INDIAN INSTITUTE OF TECHNOLOGY PATNA

EC3101: MICROCONTROLLER AND EMBEDDED SYSTEM LAB

Experiment No: 01(STM)

Submitted by:-

Name:- ABHINAV SRIVASTAVA
Roll No:- 2301EE44
Group No:- 2(Thrusday)

Blinking LED using STM32 Nucleo Board

Aim:

To write a program in STM32CubeIDE to blink the onboard LED (LD2) connected to GPIO pin PA5 of STM 32 Nucleo-F103RB board.

Component Required:

STM32 Nucleo-F103RB Development Board
USB Cable for programming and power
STM32CubeIDE software

Theory:

As mentioned earlier, we will learn to use GPIO pins of STM32 Nucleo as digital output pins. For demonstration purposes, in this example, we will control an onboard LED of STM32 Nucleo-F103RB board. This STM32 Nucleo board has an onboard LED with the name of LD2 and it is connected with the PA_5 pin of F103RB as shown below:

This LED blinking example works as mentioned below:

- Onboard LED Turns on for one second which means GPIO PA_5 goes to active high state.
- In next step, LED turns off for one second which means GPIO PA_5 goes to active low state.
- Above two steps keep repeating as shown below.

Procedure:

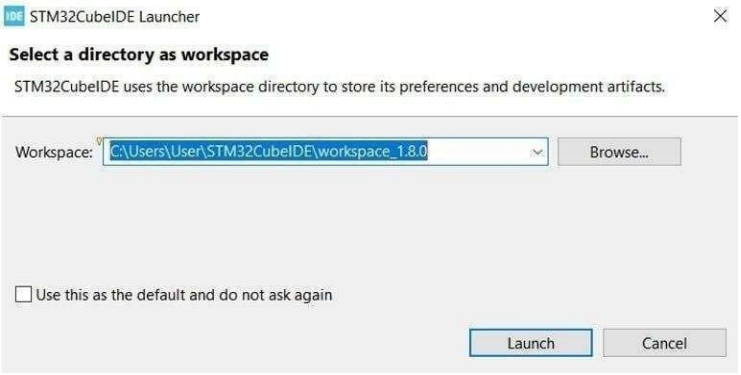
After installing STM32CubeIDE on your Windows, Linux, or MacOS system, follow these steps to create an LED blinking project. In this section, you will learn the followings:

- Create and configure the STM32CubeMX project and generate the initialization code.
- Program and use HAL functions to blink a LED on the NUCLEO-F103RB board.

STEP 1: Launch STM32CubeIDE

Open the STM32CubeIDE application on your computer. You will be asked to specify the directory for the workplace. You can specify the directory and also tick the box below to keep this as the default directory. Next, click ‘Launch’ to start the application.

