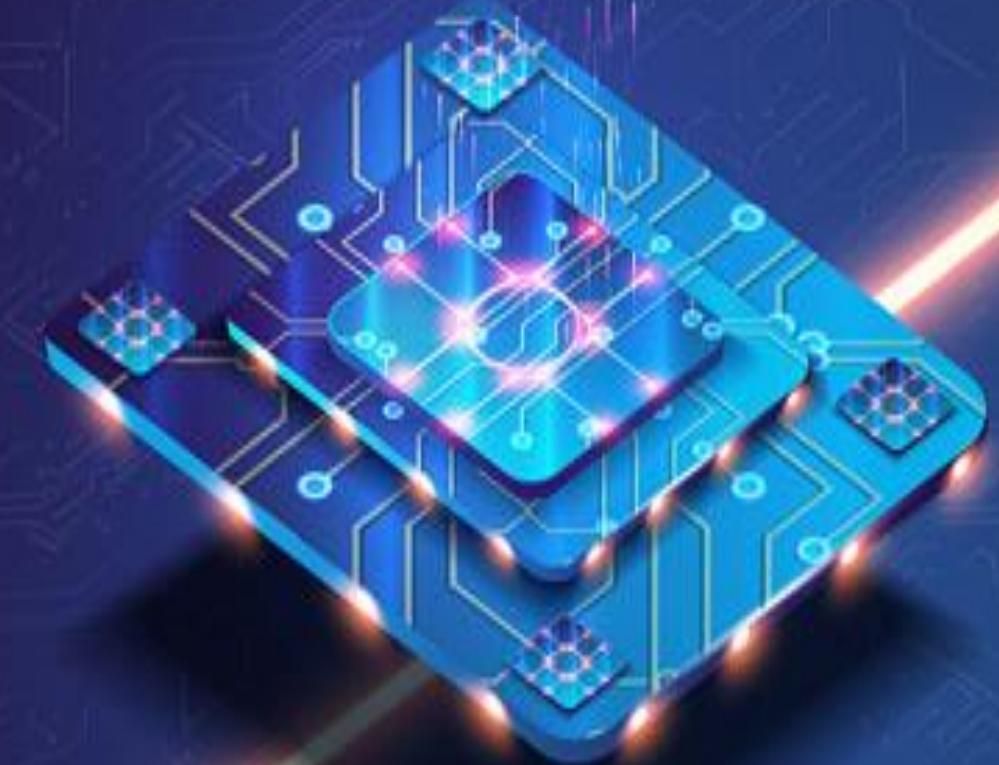


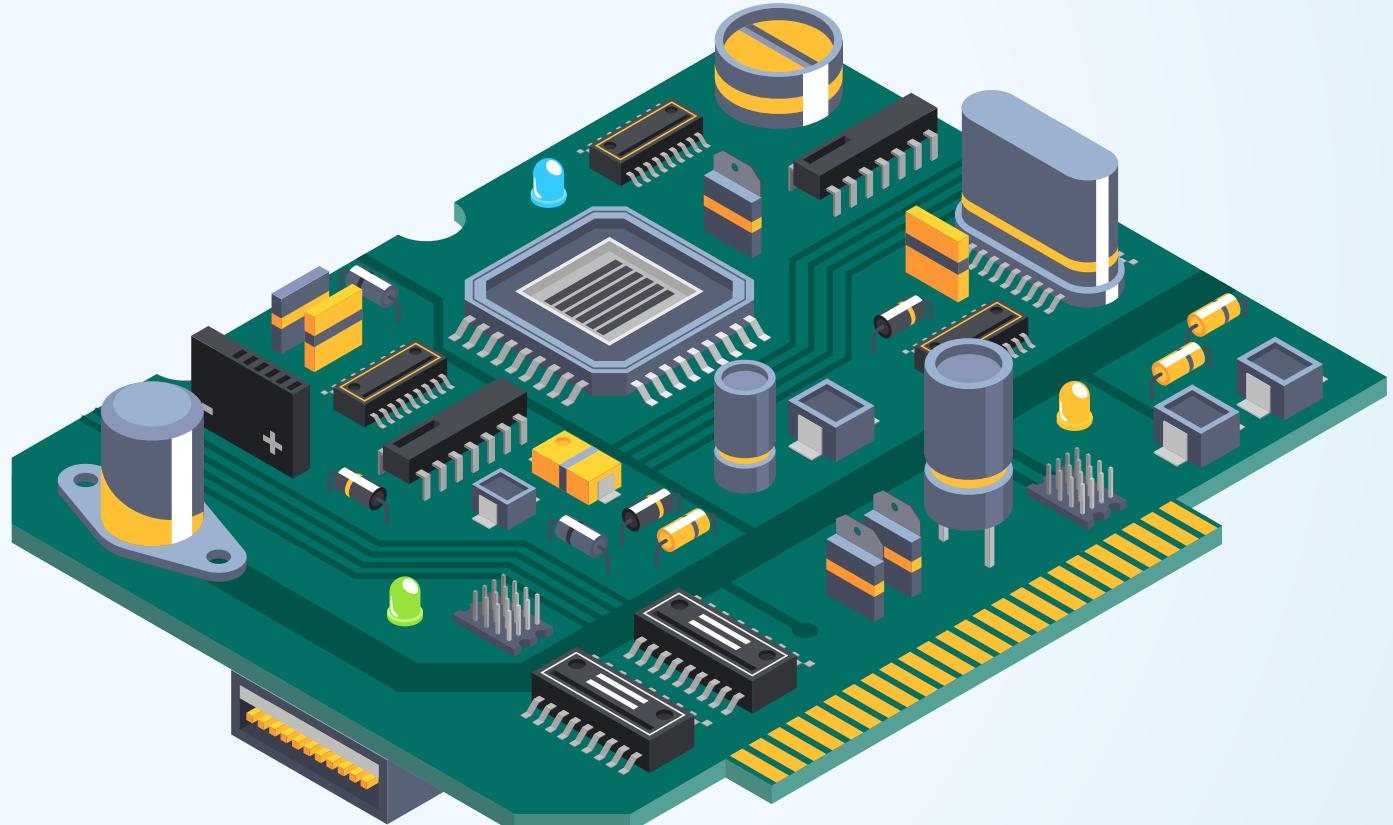
EMBEDDED SYSTEMS

By:

Mohamed Yanis
Hiou



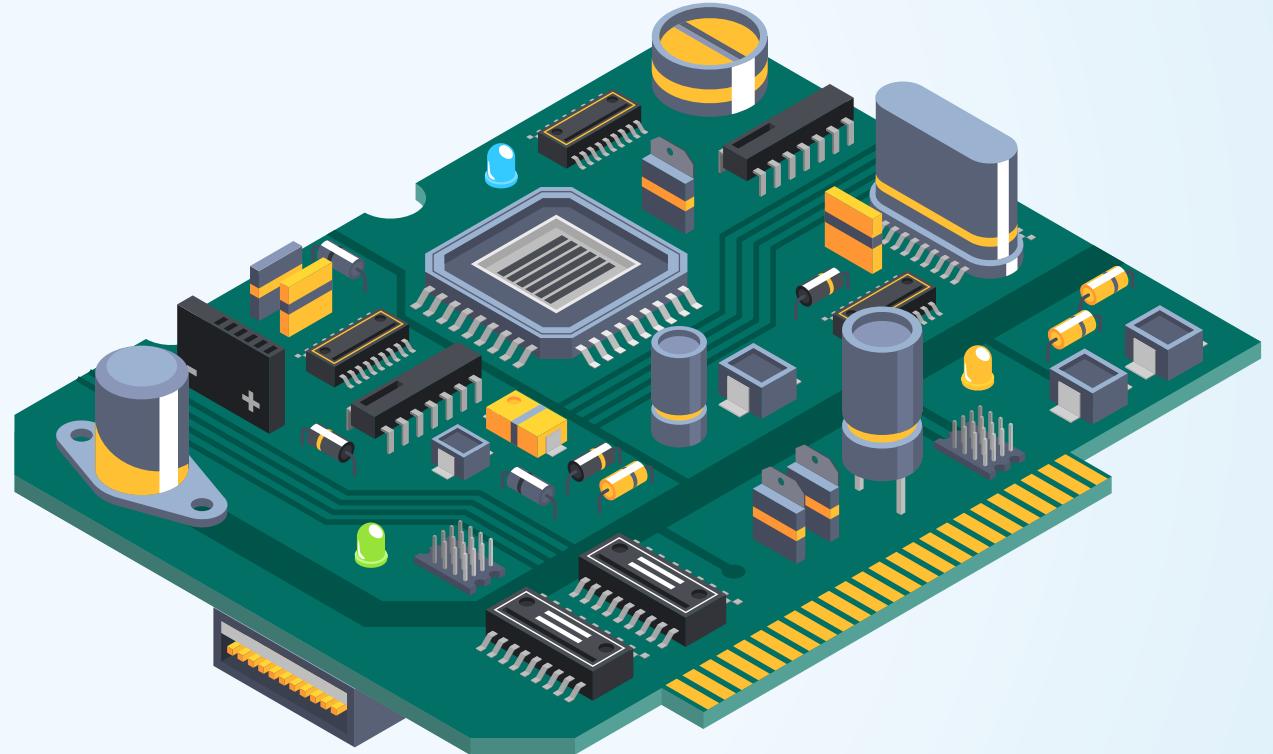
What is an Embedded System ?



An Embedded System is

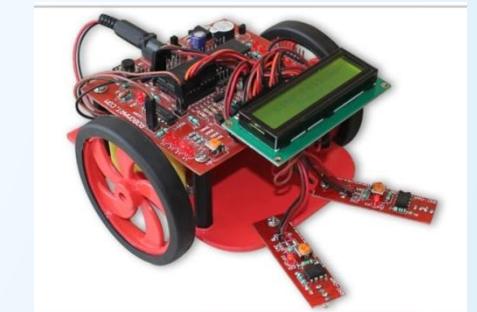
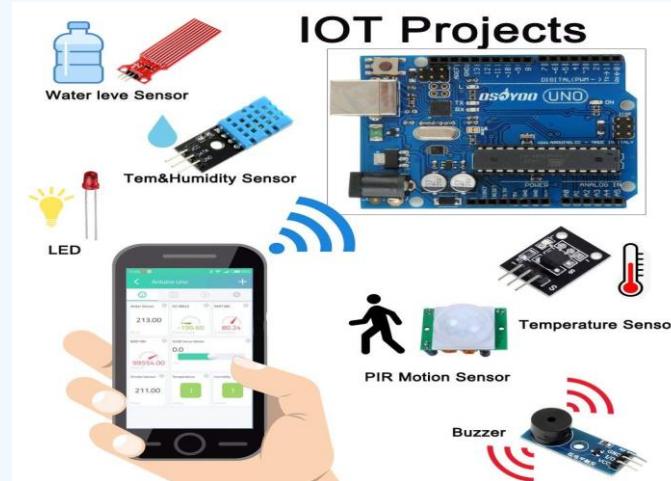
As the name implies, it is a combination between hardware and software that are integrated to perform a particular function.

It uses a Microcontroller/Microprocessor to perform a single job. It is a stand-alone device with or without an operating system.



Embedded System applications

- Smart Homes
- Offices
- Transportation
- Healthcare
- Industrial world
- Aerospace and Defense



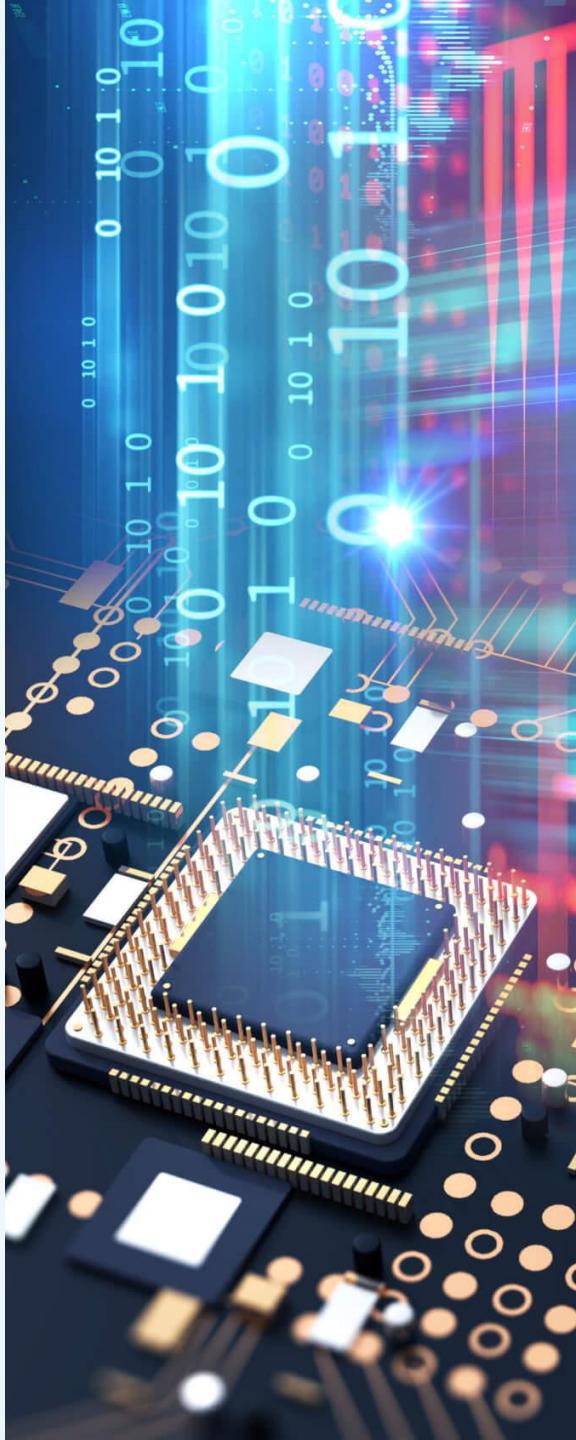
Embedded System Software & Hardware



Every complex system in the world can be made based on Software and Hardware.

To achieve that you have to start building smaller modules and integrate them to create an efficient subsystem.

