INDIAN INSTITUTE OF TECHNOLOGY PATNA

EC3101: MICROCONTROLLER AND

EMBEDDED SYSTEMS LAB

EXPERIMENT 1:
Blinking LED using STM32 Nucleo Board

Submitted by:

Name: Shaurya Aggarwal

Roll No: 2301EE56

Aim:

The goal is to develop a program in STM32CubeIDE that makes the onboard LED (LD2), connected to GPIO pin PA5 of the STM32 Nucleo-F103RB board, blink continuously.

Components / Tools Required:

- STM32 Nucleo-F103RB Development Board
- USB cable (for programming and powering the board)
- STM32CubeIDE software

Theory: LED Blinking on STM32 Nucleo Board

In this exercise, we use GPIO pins of the STM32 Nucleo board as digital output pins. For demonstration, we will control the onboard LED of the STM32 Nucleo-F103RB board.

- The Nucleo board has an onboard LED named **LD2**, which is connected to pin **PA5**.

Operation:

- The LED turns **ON** for one second when GPIO PA5 is set to a high state.
- Then the LED turns **OFF** for one second when GPIO PA5 is set to a low state.
- These steps repeat continuously, producing a blinking effect.

Procedure:

Writing Your First STM32 Project in STM32CubeIDE

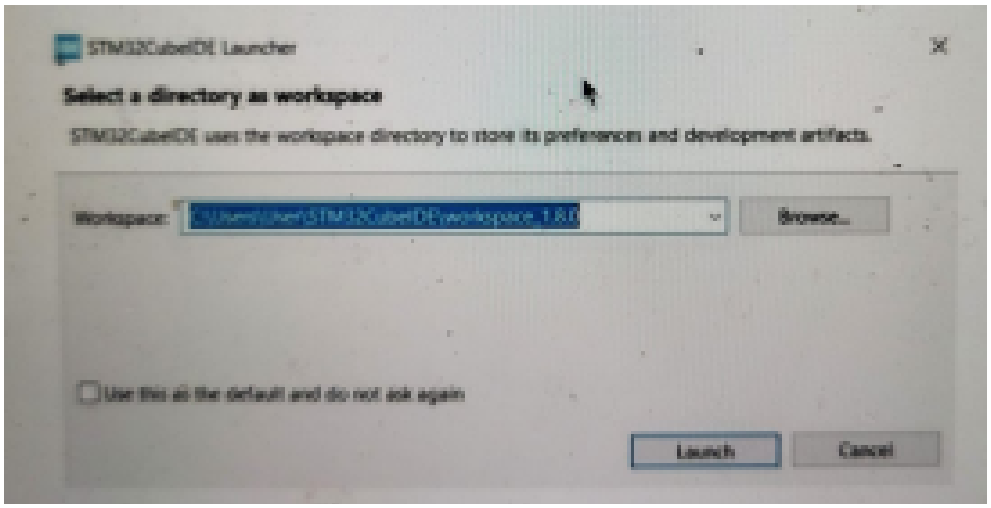
After installing STM32CubeIDE on your computer (Windows/Linux/Mac), follow these steps to create the LED blinking project.

You will learn how to:

1. Create and configure a project using STM32CubeMX.
2. Generate initialization code.
3. Write and use HAL functions to blink the onboard LED of the STM32 Nucleo-F103RB board.

