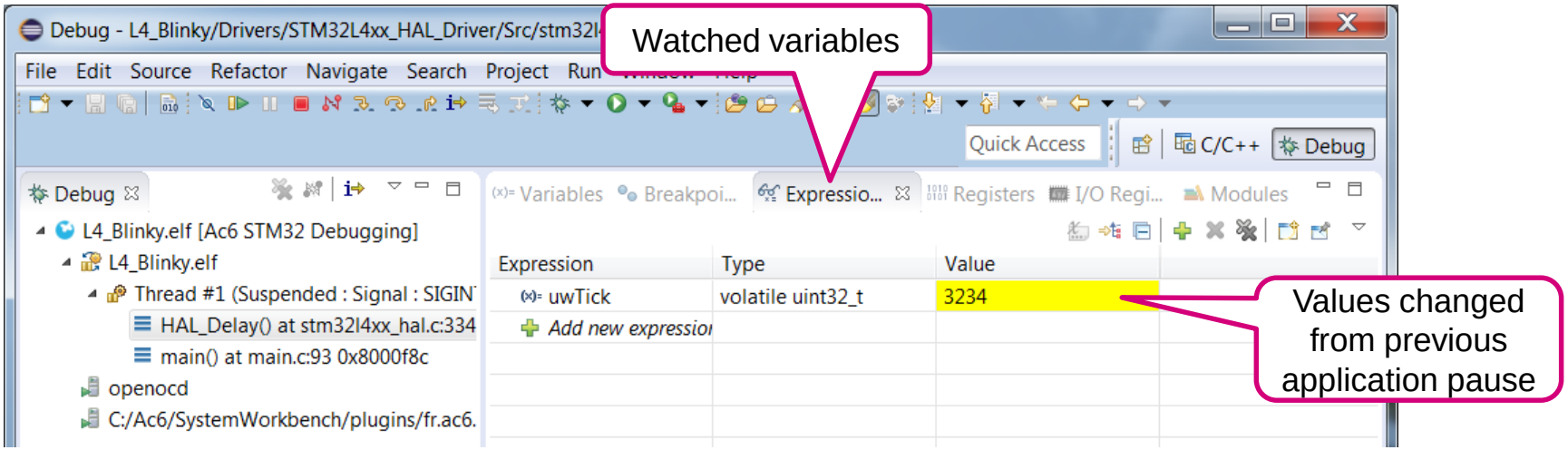




Debug session perspective

watching the variables

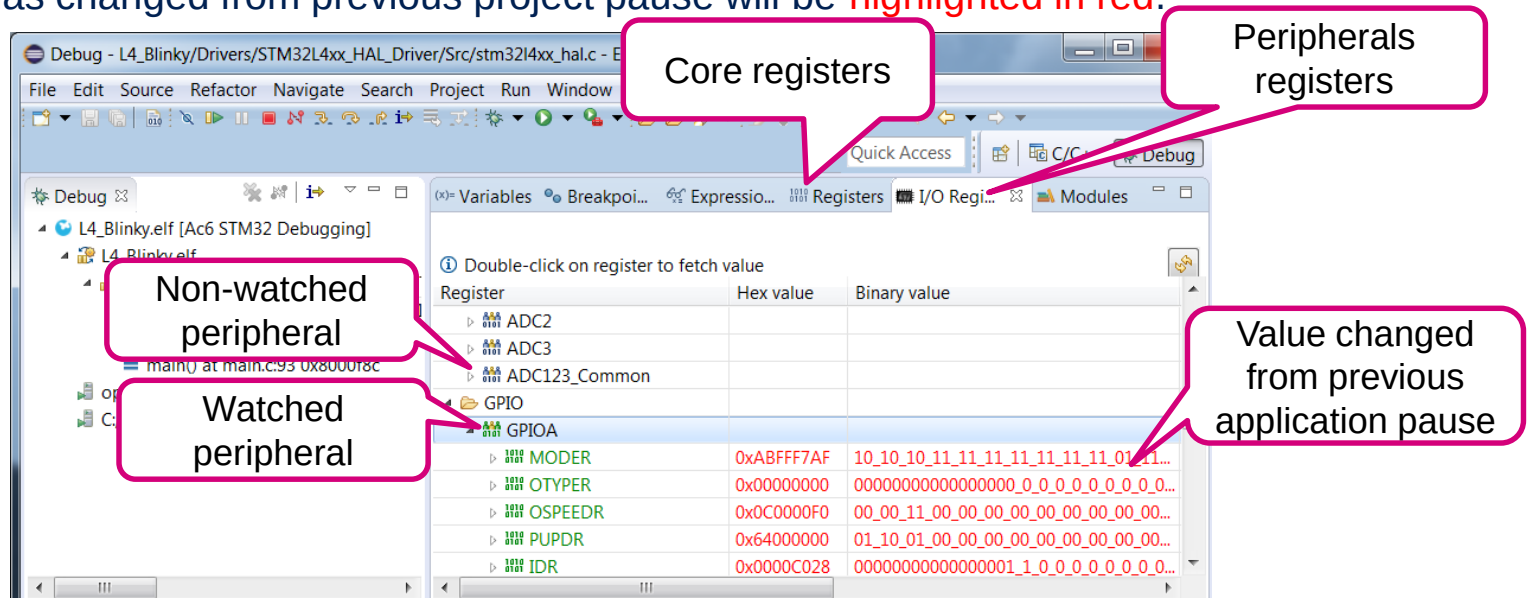
- This is possible to monitor CPU registers, peripherals registers and variables during debug session, but we need to pause the code execution (no live view is possible for the time being).
- To add variable to be monitored - highlight it, press right mouse button and select "**Add Watch Expression**". It will appear in Expressions tab then.
- Values which has changed from previous project pause will be presented on yellow background