The embedded system can be divided into software and hardware components

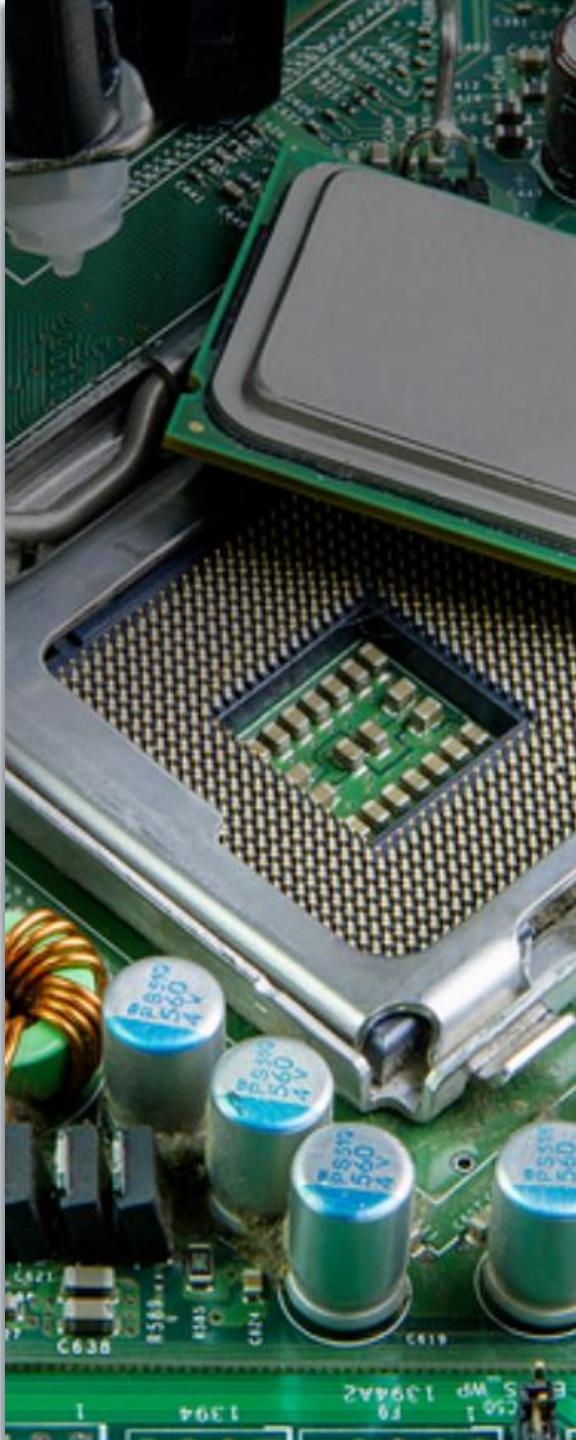


Embedded Hardware

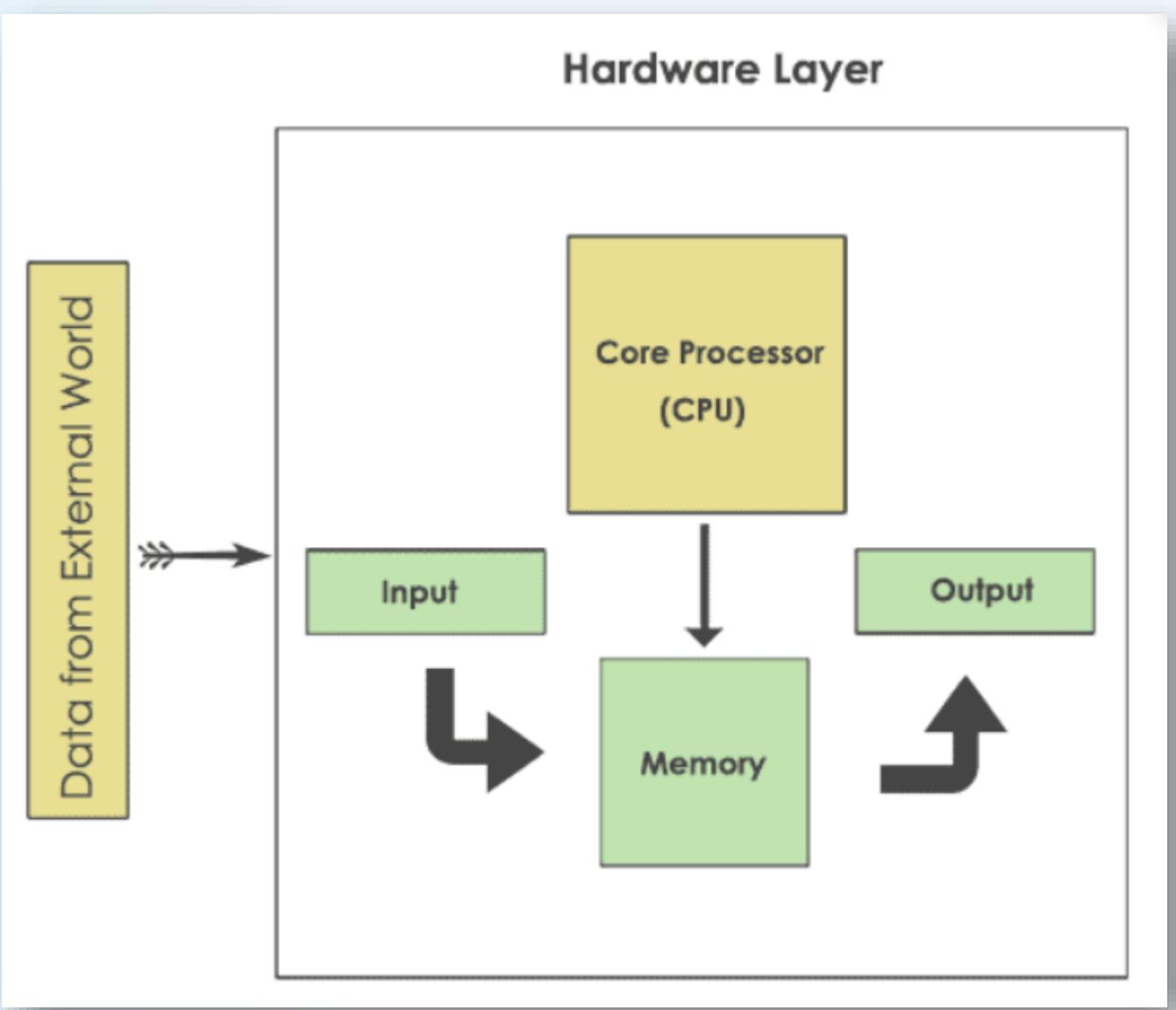


The core of any embedded target is the electronic hardware – which resides on a Printed Circuit Board.

The embedded development board is divided into five modules. They are Processor, Memory, Input devices, Output devices, and Bus controllers



Embedded Hardware

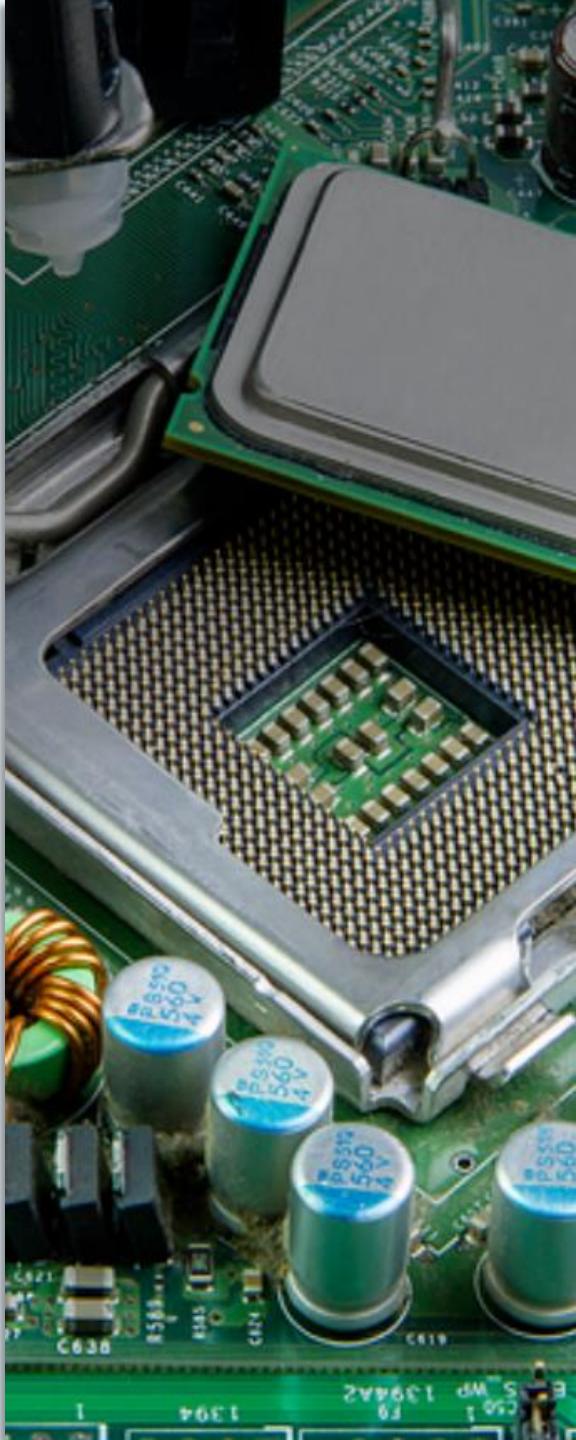


Embedded Hardware



Hardware abstraction layer (HAL) is the fundamental resource of any embedded device and choosing a particular component depends on the requirement and specification of the designer. In the global market, there are many variants of hardware produced for different applications. Some of them are:

- Microcontroller
- System on Chip (SoC)
- ASIC processor
- DSP processor
- Output Devices
- Input Devices
- Bus controllers



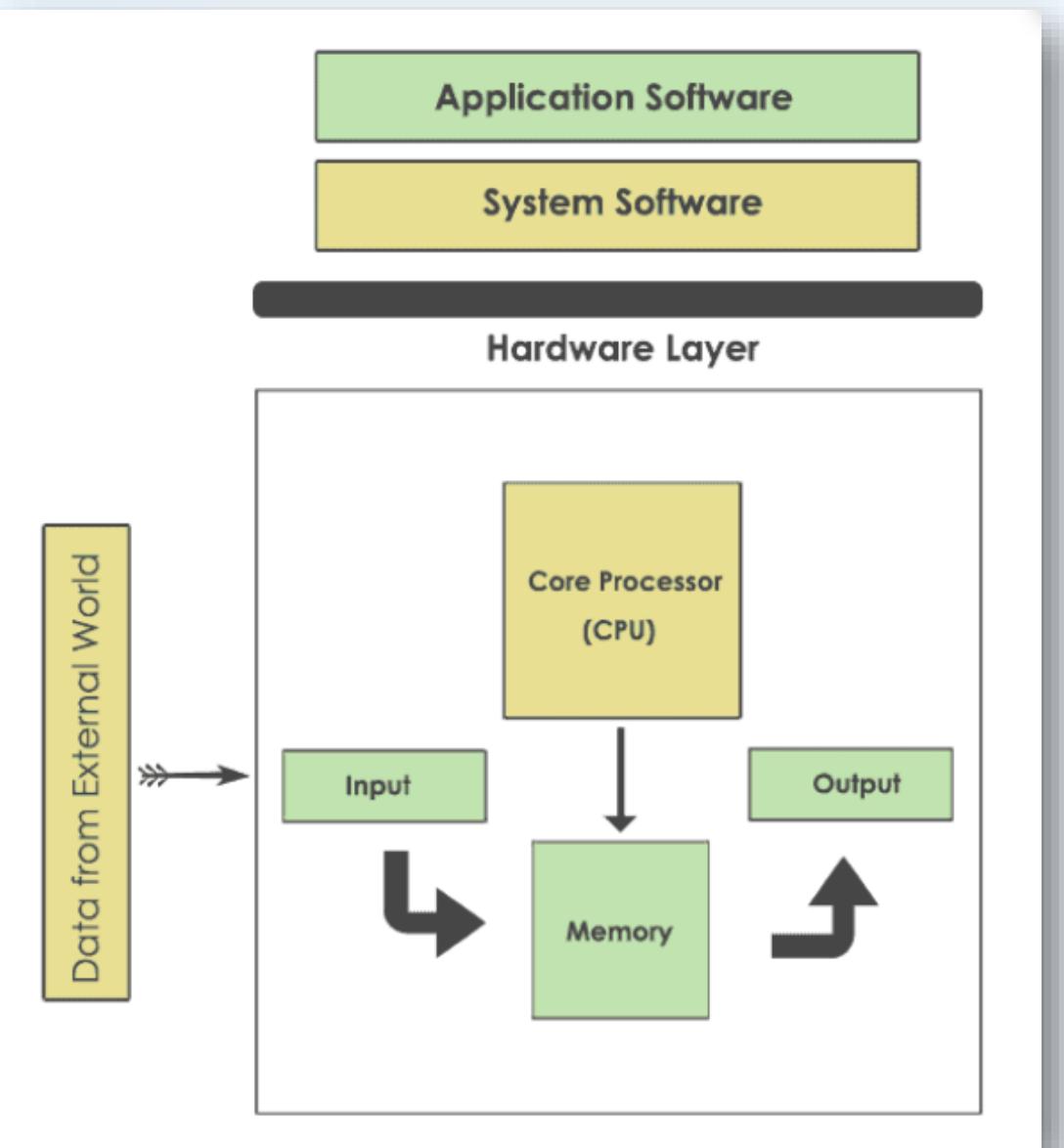
Embedded Software



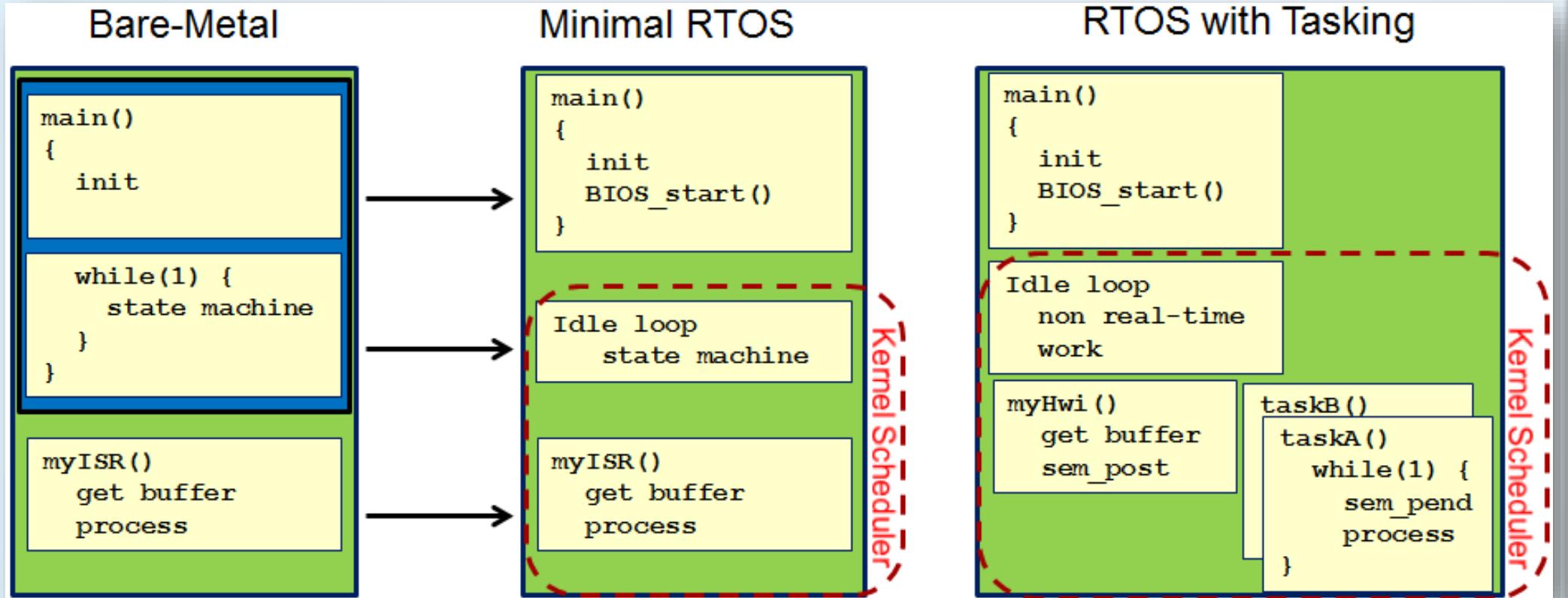
Embedded software is computer software, written to control machines or devices

A blurred background image showing a computer screen displaying code, likely related to embedded software development. The code includes various functions and variables, such as 'operator', 'mirror', 'context', and 'selected' objects, with comments like 'please select ex' and 'OPERATOR CLASSES'.