Step

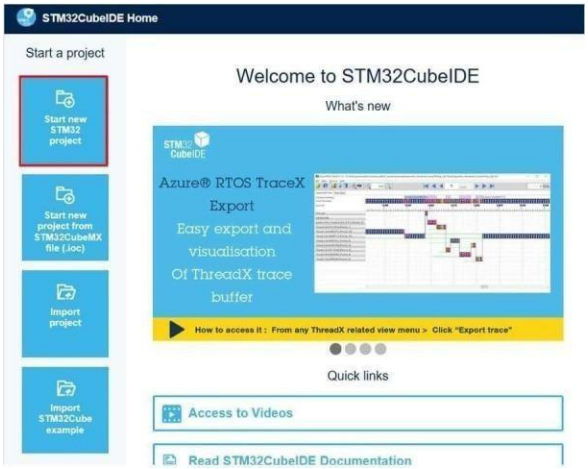


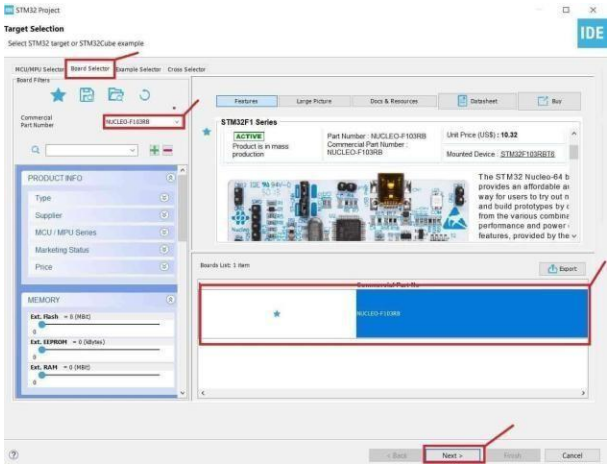
2: Creating STM32 Nucleo Project

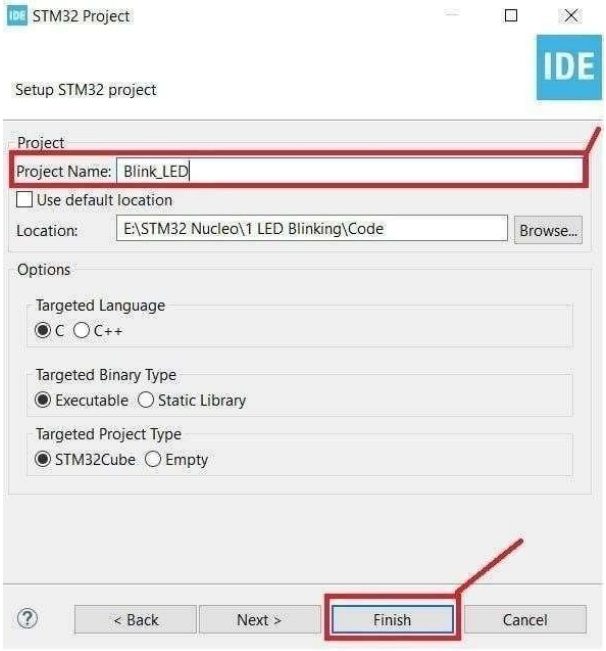
The STM32Cube workspace will open.

- Now click ‘Start new STM32 project.
- From Board Selector section, type STM32 Nucleo-F103RB and select it. Click Next.
- Give your project a name. As we are going to blink the on-board LED of STM32 Nucleo-F103RB hence we have named our project ‘BLINK_LED.’ Then click ‘Finish.

Step







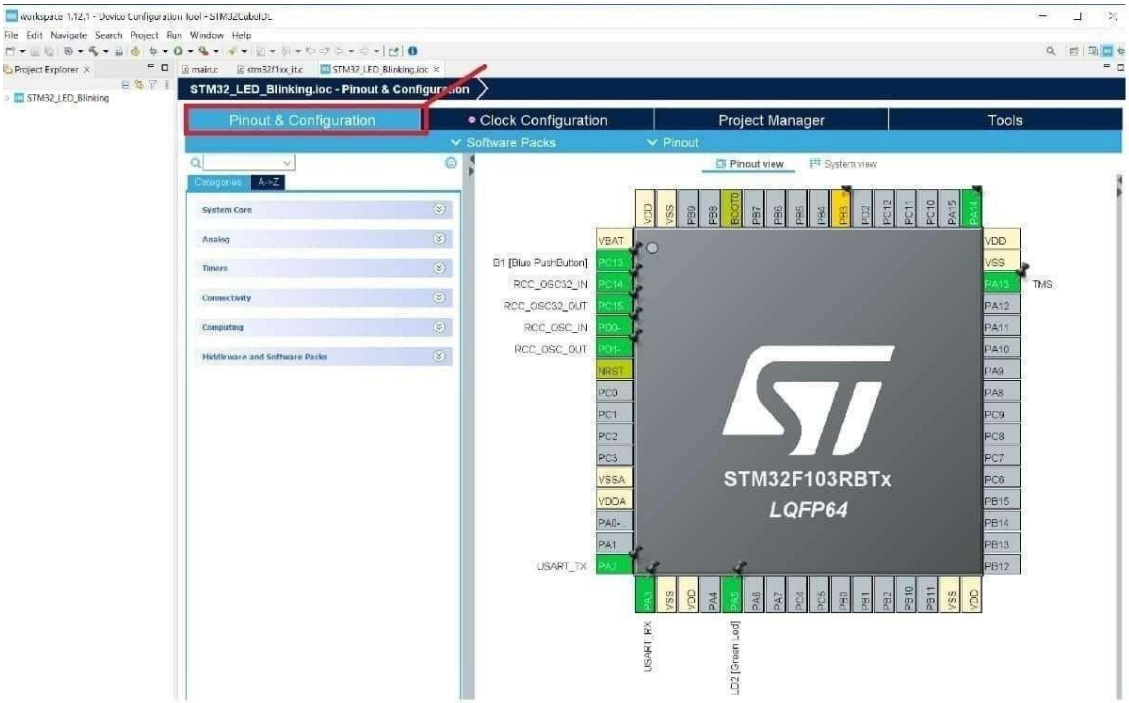
- You will be asked to open an associated perspective to start generating code with STM32CubeMX. Click 'Yes'.

