A

Project Report On

# "Interfacing LEDs, Switches and Servo Motor using Nucleo-F401RE STM32"

Submitted in the partial fulfilment for the

ENGINEERING IN ELECTRONICS AND TELECOMMUNICATION

2022-2023


Submitted By

Pradnya S. Babrekar (BE20F04F001)

Rasika Bonde(BE20F04F006)

Vaishnavi Deshmukh(BE20F04F013)

-Under guidance of

**Prof. S. S. Agrawal**

Electronics and Telecommunication Engineering

Department of Electronic and Telecommunication Engineering Government College of Engineering Aurangabad (2022-2023)

# *Acknowledgement*

*It is indeed a great pleasure of ours to present this project report after having undergone a unforgettable moment of excitement anxiety, experience and understanding these pleasure could not be have without the support extended to us by our guide Prof. S. S. Agrawal Ma'am of Electronics and Telecommunication Engineering, Government College of Engineering Aurangabad. Who not only encourages us through the venture but also to great point in going through the manuscript carefully and correction which have greatly improved the quality of the text.*

*Last but not least we wish to thanks to our parents for financing our studies college as well as constantly encouraging us to learn Engineering. Their personal sacrifice in providing opportunity to learn Engineering is great fully acknowledge and also thanks to all our friends who directly or indirectly help us in our work.*

*Thank You...!!*

***Submitted By***

*Miss. Pradnya Babrekar*
*Miss. Rasika Bonde*
*Miss.Vaishnavi Deshmukh*

# Eskill Project Report

## Interfacing of Stm32 NUCLEO-F401RE STM32 board with LED, Switch and Servo Motor

# Getting started with STM32

## STM32 Microcontrollers:

STM32 is a family of microcontrollers developed by STMicroelectronics. These microcontrollers are based on the ARM Cortex-M processor cores and are widely used in various embedded systems and Internet of Things (IoT) applications. The STM32 family offers a wide range of microcontrollers with different features and performance levels to suit different application requirements.

## Nucleo STM32F401RE Development Board:

The Nucleo STM32F401RE is a specific development board within the STM32 Nucleo family. It is designed to provide an affordable and flexible platform for prototyping, evaluation, and development of applications based on the STM32F401RET6 microcontroller. The Nucleo STM32F401RE board includes the following key components and features:

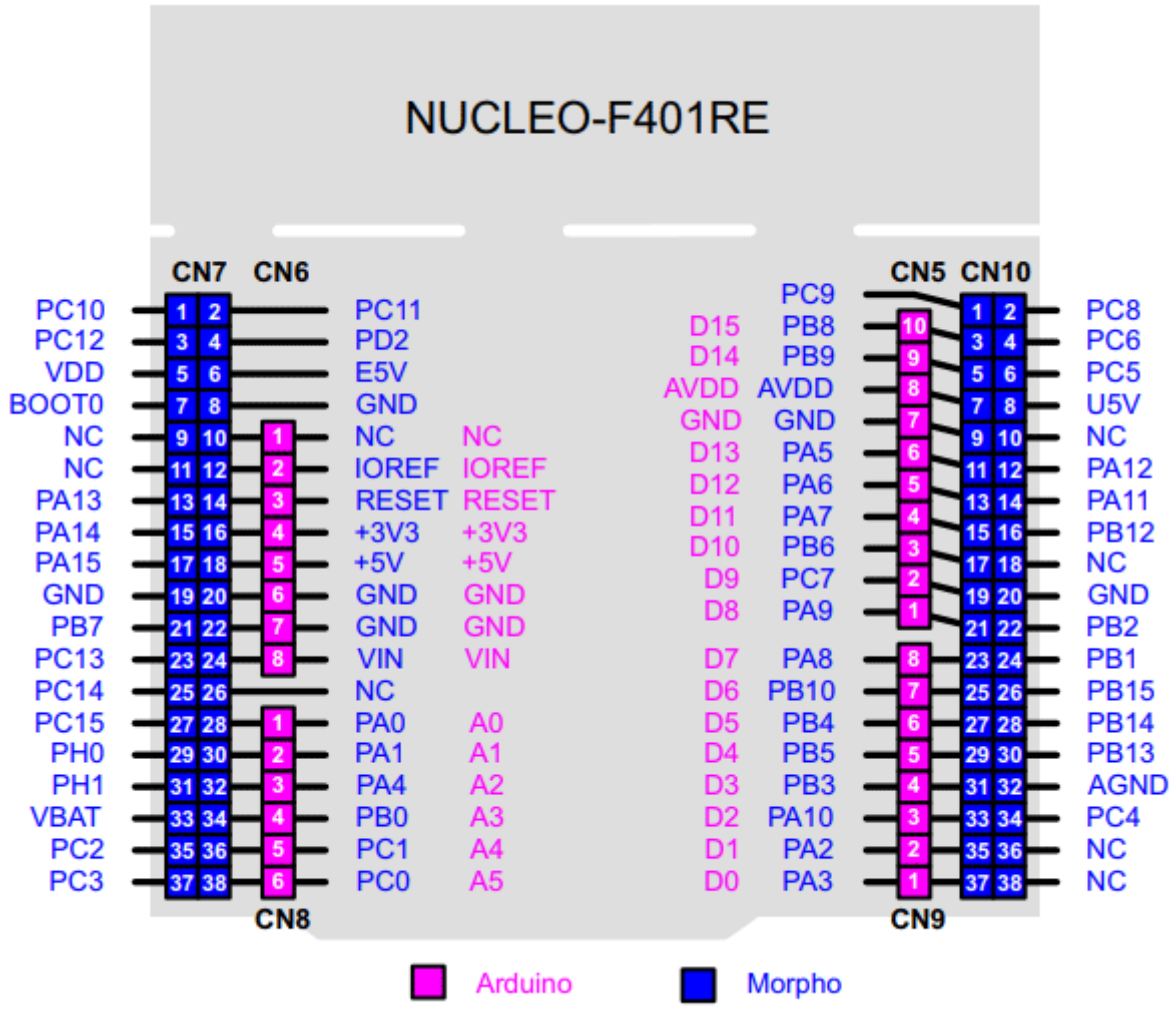
- STM32F401RET6 Microcontroller: This microcontroller is the heart of the board and provides processing power and various peripherals for interfacing with external devices.
- Flash Memory and SRAM: The microcontroller has onboard Flash memory for program storage and SRAM for data storage.
- Expansion Connectors: The board features Arduino Uno Rev3 and ST morpho expansion connectors, allowing compatibility with a wide range of expansion boards and shields.
- Integrated Debugger/Programmer: The board integrates an ST-Link/V2-1 debugger/programmer, enabling programming and debugging of the microcontroller using the onboard USB connection.
- Power Supply and Connectivity: The board can be powered via USB and includes a USB port for communication. It also has a user button and LED for general-purpose input/output

## Development Environment:

The Nucleo STM32F401RE development board is supported by the STM32Cube software development platform. This platform provides a comprehensive set of tools, libraries, middleware, and examples to facilitate software development for STM32 microcontrollers. It supports various integrated development environments (IDEs) such as STM32CubeIDE, Keil MDK, and IAR Embedded Workbench.

The combination of the STM32 microcontroller and the Nucleo STM32F401RE development board offers a powerful and accessible platform for building and testing embedded applications. The board's expansion connectors, integrated debugger/programmer, and compatibility with popular development environments make it suitable for a wide range of projects, from simple prototypes to more complex applications.