Embedded Software

A blurred background on the right side of the slide displays a block of computer code in a light blue color scheme. The code appears to be written in Python and relates to 3D rendering or game development, mentioning 'object to mirror', 'mirror_object', 'MIRROR_X', 'mod.use_x', 'mod.use_y', 'mod.use_z', 'context', 'selected', 'operator', 'mirror', 'mirror_to the selected', 'mirror_mirror_x', 'mirror_x', and 'active_object'. The code is partially cut off at the bottom.

Embedded Software Type



Embedded Software Development Life Cycle



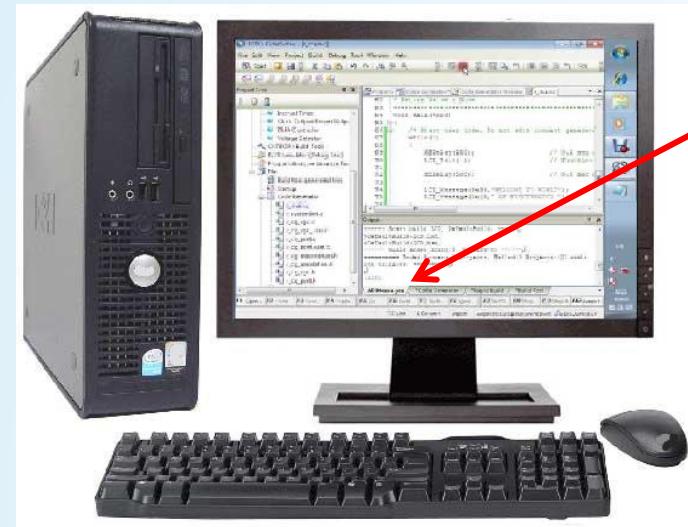
- Step 1: Understand the requirements
- Step 2: Examine
- Step 3: Design
- Step 4: Develop
- Step 5: Test
- Step 6: Deploy
- Step 7: Support and Upgrade



Development

Tools user's guide

- Getting Started
- Release Notes
- Complete Users Guide Selection



Board Books

- Development board Reference manual
- Getting started
- Quick Start Guide
- Schematic
- User-manual
-

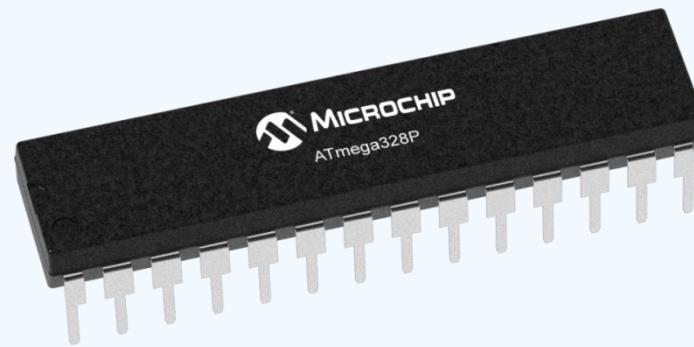


Device Data Books

- Data sheet
- Schematic
- Reference manual
- AN
- -----

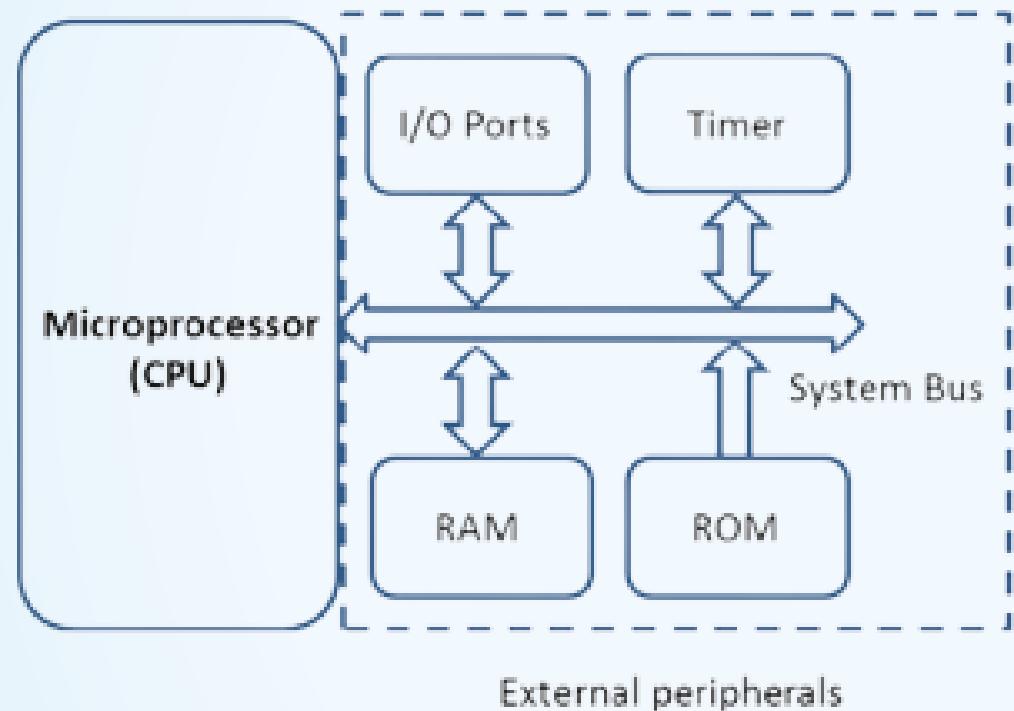
What's a Microcontroller?

A Microcontroller or an **MCU** is a small computer that contains **RAM**, **ROM**, **Flash Memory**, and other **Peripherals** all in a single **Integrated Circuit (IC)**.

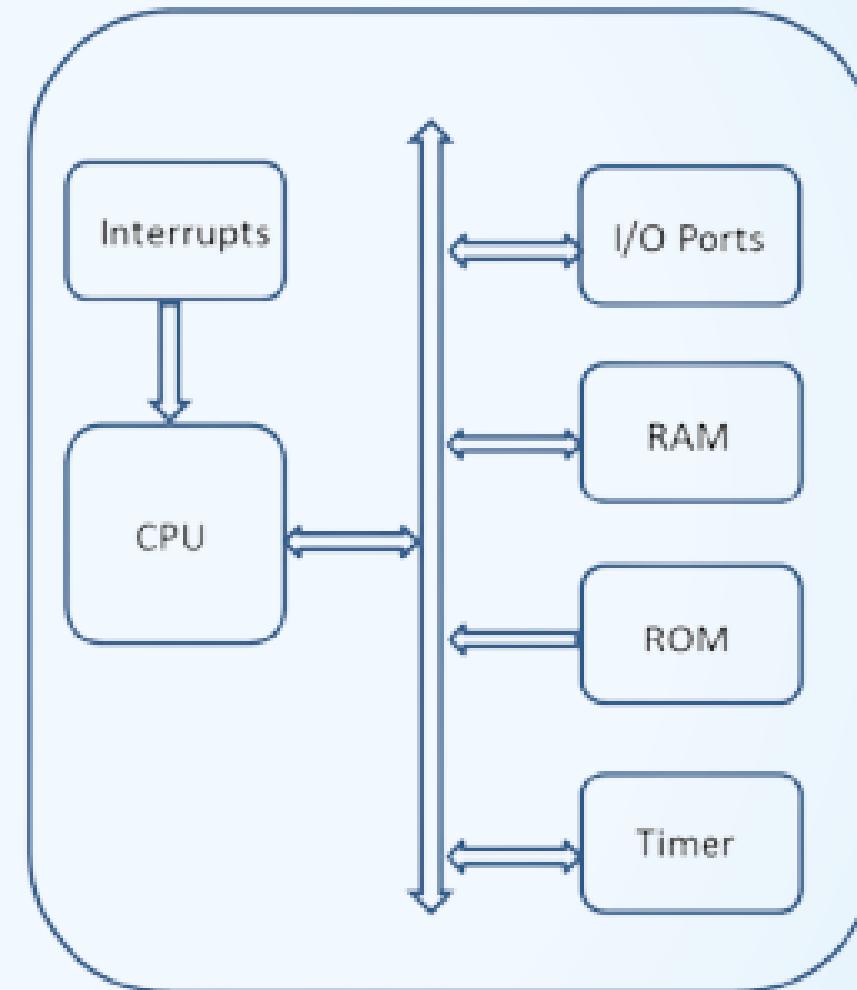


A Microcontroller have **very limited hardware!**
So they generally run a **Single Program** and are used for
Specific Purposes applications.

MPU vs MCU



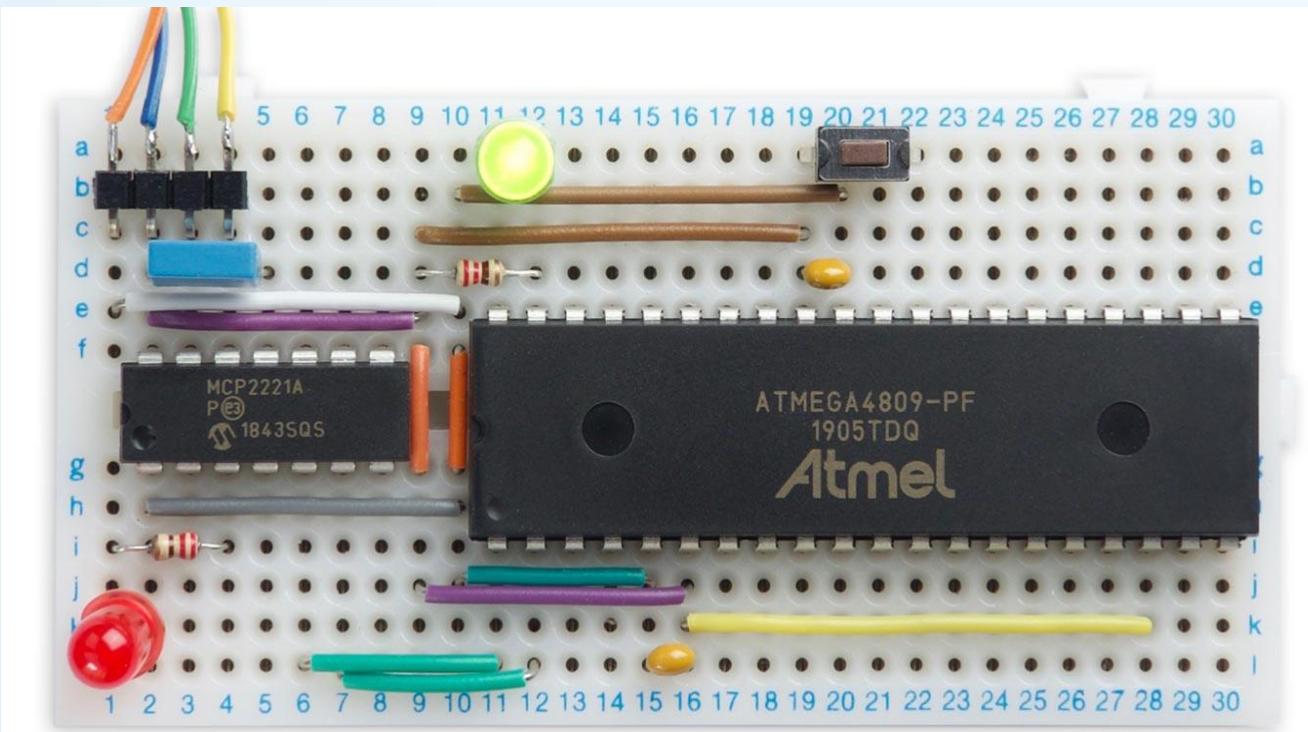
Micro-Processor



Micro-Controller

What's a Development Board?

The **Dev Board** is meant for hobbyists and for people who don't have background in **Electronics**. Without a development board, you'll have to do this:

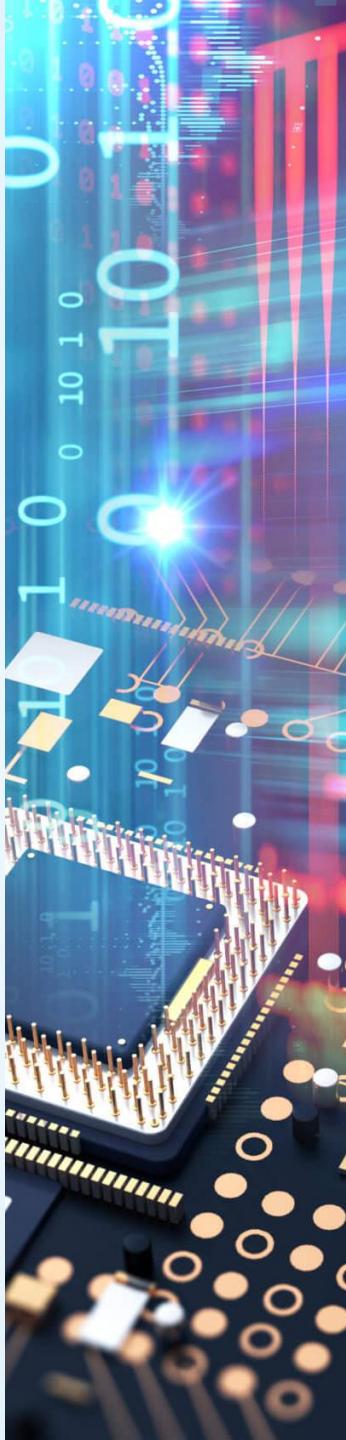
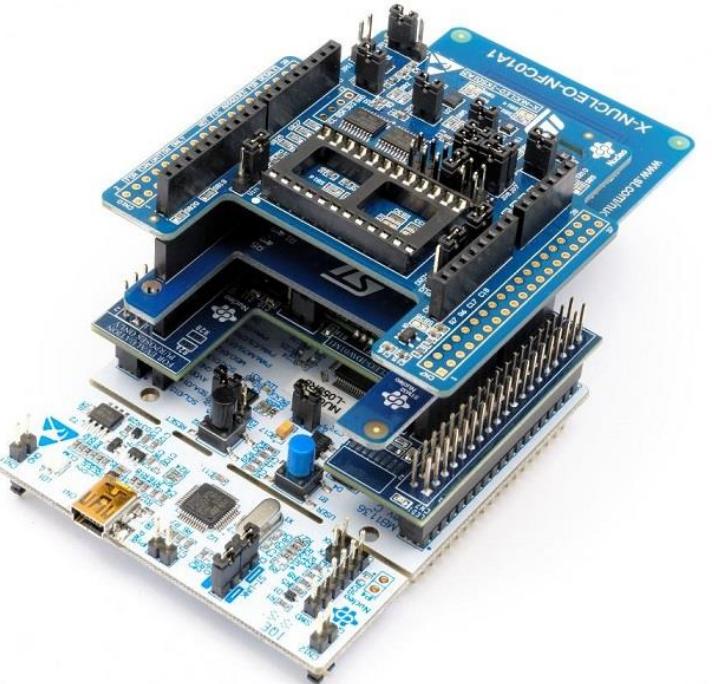
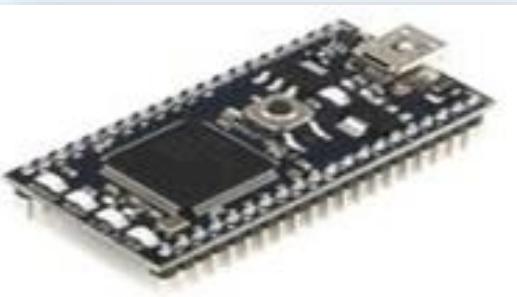


Which is already a too
Steep Learning Curve
for beginners.