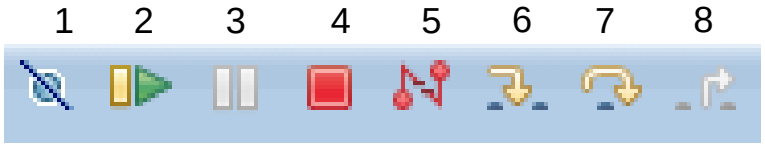
Debug session perspective

watching the registers content

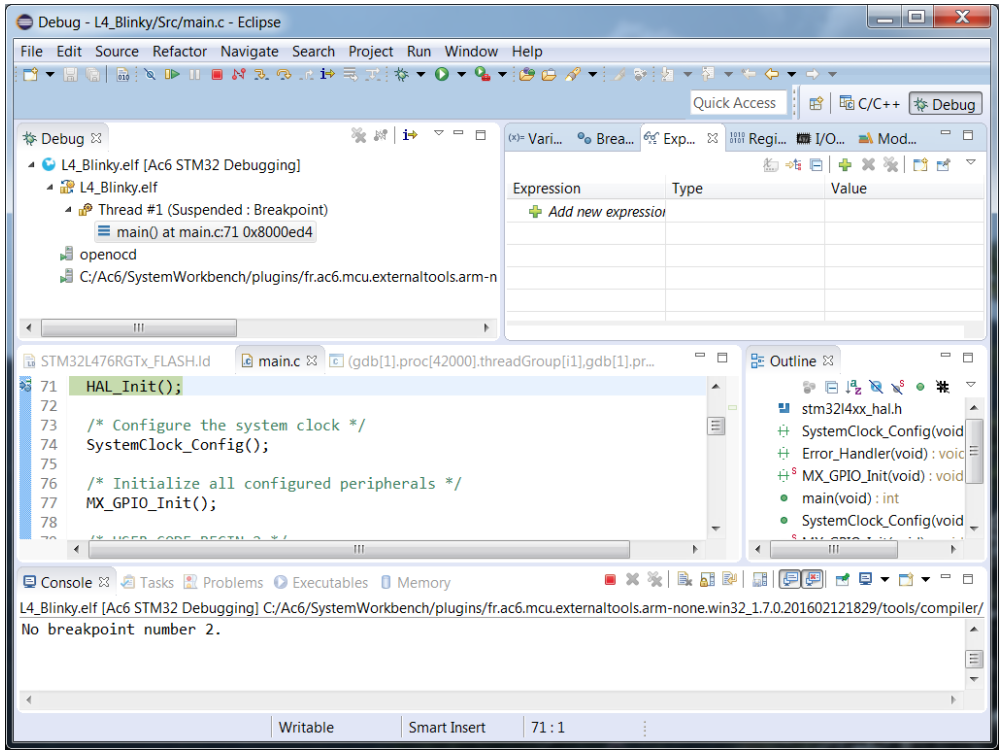
- This is possible to monitor CPU registers, peripherals registers and variables during debug session, but we need to pause the code execution (no live view is possible for the time being).
- To add peripheral register to watch - click right mouse button and select "**Activate**". Peripheral icon and its registers names will be highlighted in green and will contain "caught" values on next debug pause.
- Values which has changed from previous project pause will be highlighted in red.