3: Device Configuration Tool

Now the Device Configuration Tool window will open up. In the pinout and configuration window, we can select a specific pin and its function.

Step

A single GPIO pin may have multiple alternate functions. For example, the same GPIO pin can be used for ADC, SPI, UART peripherals, etc. But only one function can be used at a time.



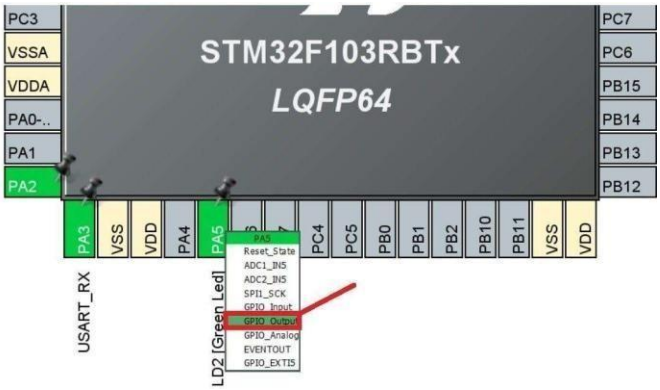
Step 4: Select GPIO Pin

As mentioned earlier, we will see an LED blinking example in this tutorial, and for demonstration purposes, we will use an onboard LED of STM32 Nucleo-F103RB board. The onboard LED of Nucleo-F103RB board is connected with a PA5 pin.

When you create a project for Nucleo-F103RB, you will see that it has already been configured for the PA5 pin. But you can change it according to your requirement.

Click on the PA5 pin, and a list of all alternate functions will appear associated with the PA5 pin. We will set this pin as a 'GPIO Output.'

This is how it will look. Here you can see that we have attached GPIO_Output to PA5.



5: Pin Configuration

Step

Additionally, we have more options for pins as well.

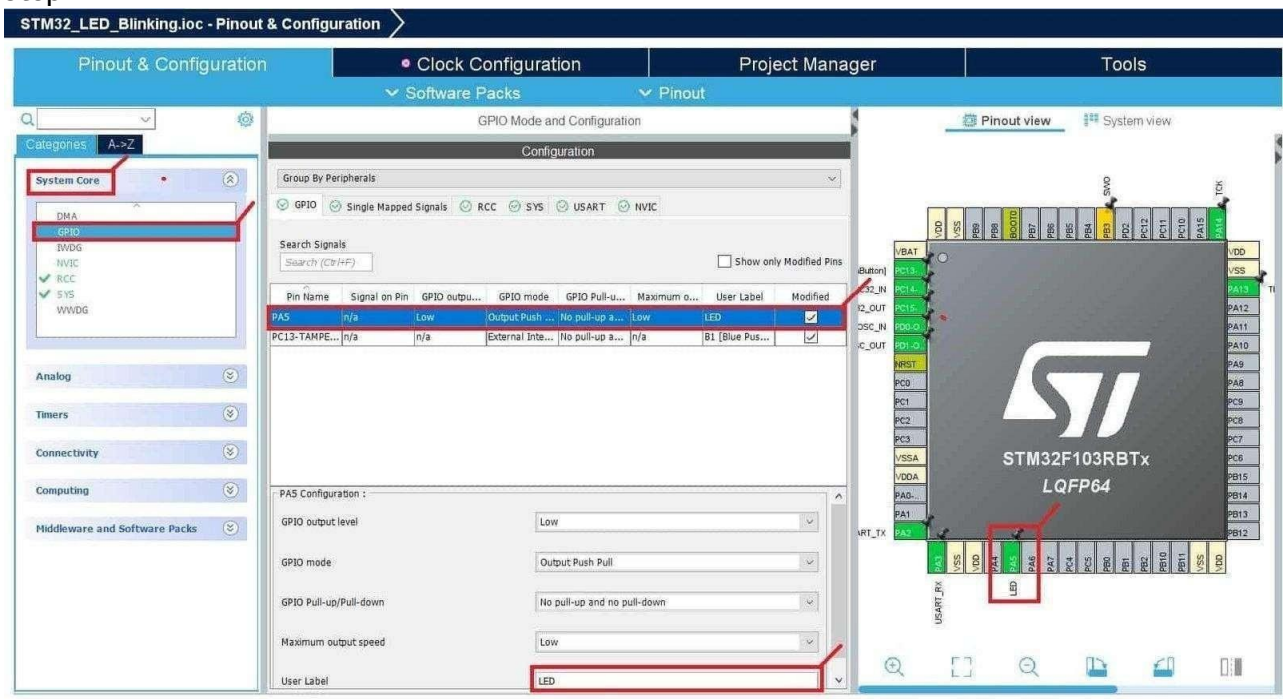
- Go to System Core > GPIO > PA5 and the pin configuration for PA5 will open up.
- Here we have given the pin a user label of ‘LED.’ You can see in Device Configuration Tool, that now PA5 has been given the label ‘LED.’