Fast Prototyping



This concept of rapid prototyping based on the reuse of what already exists helps designers and system integrators to develop ideas through assembled modules (shield) on the basis of a Prototype Board



STM32 Hardware Toolkit



STM32 Nucleo
development boards

Flexible
prototyping



Discovery kits

Key feature
prototyping



Evaluation boards

Full feature
evaluation



STM32 Nucleo
expansion boards

Add-on
functionalities



Accessories for
STM32 boards

Add-on
functionalities



Partners
boards

From full
evaluation to
open hardware

STM32 Hardware Toolkit



STM32 MCUs 32-bit Arm® Cortex®-M



| High Performance | STM32F2 | STM32F4 | STM32H5 | STM32F7 | STM32H7 |
|------------------|-----------------------------------|-----------------------------------|---|------------------------------------|---|
| | 398 CoreMark 120 MHz Cortex-M3 | 608 CoreMark 180 MHz Cortex-M4 | Up to 1023 CoreMark 250 MHz Cortex-M33 | 1082 CoreMark 216 MHz Cortex-M7 | Up to 3224 CoreMark Up to 550 MHz Cortex-M7 240 MHz Cortex-M4 |

| Mainstream | STM32G0 | STM32G4 | STM32C0 | STM32F0 | STM32F1 | STM32F3 |
|------------|-----------------------------------|-----------------------------------|-----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| | 142 CoreMark 64 MHz Cortex-M0+ | 569 CoreMark 170 MHz Cortex-M4 | 114 CoreMark 48 MHz Cortex-M0+ | 106 CoreMark 48 MHz Cortex-M0 | 177 CoreMark 72 MHz Cortex-M3 | 245 CoreMark 72 MHz Cortex-M4 |

● Optimized for mixed-signal applications

| Ultra-low-power | STM32L4+ | STM32U5 | STM32L0 | STM32L4 | STM32L5 |
|-----------------|-----------------------------------|------------------------------------|----------------------------------|----------------------------------|------------------------------------|
| | 409 CoreMark 120 MHz Cortex-M4 | 651 CoreMark 160 MHz Cortex-M33 | 75 CoreMark 32 MHz Cortex-M0+ | 273 CoreMark 80 MHz Cortex-M4 | 443 CoreMark 110 MHz Cortex-M33 |

| Wireless | STM32WL | STM32WB | STM32WBA |
|----------|---|---|------------------------------------|
| | 162 CoreMark 48 MHz Cortex-M4 48 MHz Cortex-M0+ | 216 CoreMark 64 MHz Cortex-M4 32 MHz Cortex-M0+ | 407 CoreMark 100 MHz Cortex-M33 |

● Cortex-M0+
Radio co-processor

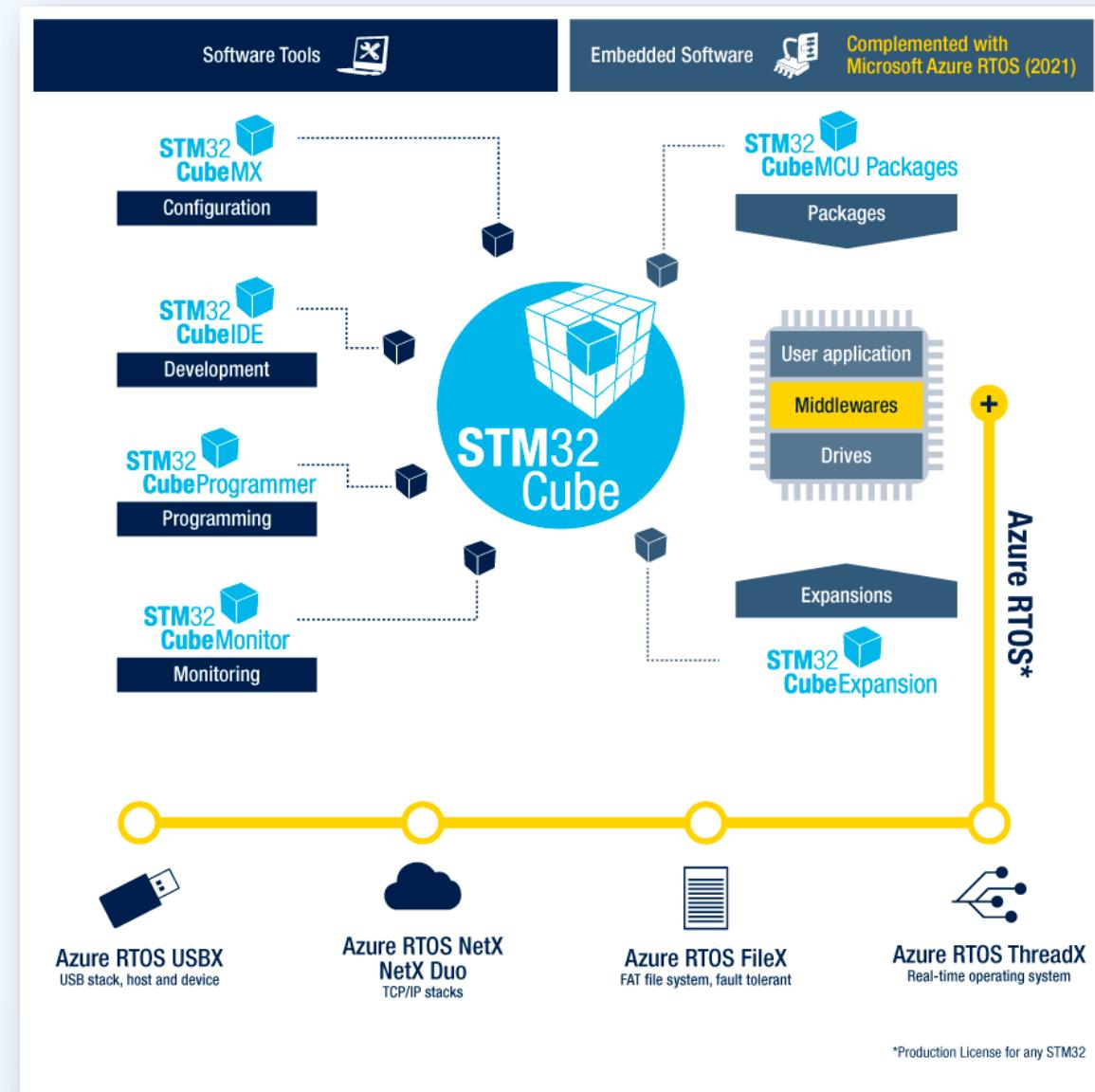
STM32 Hardware Toolkit



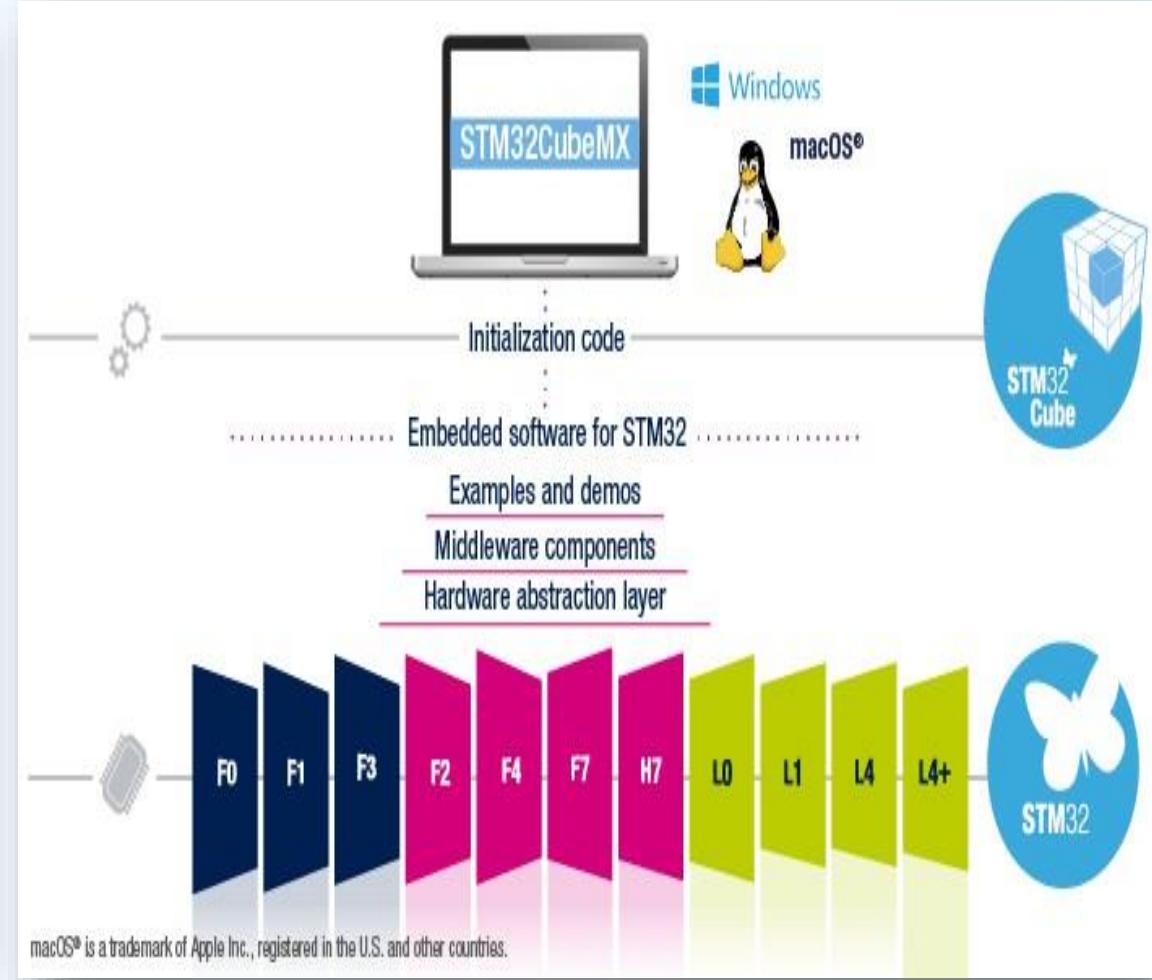
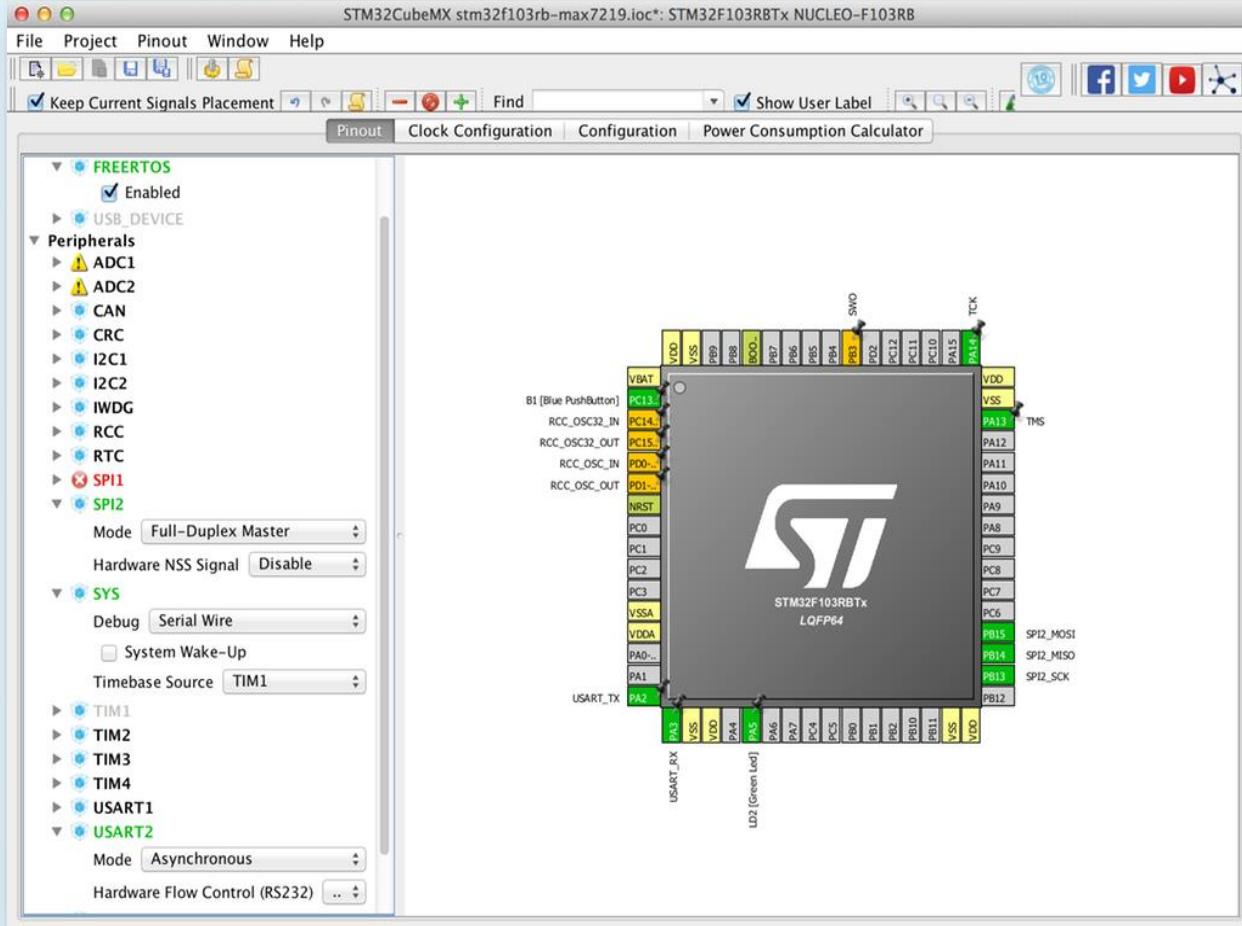
STLINK-V3 probe for MCUs
multipath debugger/programmer