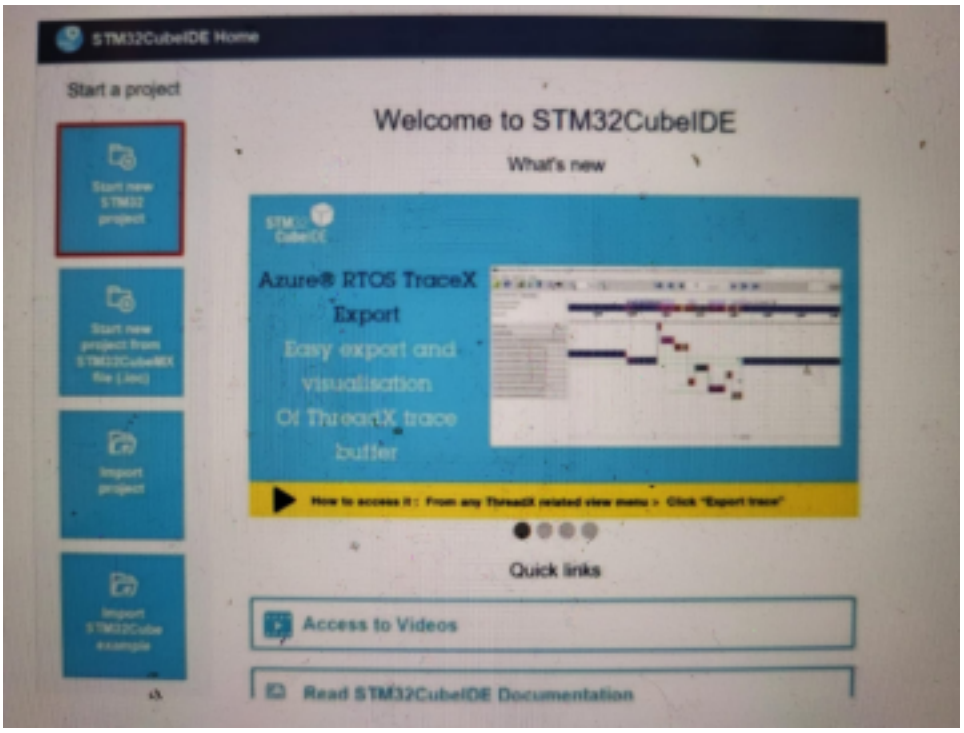
Step 1: Launch STM32CubeIDE

- Open the STM32CubeIDE software.
- Choose a workspace directory (can also be set as default).
- Click **Launch** to start the IDE.



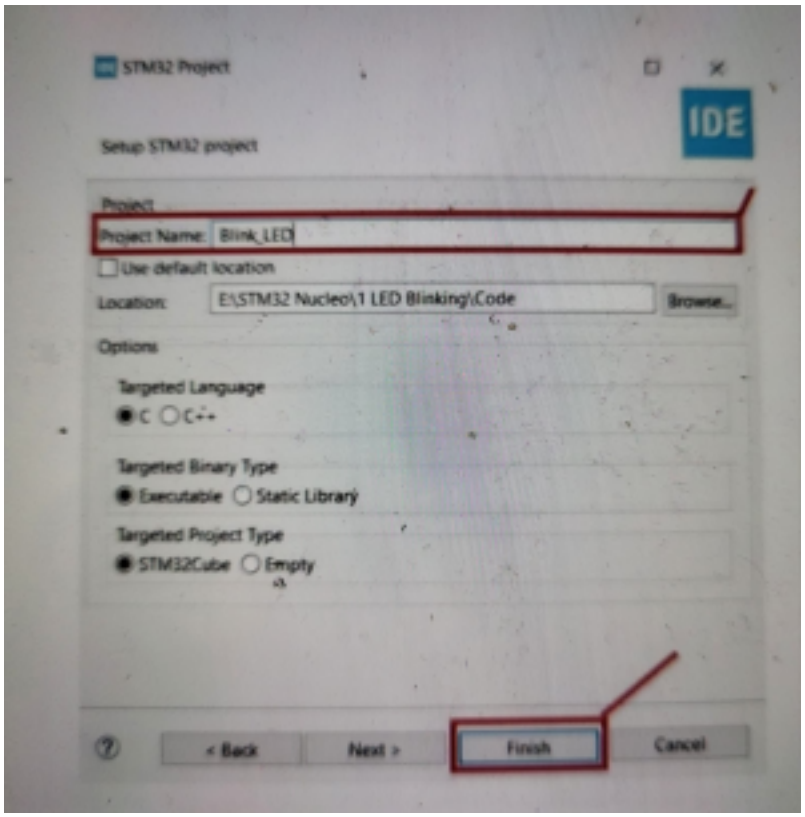
Step 2: Create New STM32 Project

• Click on **Start new STM32 project**.



• In the **Board Selector**, type **STM32 Nucleo-F103RB** and select it. Click **Next**.

• Enter a project name (e.g., **BLINK_LED**) and press **Finish**.



• You may be asked to open the STM32CubeMX perspective – click **Yes**.