Apart from the label, there are several other options to choose from including:

- GPIO Output level: By default, the output level is set to Low, meaning the pin is at a low voltage. However, you can change it to High if needed.
- GPIO mode: The mode automatically configures the pins with the appropriate alternate function and sets them to Output Push Pull mode, ensuring compatibility and stability.
- GPIO Pull-up/Pull-down: By default, no pull-up or pull-down resistors are enabled. However, depending on your requirements, you can configure pull-up or pull-down options if supported by the specific pin.
- GPIO Maximum output speed: The maximum output speed is initially set to Low for power consumption optimization. However, if your application requires a higher frequency, you can change this setting accordingly.
- User Label: This is a customizable name assigned to a GPIO pin for easier identification and access. You can assign a meaningful label and later use the Find menu to locate and work with the GPIO pin based on its label.

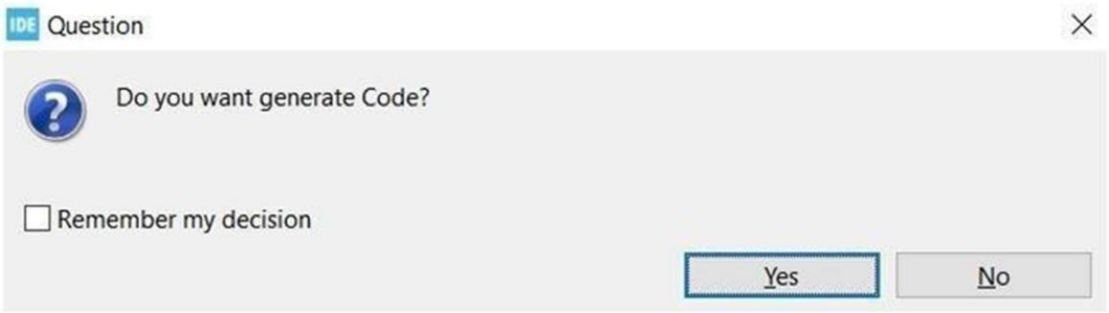
These configurations and parameters allow you to customize the behavior of the GPIO pins to suit the needs of your specific project and ensure proper functionality.

Step

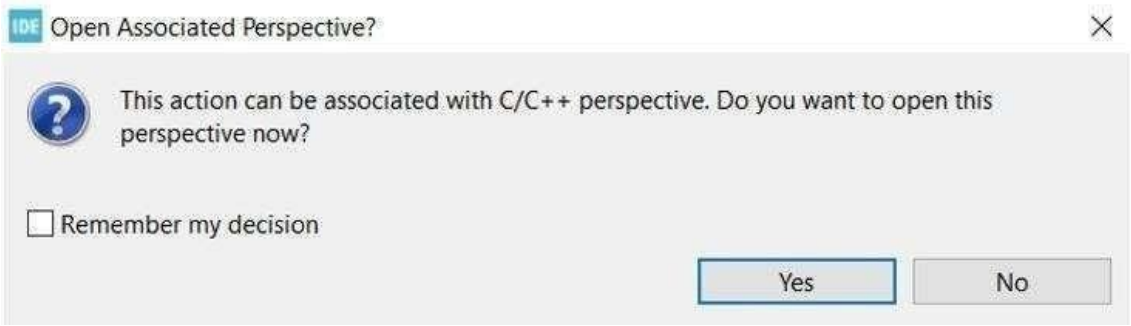


6: Save Project and Generate Code

Now we will save our file. Press Ctrl + S. Thefollowing window will appear. Click ‘Yes.’ This will generate a template code for you.