STM32 Software Toolkit



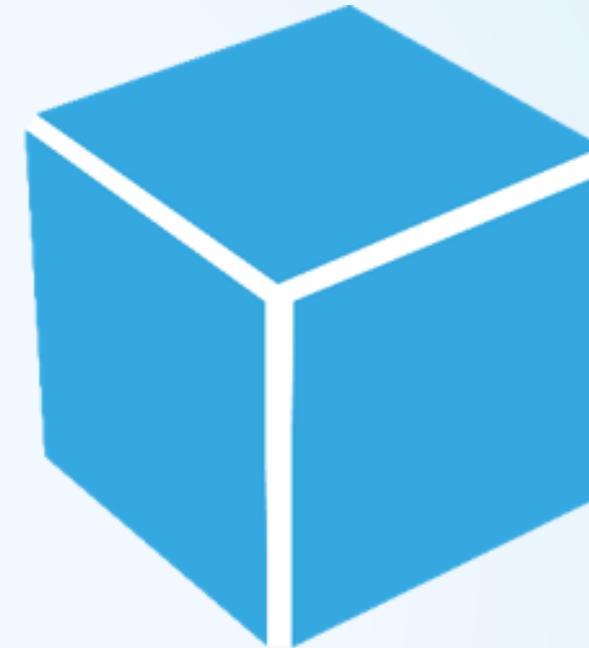
STM32 Software Toolkit



STM32 Software Toolkit



STM32
CubeIDE

A small blue silhouette of a butterfly with its wings spread, positioned above the 'STM32' text.A large, solid blue cube with white edges, positioned to the right of the 'CubeIDE' text.

STM32 Software Toolkit



Runtime monitoring tool for STM32
helps visualize any application behavior



STM32 Software Toolkit



Other MCUs

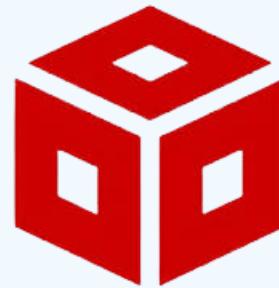


Other MCUs



Softopc.com

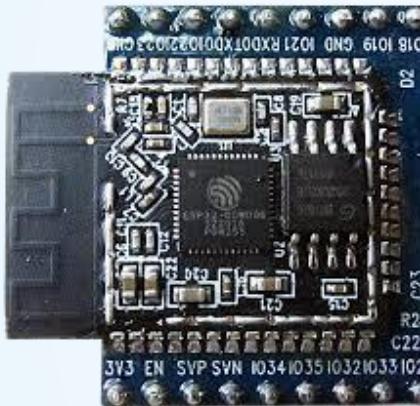
Other MCUs



Code Composed Studio



Other MCUs



 **ESPRESSIF**





Questions



Legenda

Slides including following brain symbol are purely theoretical ones



Optional steps during development in Labs are marked with a grey bar



Yellow bar in Lab#2 show differences vs STM8 microcontrollers

STM32F103RCT6

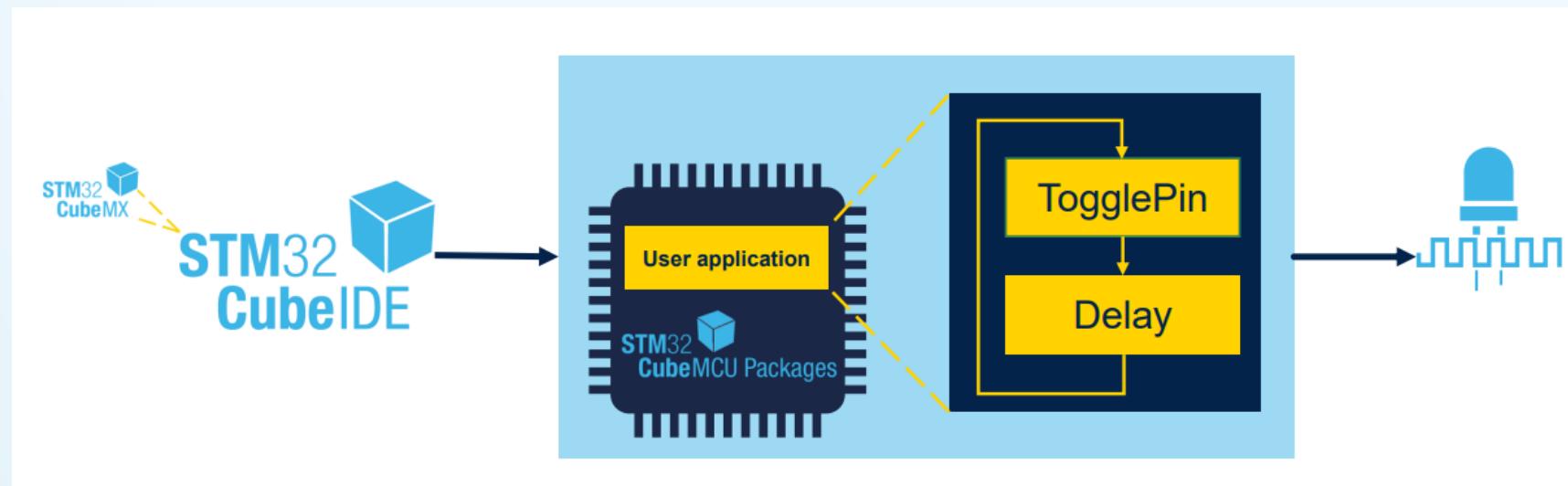
Source code for development is included inside pink boxes

HAL_Delay(1000);

LAB#1 LED Blinking in 6 Clicks

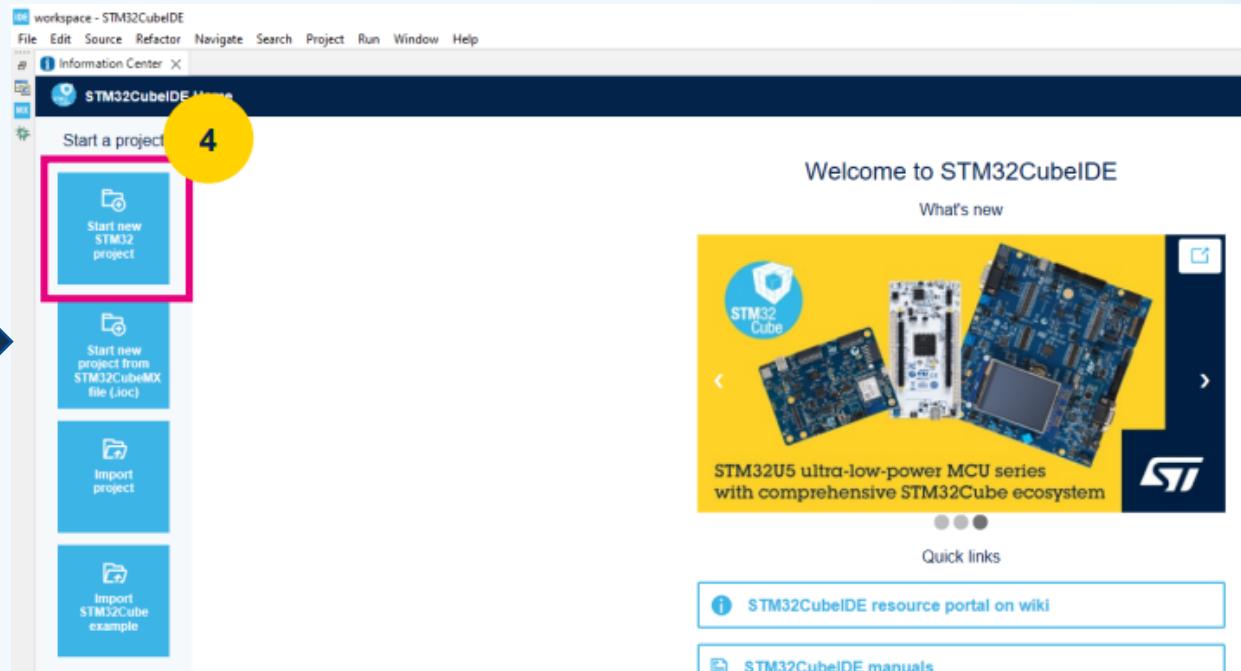
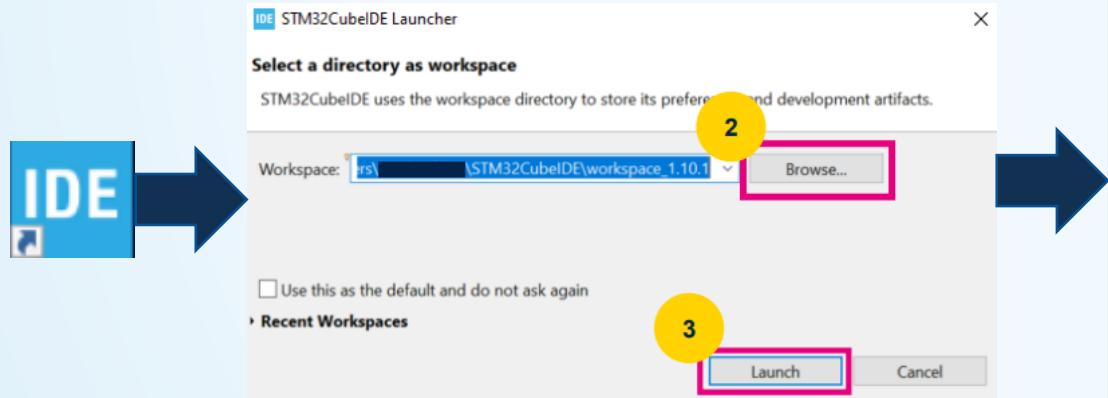


- This short hands-on will show you how to start a project for STM32 kit from scratch using only STM32CubeIDE
- The target is to get a running blinking LED in a few clicks



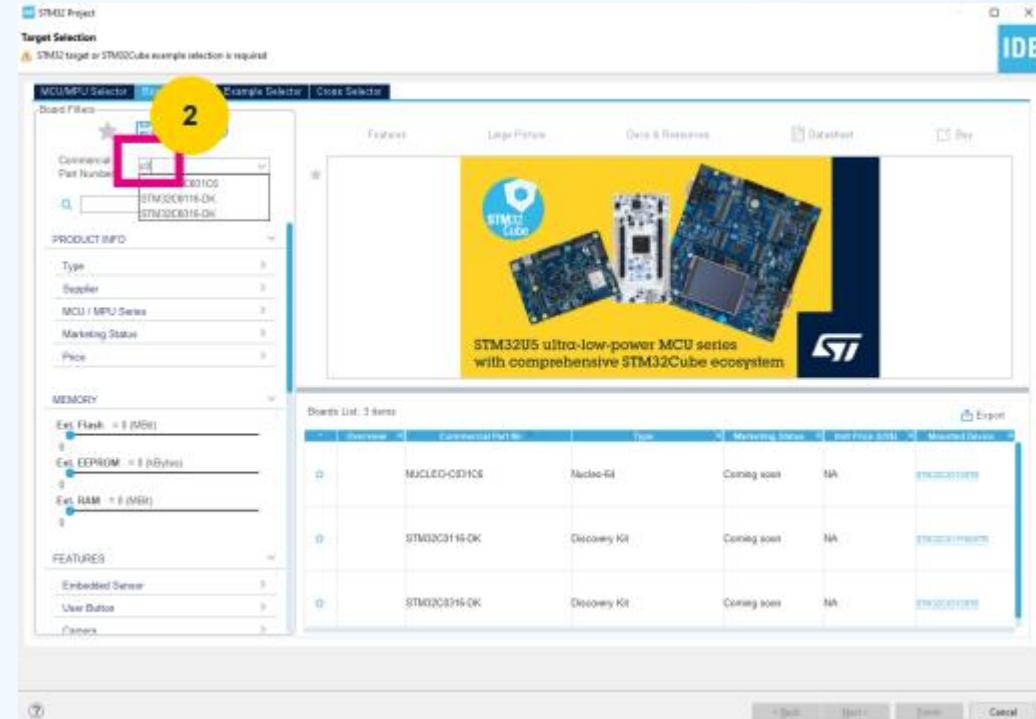
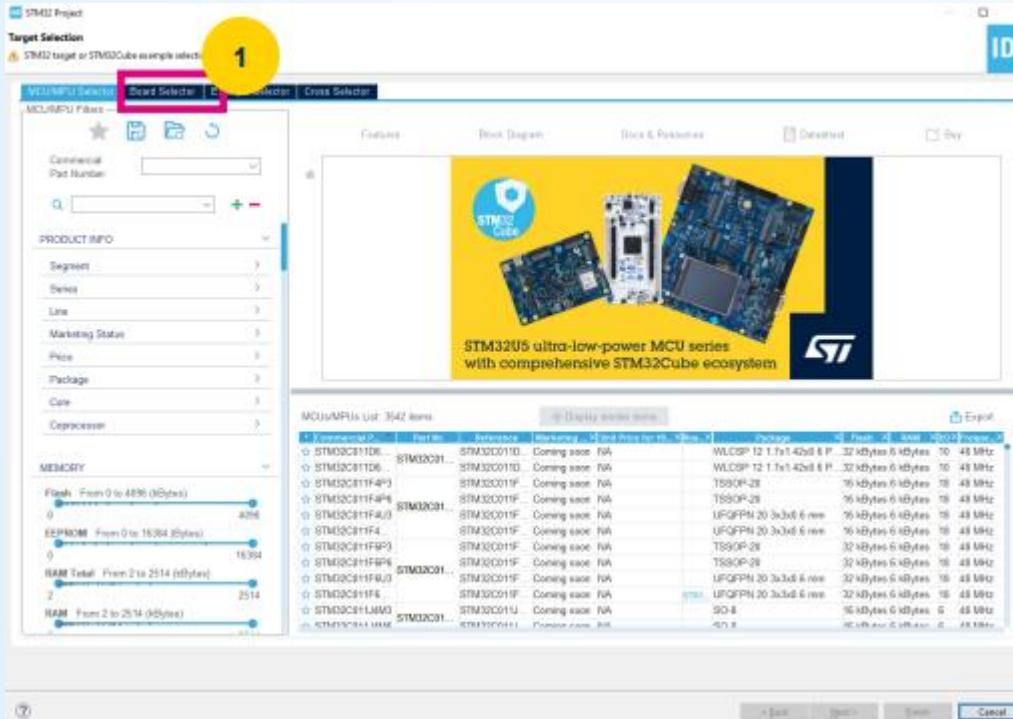
LAB#1 LED Blinking in 6 Clicks

1. Launch STM32CubeIDE.
2. Select a new workspace.
3. Click **Launch**.
 - Steps 2 and 3 are usually optional: only needed after new workspace creation.
4. Click **New STM32 Project**.