## Nucleo STM32F401RE Pinout:



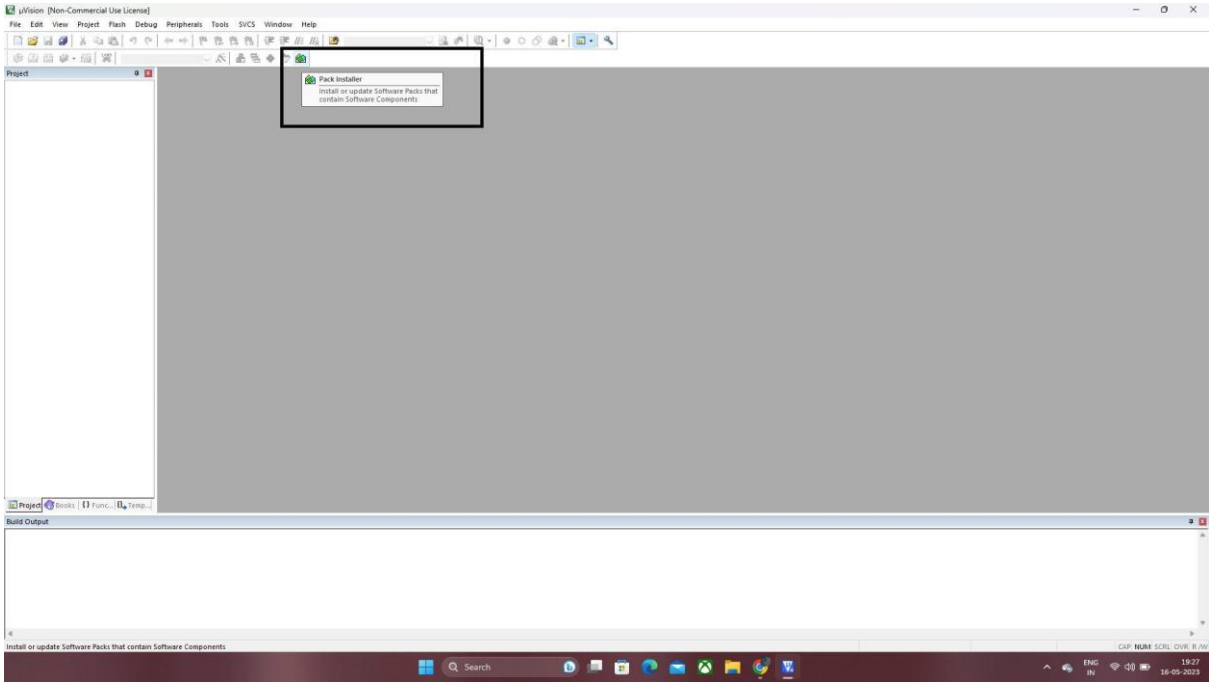

## Software Requirements:

1. Keil uVision 5 MDK

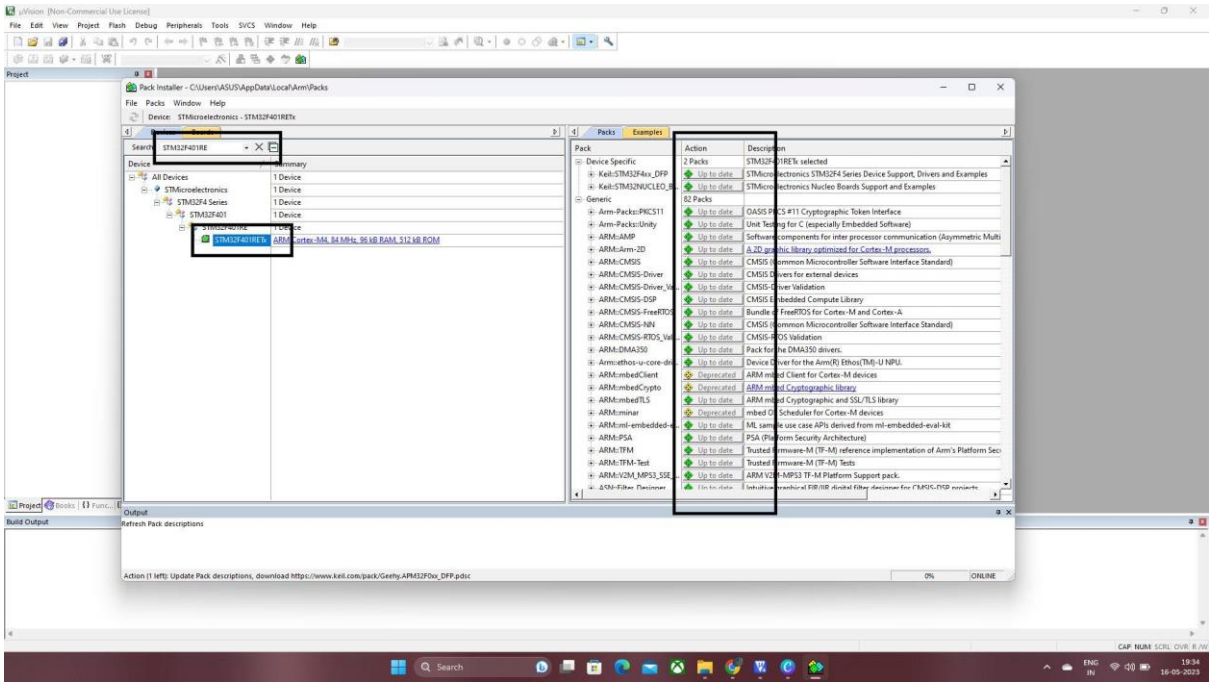
2. STM32CubeMx
3. STM-Studio
4. STM32CubeIDE