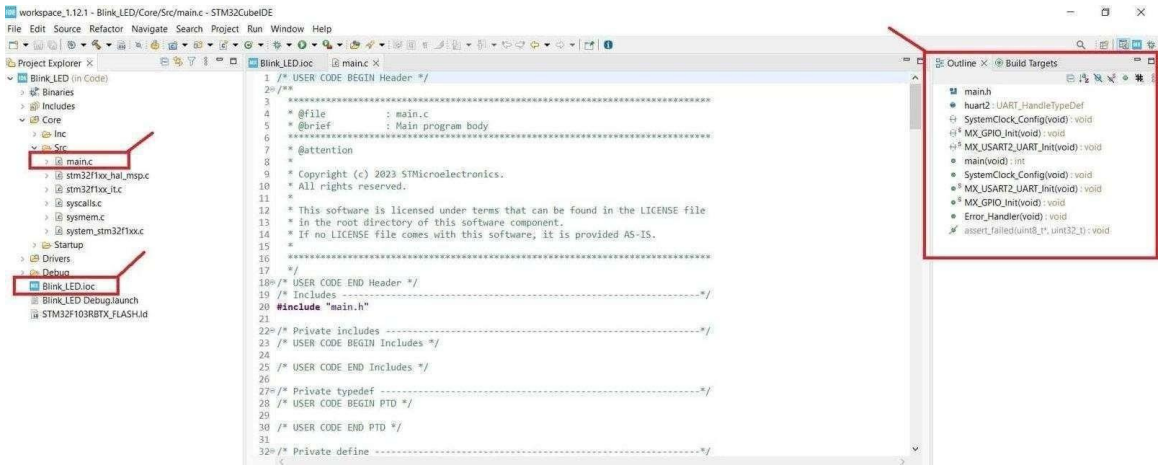
Another window will appear that will ask if you want to open the perspective. Click ‘Yes’.



STM32CubeMX Generated Code:

Step

Now the following page opens. On the right side, you will be able to view the Outline of the code. In the center, you can view the main.c file and on the left, you can view the Project Explorer.



If you want to go to the Device Configuration Tool, then click the BLINK_LED.ioc tab.

STM32CubeMX will add all required HAL libraries and system level HAL libraries to use the PA5 pin of STM32 Nucleo as a digital output pin.

Code: STM32 Nucleo LED Blinking:

Now let's modify the main.c file for our BLINK_LED project. Our aim is to blink the onboard LED indefinitely after a delay.

First, inside the main() function go to while(1) and insert the following lines of code. These will be responsible to blink the onboard LED infinitely.

```
while (1)
{
  /* USER CODE END WHILE */
  HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
  // Toggle the GPIO pin state HAL_Delay(1000); // Delay for 1000 milliseconds
  /* USER CODE BEGIN 3 */
}
```

Save the main.c file after modifying it. Now we are ready to build our project.

STM32 GPIO HAL Driver (Overview):

The HAL driver provides functions to control the GPIO pins of the STM32 family of microcontrollers. The GPIO pins can be configured in one of the following modes:

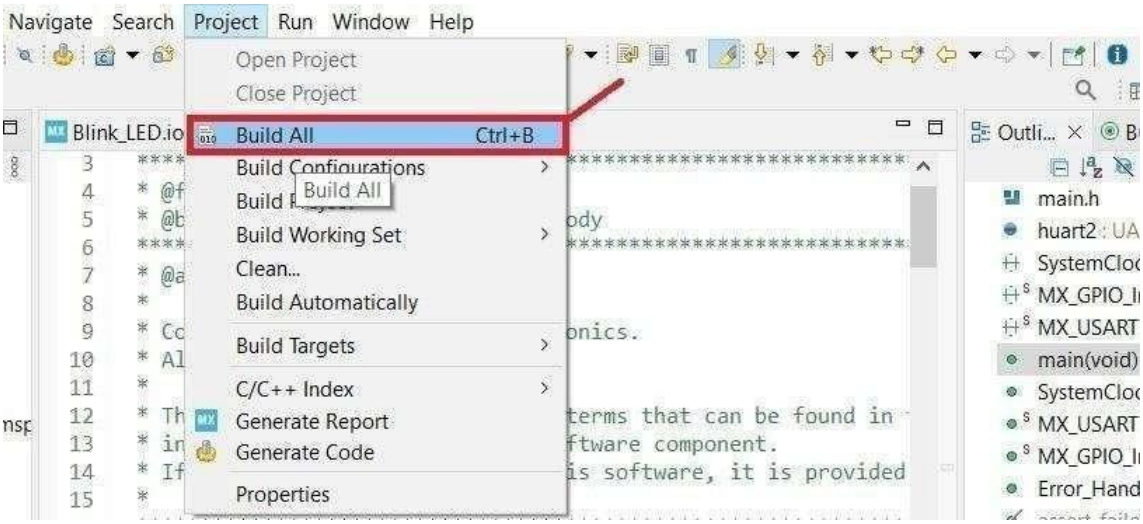
- Digital Input mode
- Analog mode
- Digital Output mode
- Alternate function mode
- External interrupt/event lines