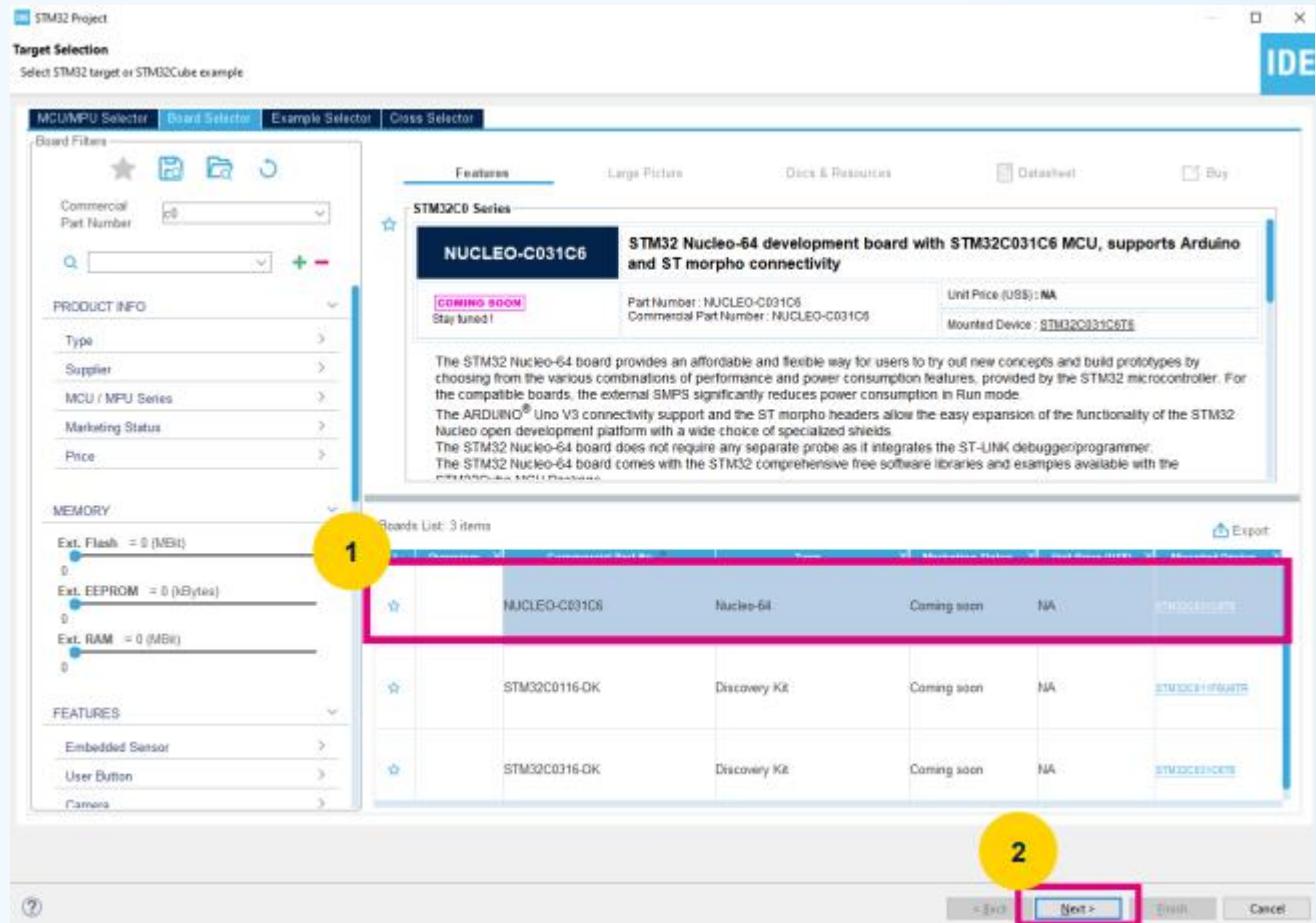
LAB#1 LED Blinking in 6 Clicks

1. Click on Board Selector.
2. Type your board name in Commercial Part Number search box.



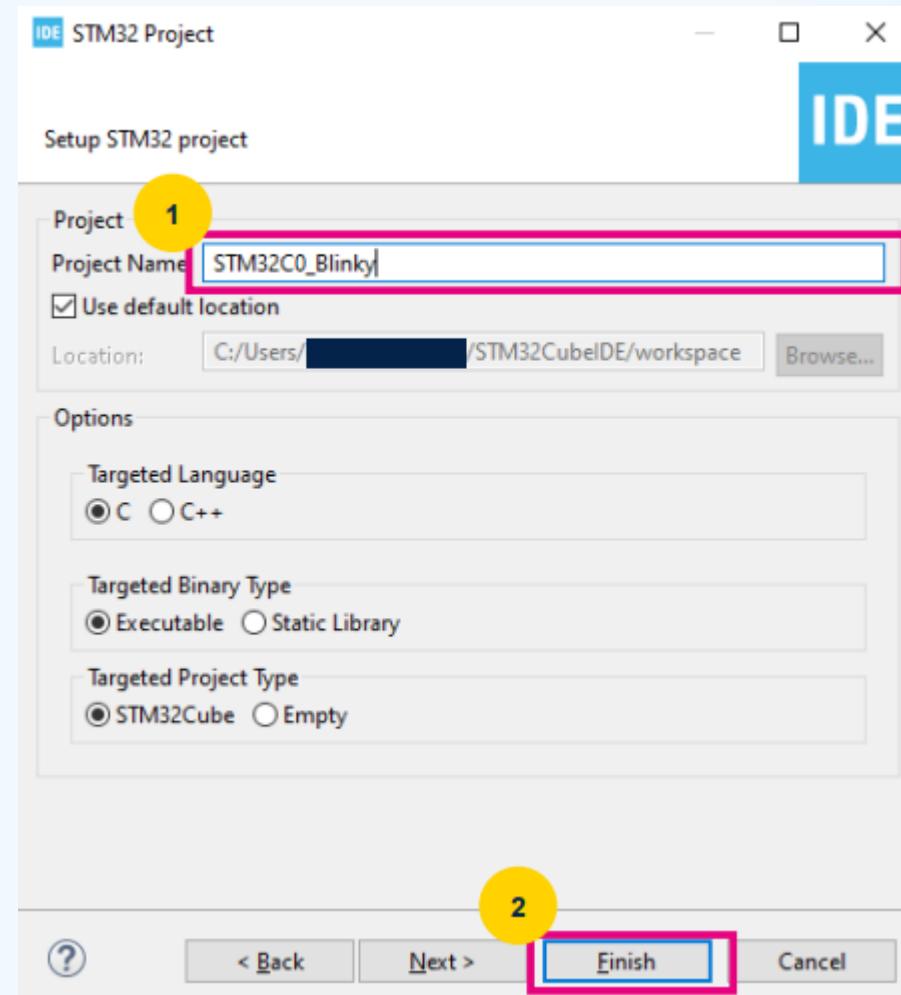
LAB#1 LED Blinking in 6 Clicks

1. Select your board or your chip.
2. Click **Next**.



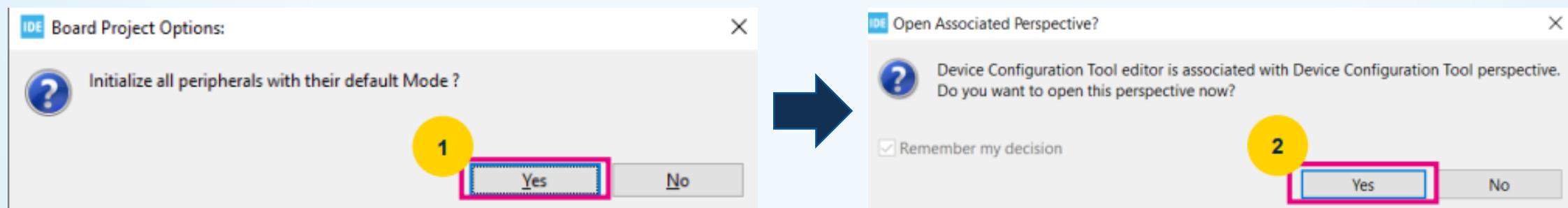
LAB#1 LED Blinking in 6 Clicks

1. Give a name to your project.
 - Here “STM32C0_Blinky”.
2. Click **Finish**.



LAB#1 LED Blinking in 6 Clicks

1. Click **Yes** in pop-up asking of peripherals initialization.
 - This will setup all STM32C0 GPIOs to work with the NUCLEO-C031C6 (LED, Button, etc).
2. Click **Yes** in pop-up asking to open device configuration perspective.
 - This will create your project structure with all needed files and open the STM32CubeMX embedded in STM32CubeIDE.
 - This pop-up may not appear if already acknowledged with “Remember my decision” flag.



LAB#1 LED Blinking in 6 Clicks

What you got after 6 clicks

Project Structure

- Core (Src, Inc & Startup)
 - User's application inside
 - We will work with main.c
- Drivers (CMSIS & HAL)

IOC File

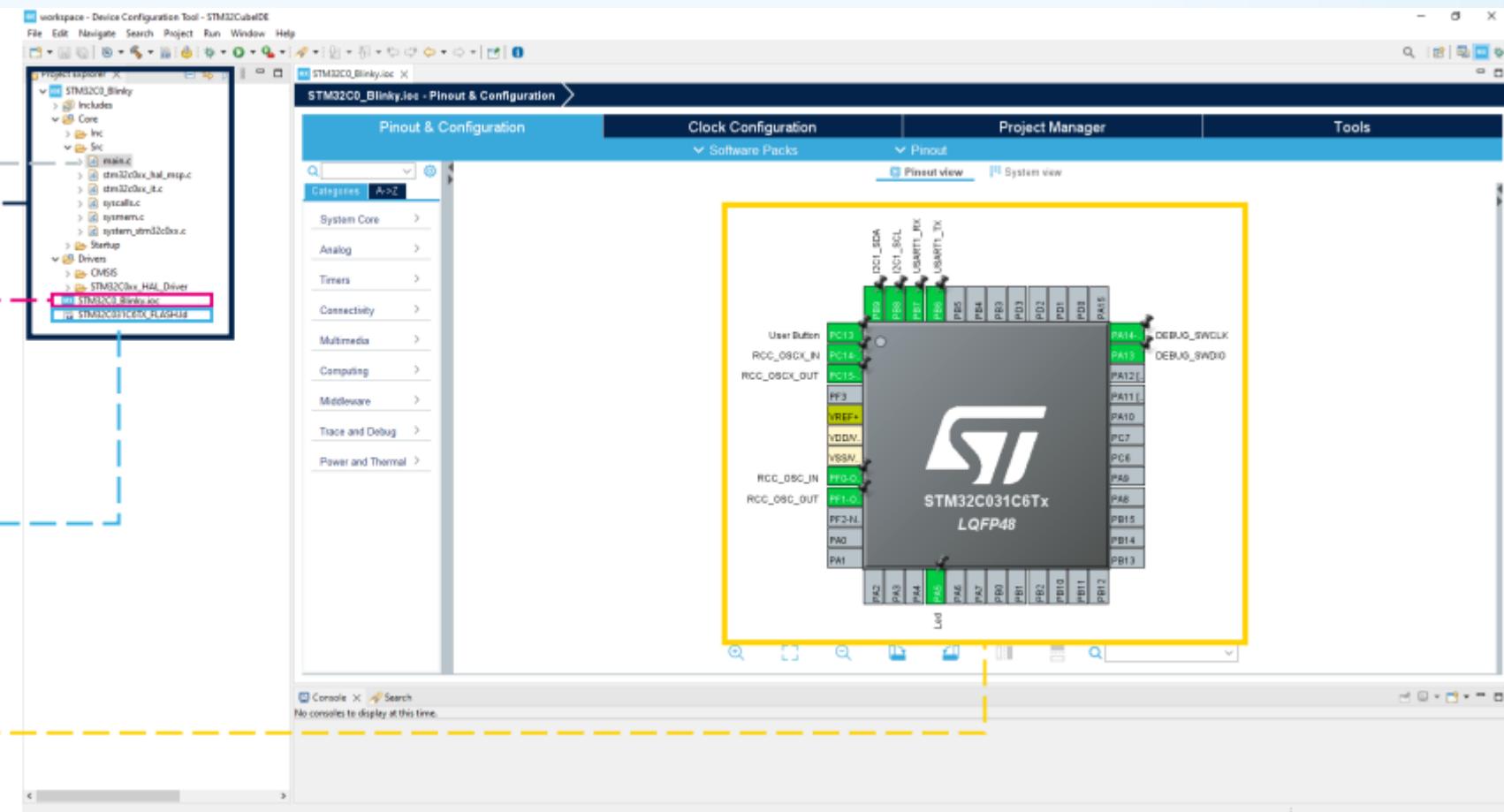
- Already open at startup
- STM32CubeMX plugin into STM32CubeIDE