## Initial Setup:

1) First you need to set up the Keil.
2) Click on software pack



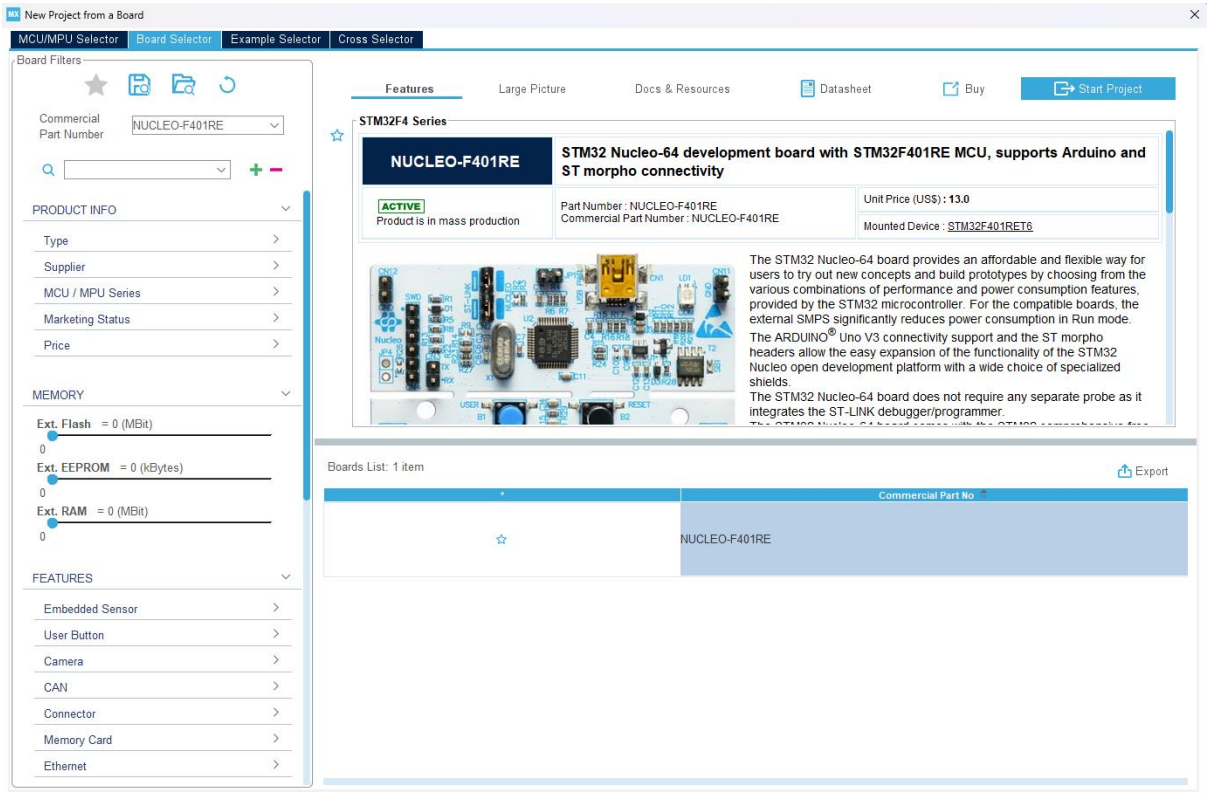

3) Next select the board, type in your board name