Step 3: Device Configuration Tool

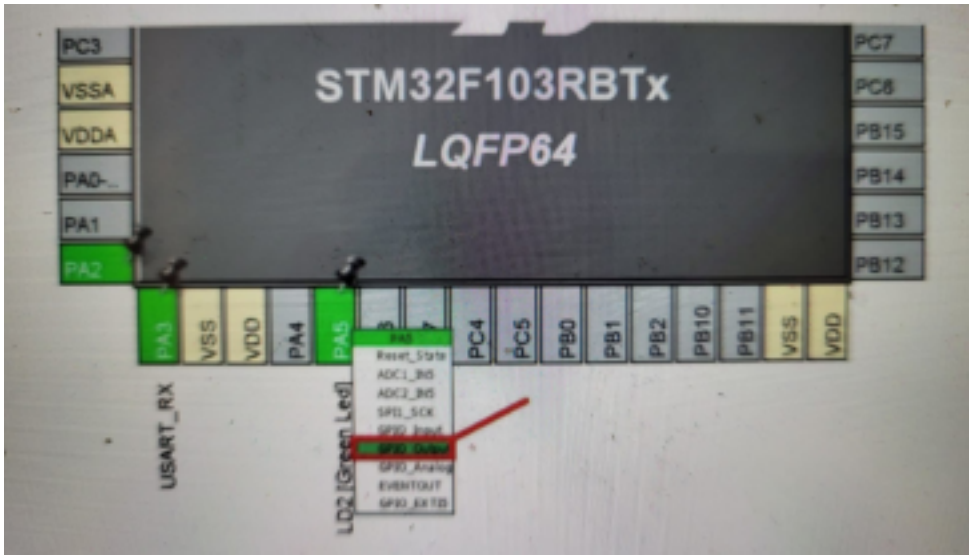
• The Device Configuration window will open.
• Select pins and assign functions.
• Note: A single GPIO pin can serve multiple functions (ADC, SPI, UART, etc.), but only one function can be active at a time.

Step 4: Select GPIO Pin

• For this experiment, choose **PA5** as the output pin since it is linked to the onboard LED (LD2). • Configure PA5 as **GPIO Output**.
• The pin will be labeled **LED** in the configuration.

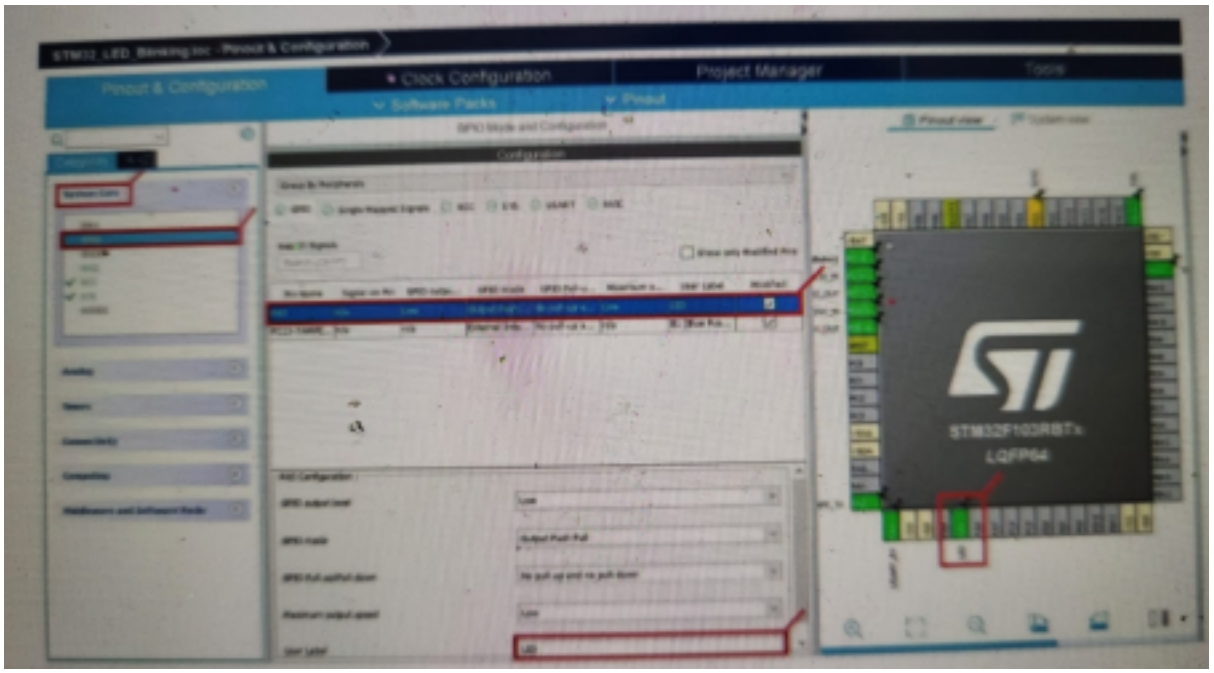
Additional GPIO settings include:

• **Output Level**: Default is Low (can be set to High).
• **Mode**: Output Push-Pull (default).
• **Pull-up/Pull-down**: None by default (can be changed).
• **Maximum Speed**: Initially Low for power saving (can be increased).
• **User Label**: Custom name assigned for easy reference.



Step 5: Pin Configuration

- Navigate to **System Core > GPIO > PA5** for configuration.
- Ensure PA5 is labeled as **LED** and set as **GPIO Output**.



Step 6: Save and Generate Code

- Press **Ctrl + S** to save the configuration.
- A window will appear – click **Yes** to generate code.
- Another window may ask to open the code perspective – click **Yes**.

STM32CubeIDE now generates a template project with the necessary HAL libraries.

Code for LED Blinking

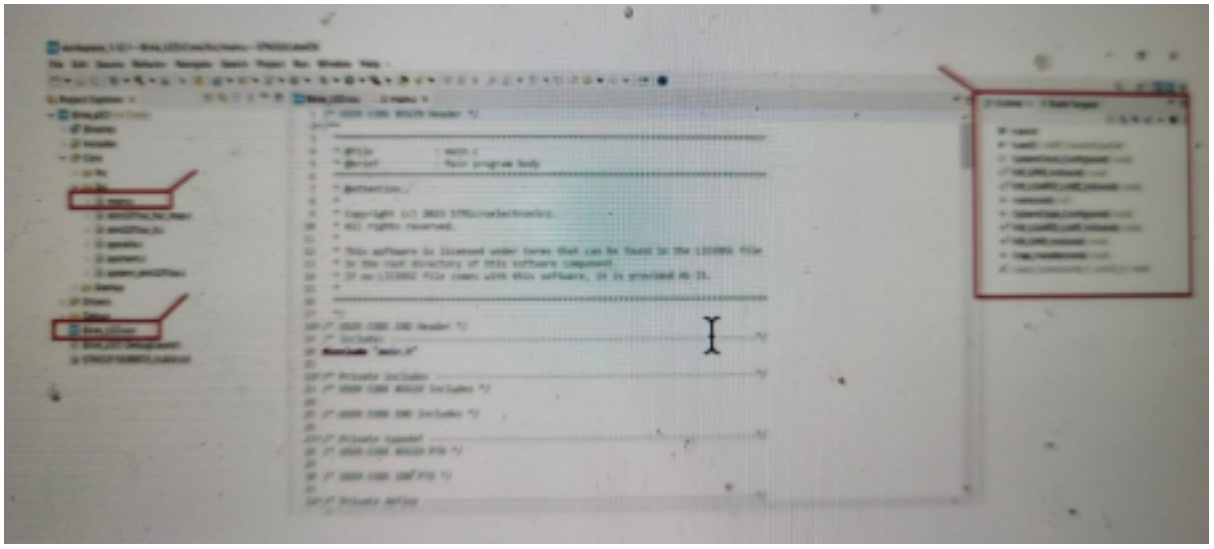
Open **main.c** in your project. Inside the while(1) loop, add the following:

```
while (1)
{
 HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5); // Toggle LED on PA5
 HAL_Delay(1000); // Wait for 1 second
}
```

This code will blink the LED indefinitely with a 1-second delay.