Handling the debug session
SW4STM32



1. Skip all breakpoints
2. Run/resume
3. Suspend
4. Terminate debug session
5. Disconnect from the target
6. Step into
7. Step Over
8. Step Return



windows configuration in debug perspective

What have we learnt?

- ✓Handling the projects generated by STM32CubeMX in SW4STM32
  - ✓Import project generated by STM32CubeMX
  - ✓Tune sources to run selected peripherals in desired algorithm
  - ✓Build project
  - ✓Configure debug session
  - ✓Run debug session
  - ✓Debug perspective
  - ✓Watching the variables and registers content
  - ✓Handling errors

Summary
Benefits of using Arduino

- Rapid development of application
  - API is very easy to use:
    - it contains mainly basic C forms (functions)
    - one or few lines of code is/are enought to make peripherals work
    - common API for all supported MCUs (easy porting/migrating)
  - API covers not only MCU's resources, but also external components (sensors, wireless communication, …)
- Multiple MCUs platforms
  - Supports ARM and non-ARM MCU platforms from different vendors
- Supports many operating systems
  - Version for Windows, Linux and MaxOS

- Free of charge
  - IDE is free of charge and doesn't have any time or code size limitations
- Open source
  - Libraries for MCUs and companion devices are developed by community

Summary

Benefits of using STM32 cube

- Rapid development of application

  API is very

  - it
  - or
  - co

  API cover

  wireless co

  Default working (fast) config
  Allocate time necessary to fine tune device.
  LL api and HAL api

  pherals work

  igrating)

  ponents (sensors,

- Multiple MCUs platforms

  Supports

  STM8 & STM32

  ndors

- Supports many operating systems

  Version f

  OSX , Linux & Windows

- Free of charge

  IDE is fre

  0000000.00$

  mitations

- Open source

  Libraries

  Loads of githubs, open source project avail. STM32

  community

Summary
Drawbacks of using Arduino

- Limited selection of MCUs within family
  - Only part of MCUs from selected family are supported by Arduino
- Limited control of MCU
  - Detailed configuration of MCU is not possible like system clock configuration, low-power modes and more
- Limited debugging posibilites
  - There is no real debugger, which allows to run the code step by step, analyze MCU's registers, place breakpoints and more. Main debugging tool is serial monitor
- Limited number of pins for InterConnect
  - Arduino standard connect allows a few IO pins (PWM, UART, I2C, SPI…)
  - Which had been extended with the MEGA

Benefits

Summary

Drawbacks of using Arduino

STM32 cube

- Limited selection of MCUs within family
  - Only part of Arduino
  - Full list of STM32 family members available
- Limited control of MCU
  - Detailed clock configuration, low-power mode
  - Fine & details configuration possible
- Limited debugging posibilites
  - There is no step by step, analyze MCU's register
  - **All debugging facilities (BRK, STEP, WATCH….)**
  - Debugging tool is serial monitor

- Limited number of pins for InterConnect
  - Arduino standard connect allows a few IO pins (PWM, UART, I2C, SPI…)
  - Which had
  - All STM32's pin from package configurable

Useful links

- STM32L4 Discovery kit IoT node
  - http://www.st.com/en/evaluation-tools/b-l475e-iot01a.html
- STM32duino
  - https://github.com/stm32duino/wiki/wiki/Boards-Manager
- ST Community thread about Arduino
  - https://community.st.com/community/stm32-community/blog/2017/07/13/stm32-cores-enabled-in-arduino-ide
- STM32 libraries for Arduino
  - http://www.arduinolibraries.info/architectures/stm32

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

Thank you