Linker File

- Handles placement in STM32C0 memories

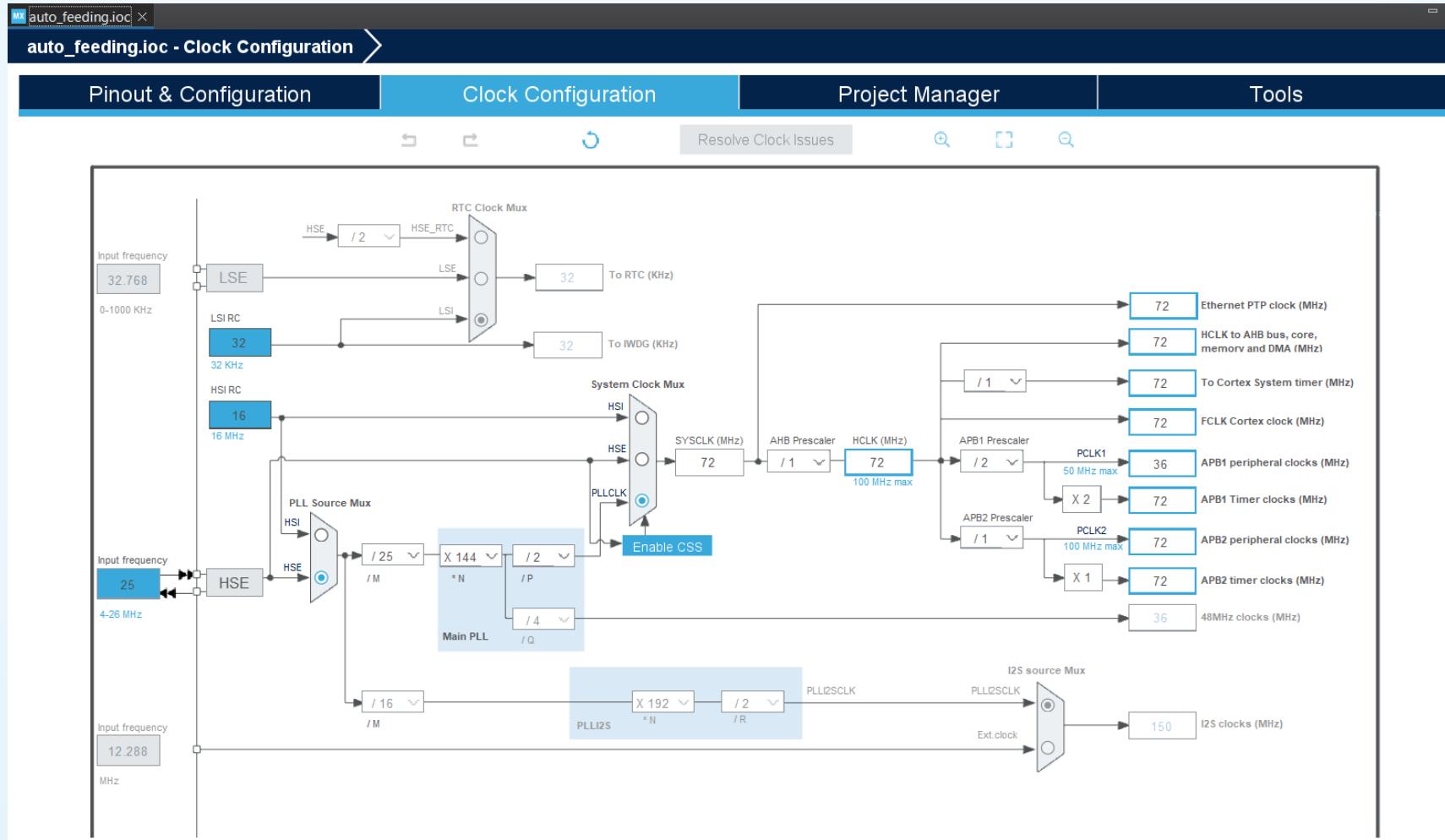
STM32C0 Pinout

- LED and Push-Button
- HSE and LSE Oscillators
- UART and SWD



LAB#1 LED Blinking in 6 Clicks

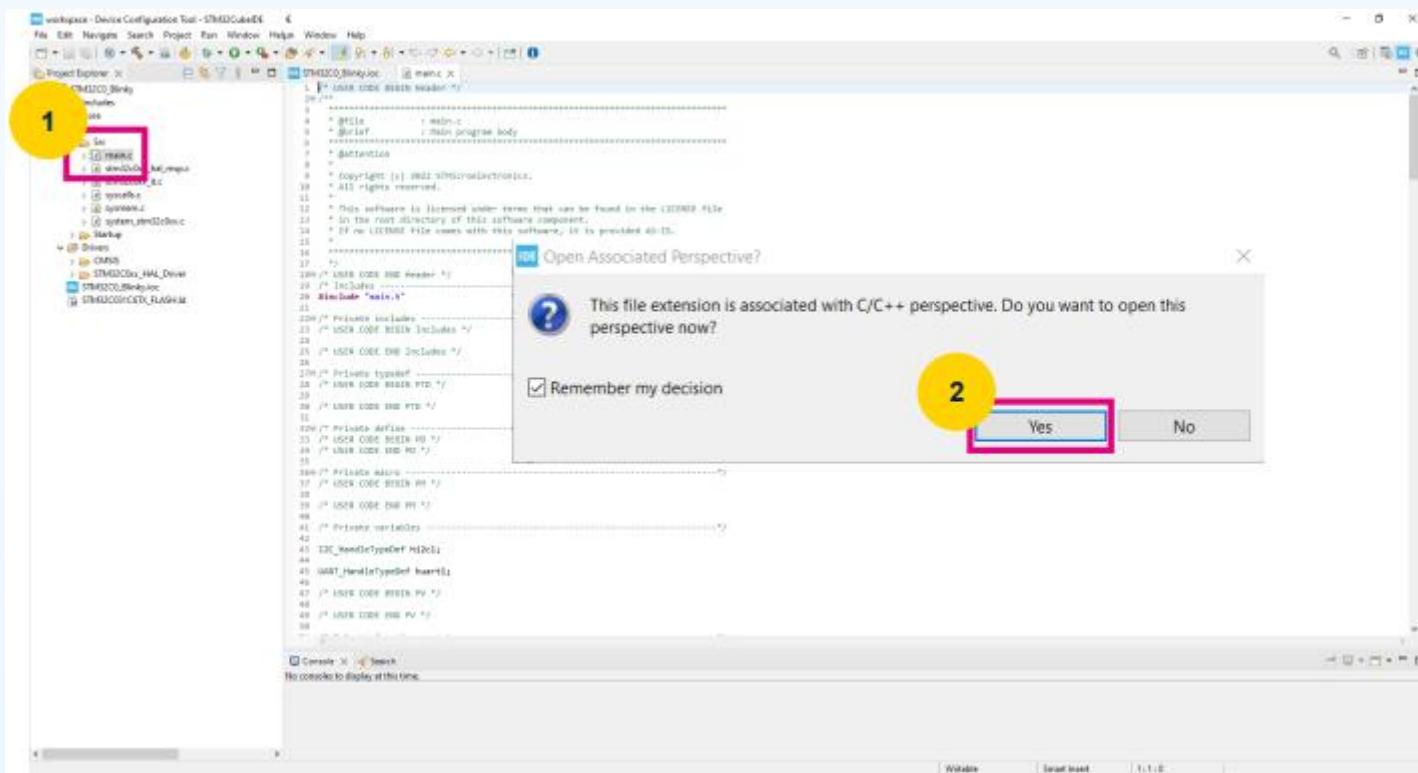
Clock configuration



LAB#1 LED Blinking in 6 Clicks



1. Double Click on **main.c**
 2. Click **Yes** in pop-up asking to change perspective.
 - This pop-up may not appear if already acknowledged with “Remember my decision” flag



LAB#1 LED Blinking in 6 Clicks

1. Scroll the main.c file till the USER CODE BEGIN/END WHILE section.
 - Source code **within** “USER CODE BEGIN WHILE” and “USER CODE END WHILE” section **will be preserved** after regeneration.
2. Add code inside the USER CODE BEGIN/END WHILE section.
 - HAL_Delay uses STM32C0 SysTick peripheral, which is enabled to fire every 1s.
 - **Without** a delay, it's impossible to see GPIO toggle in run mode: only in debug stepping mode

1

```
/* Infinite loop */  
/* USER CODE BEGIN WHILE */  
while (1)  
{  
    /* USER CODE END WHILE */  
  
    /* USER CODE BEGIN 3 */  
}
```



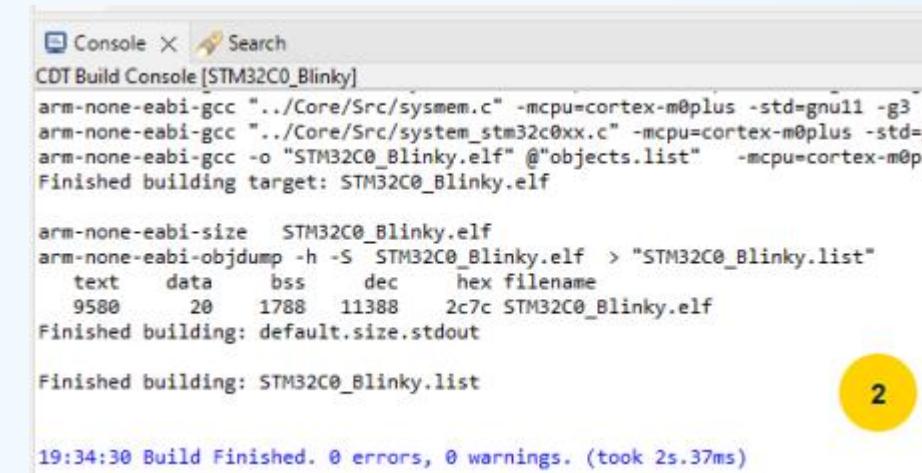
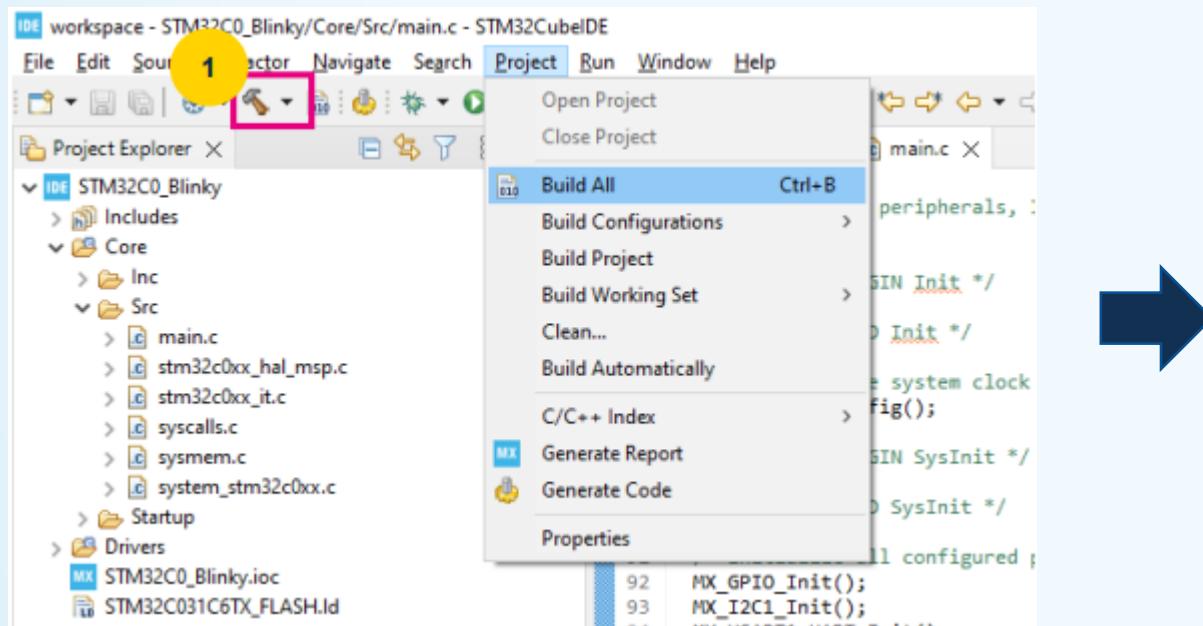
2

```
99  /* Infinite loop */  
100 /* USER CODE BEGIN WHILE */  
101  file (1)  
102  
103  HAL_GPIO_TogglePin(Led_GPIO_Port, Led_Pin);  
104  HAL_Delay(500);  
105  /* USER CODE END WHILE */  
  
106  
107  /* USER CODE BEGIN 3 */  
108 }  
109 /* USER CODE END 3 */
```