To blink the onboard LED of STM32 Nucleo board with HAL_GPIO_WritePin() function, we can set the corresponding pins for one second and then reset the pins for one second.

```
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
Delay_ms(1000);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
Delay_ms(1000);
```

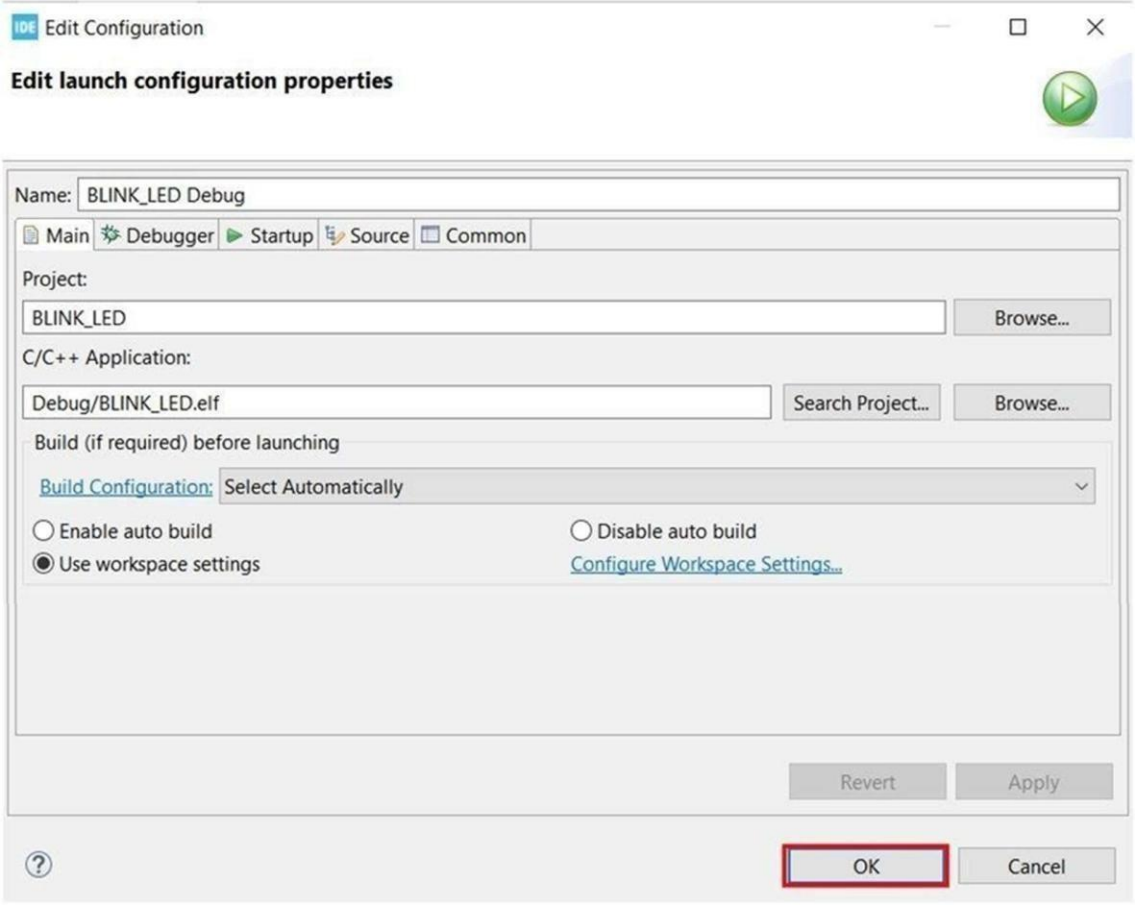
Building the Project

- To build LED Blinking project press Ctrl + B or go to Project > Build All.