Alternative using WritePin

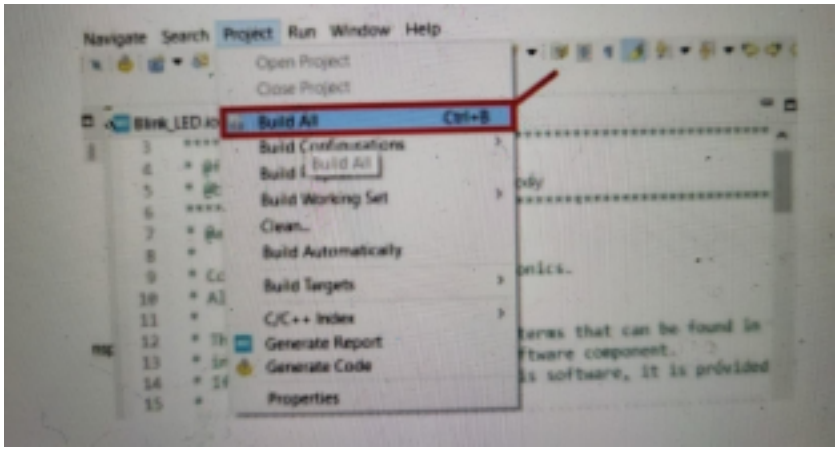
```
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET); // LED ON
HAL_Delay(1000); // Delay 1 sec
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET); // LED OFF
HAL_Delay(1000); // Delay 1 sec
```

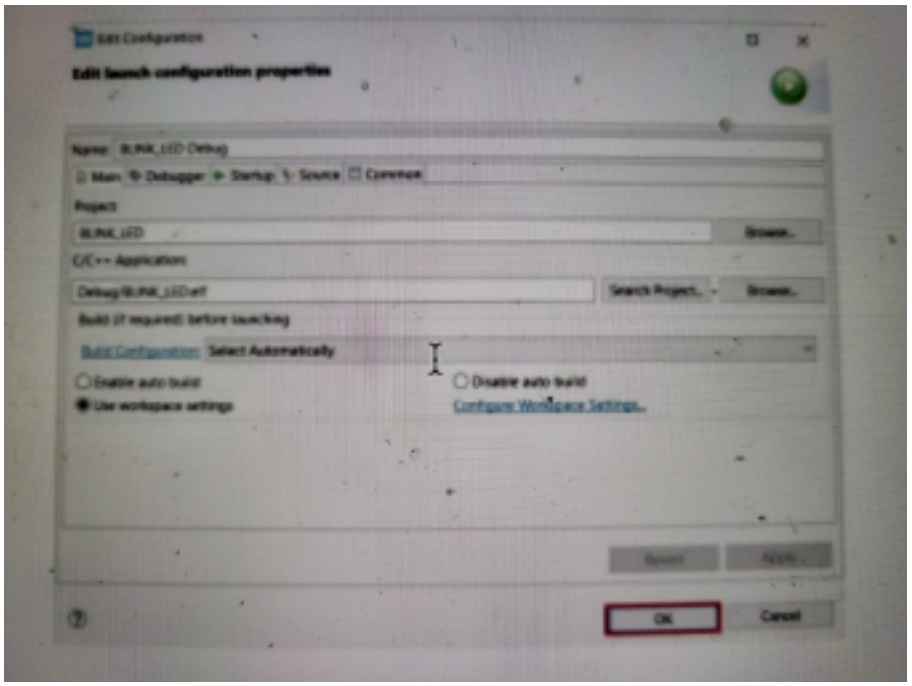


Building and Running the Project

1. Build the project with **Ctrl + B** or via **Project > Build All**.
2. Connect the STM32 Nucleo board via USB.
3. Click **Run** in STM32CubeIDE.
4. If prompted, select the default debug configuration and click **OK**. 5. The code will be uploaded to the board, and the LED will start blinking.

If it does not run automatically, press the **RESET** button on the Nucleo board.





Results:

The onboard LED (LD2) connected to **PA5** successfully blinked with a 1-second delay, confirming correct GPIO configuration and code execution using STM32CubeIDE.

Conclusion:

This experiment demonstrated how to configure and control a digital output pin on an STM32 microcontroller using STM32CubeIDE. By setting **PA5** as a GPIO output and writing HAL-based code, we were able to toggle the onboard LED with delays. This forms a fundamental step for working with hardware peripherals on STM32 boards.