LAB#1 LED Blinking in 6 Clicks

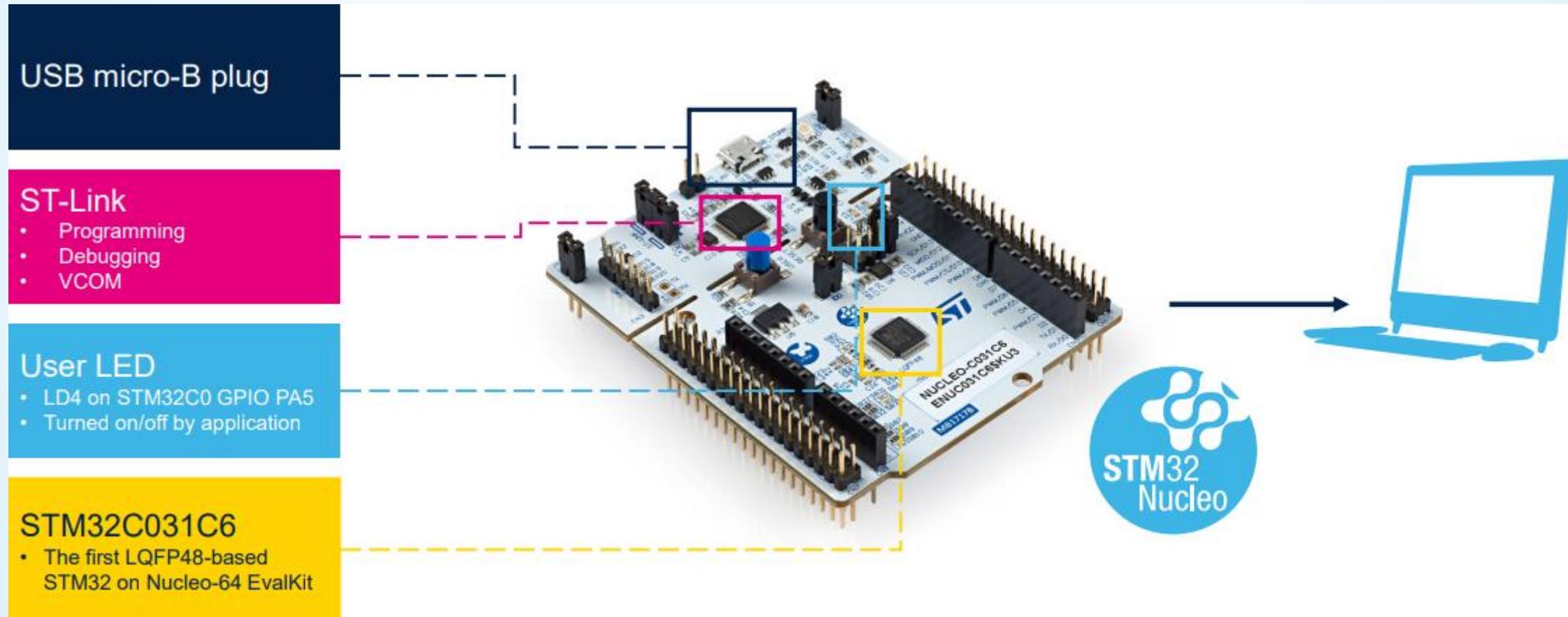


1. Click on Hammer icon, **or** press Ctrl + B, navigate to project → Build All.
 2. Check no error on Console.



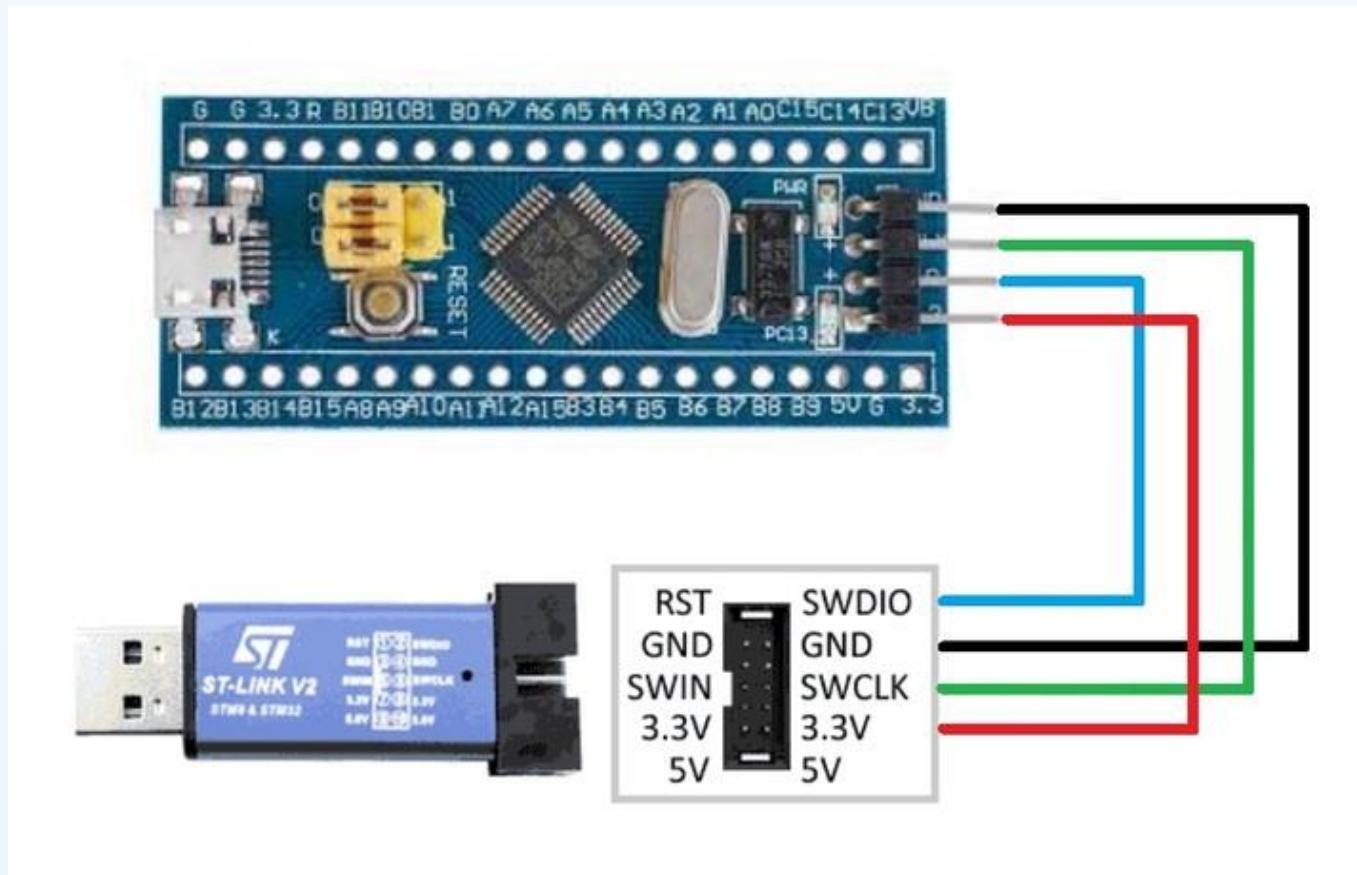
LAB#1 LED Blinking in 6 Clicks

Connect your board to your laptop.



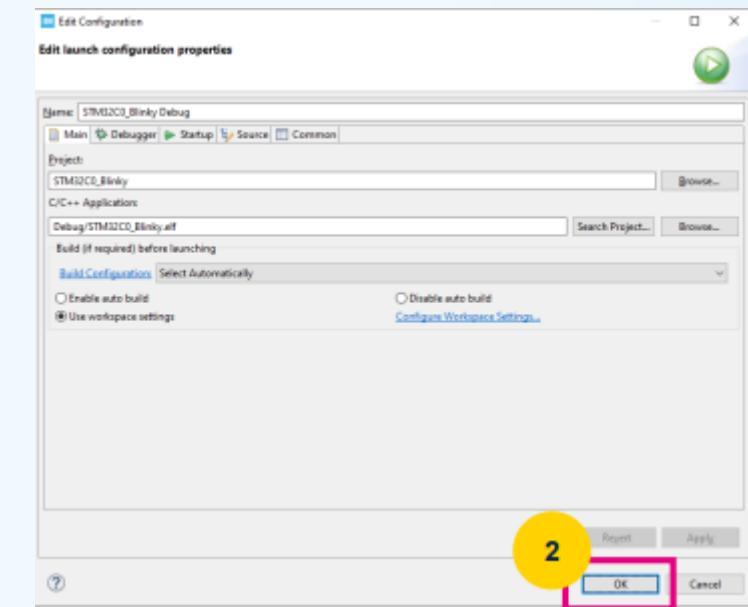
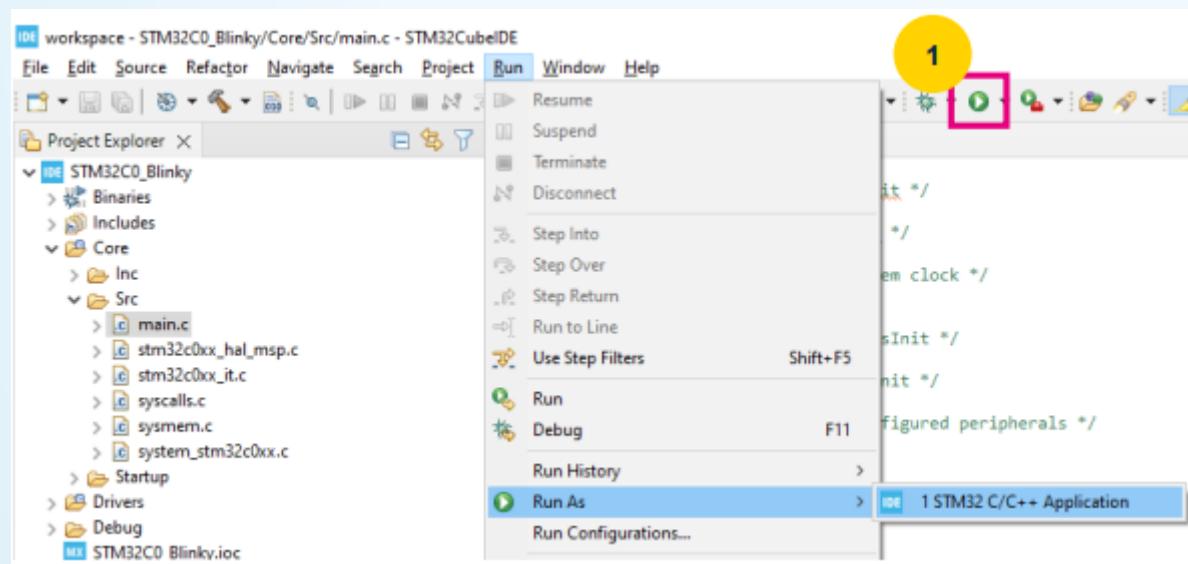
LAB#1 LED Blinking in 6 Clicks

1. Connect the SWD pins to the STM32 board.
2. Connect the STLink to your lapyop.



LAB#1 LED Blinking in 6 Clicks

1. Click on Play icon, or navigate to Run → Run as → STM32 C/C++ Application.
2. Click **OK** in Edit Configuration window.
 - This window will not appear in the future for same project.
 - To change project configuration in the future, click on down arrow, or navigate Run menu.



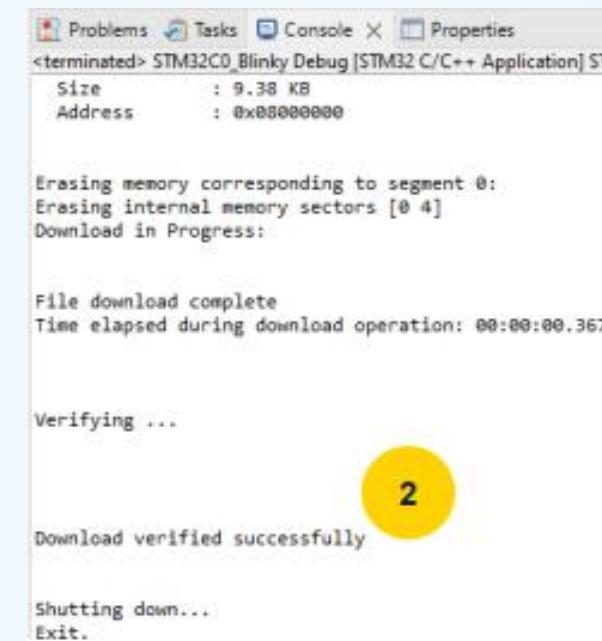
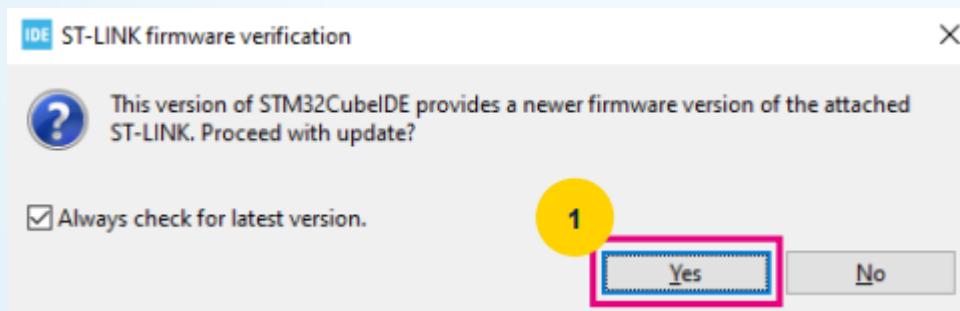
LAB#1 LED Blinking in 6 Clicks

1. Click Yes in pop-up asking to upgrade the ST-Link FW. This pop-up may not appear if “Always check...” flag is not set, or ST-Link FW is already updated.

- Follow instructions:

1. Click **Open in upgrade mode**
2. Click **Upgrade**.
3. Click the pop-up.
4. Get back to previous slide to start again the Download process.

2. Check no errors on Console.



```
Problems Tasks Console X Properties
<terminated> STM32CubeIDE Blinky Debug [STM32 C/C++ Application] ST
Size : 9.38 KB
Address : 0x00000000

Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 4]
Download in Progress:

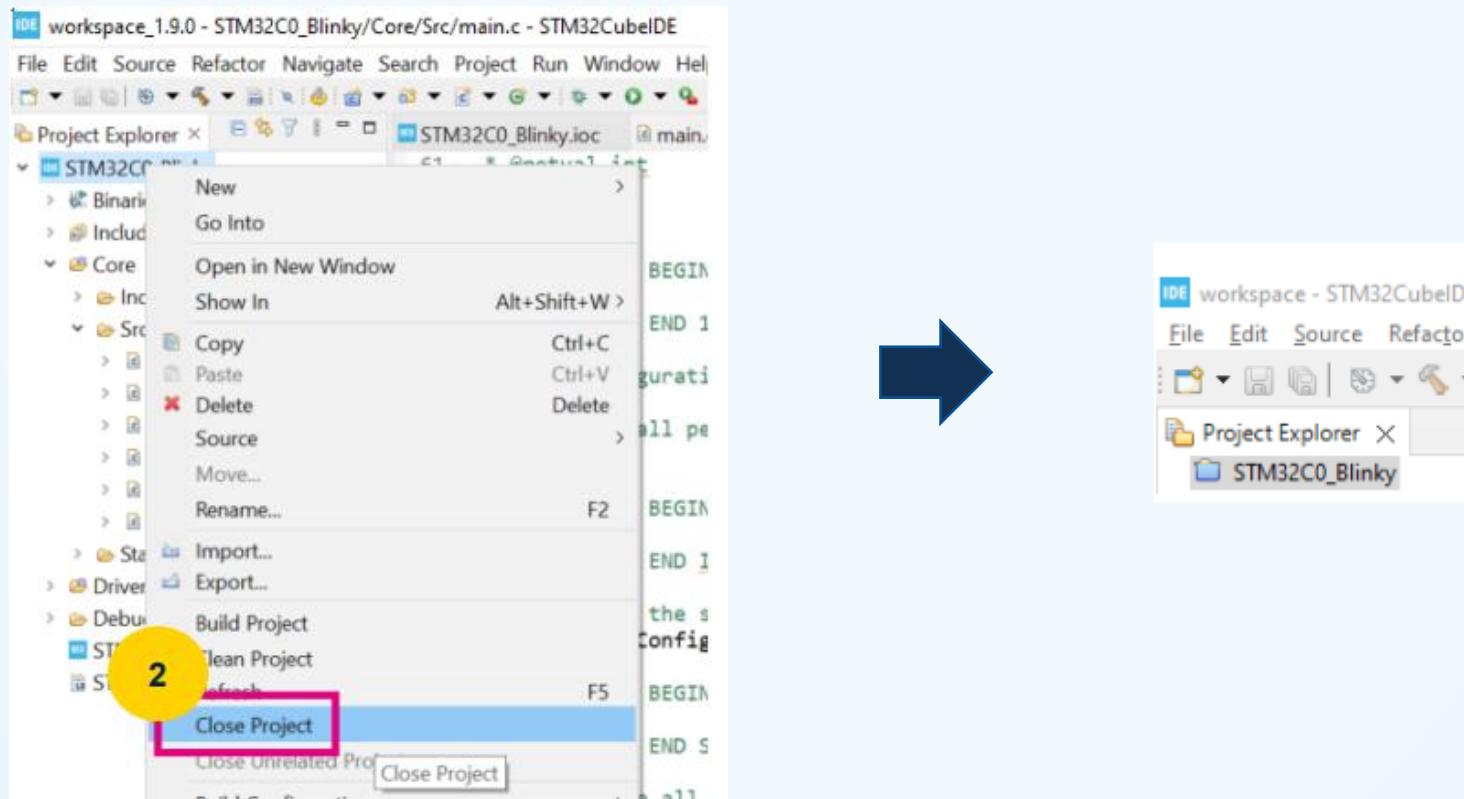
File download complete
Time elapsed during download operation: 00:00:00.367

Verifying ...
Download verified successfully

Shutting down...
Exit.
```

LAB#1 LED Blinking in 6 Clicks

1. Right click on your project name in Project Explore, and click on Close Project
 - Let's move to the LAB#2





THANK
YOU

Follow Us!
Electro Scientific Club