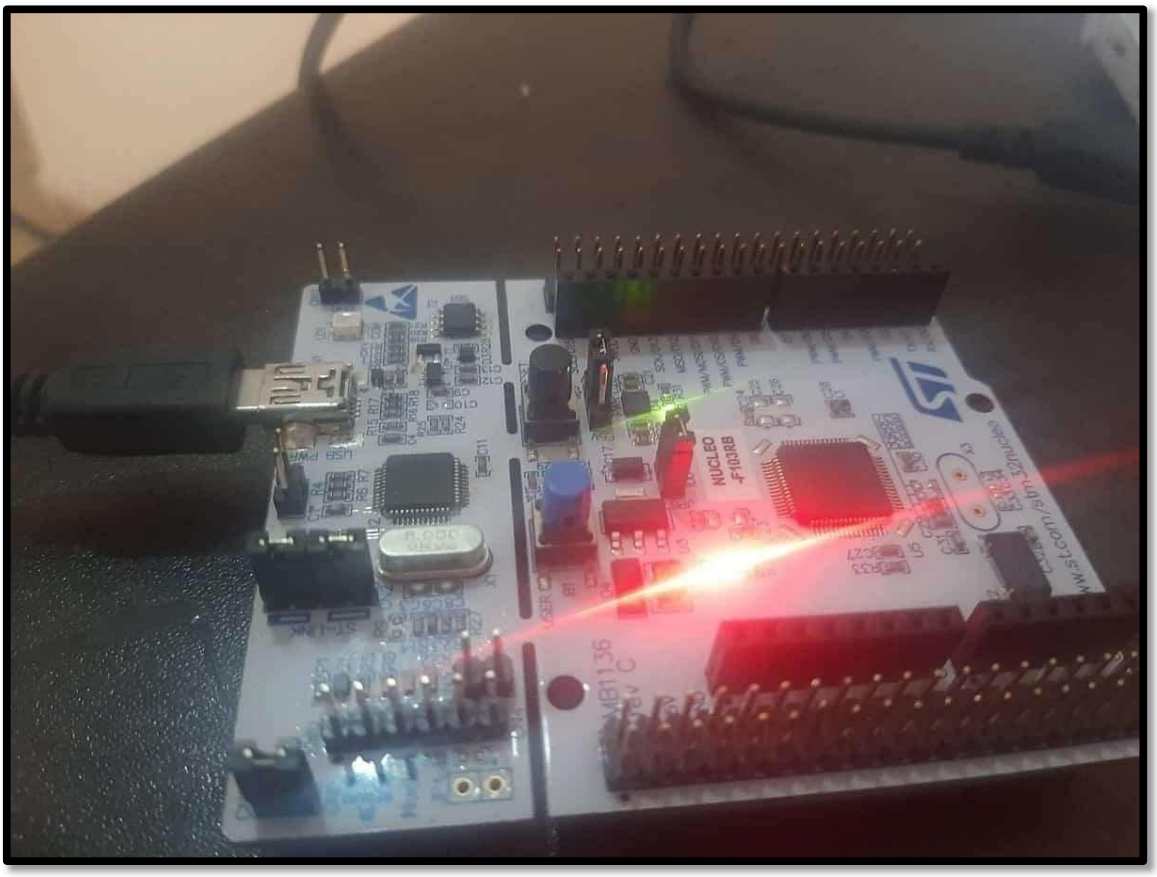
- Your project will start building. After a few moments, your project will be built if there are no errors.

Connect STM32 Nucleo with your computer and next press the RUN button in the IDE. The following 'Edit configuration' window will open up. Click 'OK'.



- After a few moments, the code will be successfully sent to the STM32 board. You can view it in the Console terminal.

Otherwise, press the RESET button on your STM32 board.



After pressing the RESET button on the board. Immediately, the onboard LED will start blinking.

Result:

The onboard LED LD2 connected to pin PA5 of STM32 Nucleo-F103RB board was successfully made to blink with a 1-second delay using HAL functions in STM32CubeIDE.

Conclusion:

The experiment demonstrates the use of STM32 GPIO HAL driver for digital output applications. We configured pin PA5 as a GPIO output, generated code using STM32CubeMX, and programmed the microcontroller to toggle the LED state with a delay. This forms the basic building block for controlling hardware peripherals using STM32.