and install all the packs.



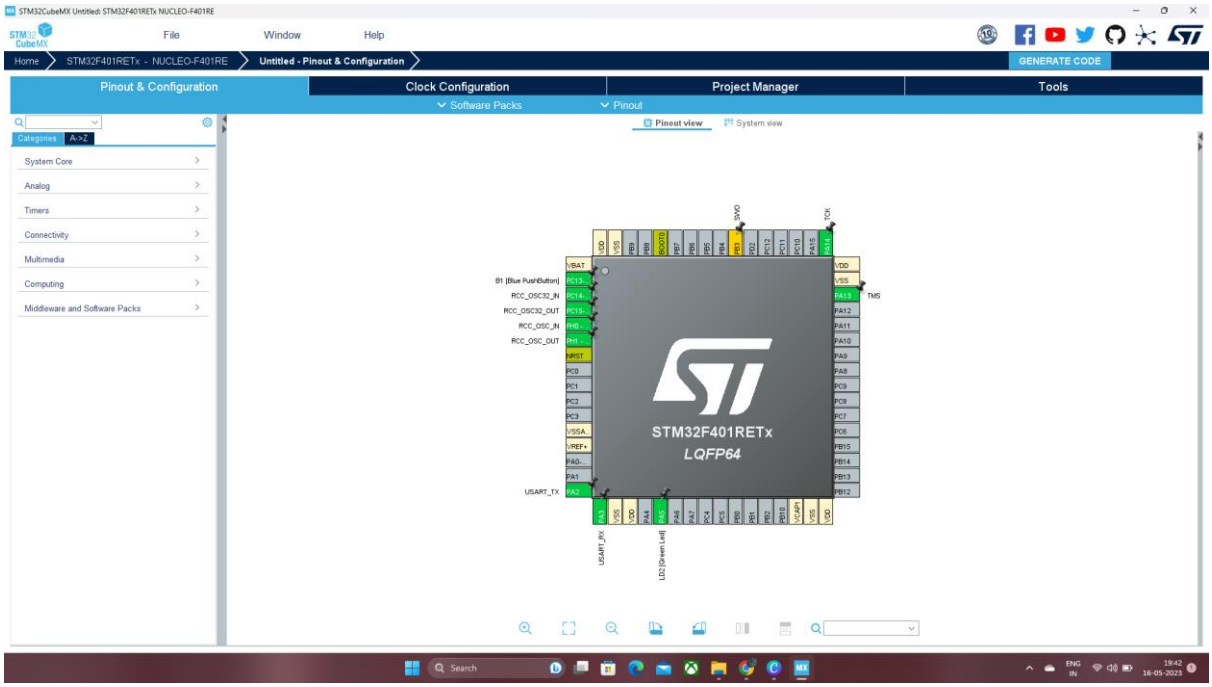
4) Install the STM32CubeMx.
5) Install the STM-Studio.
6) Install the STM32CubeIDE.

# Create your project:

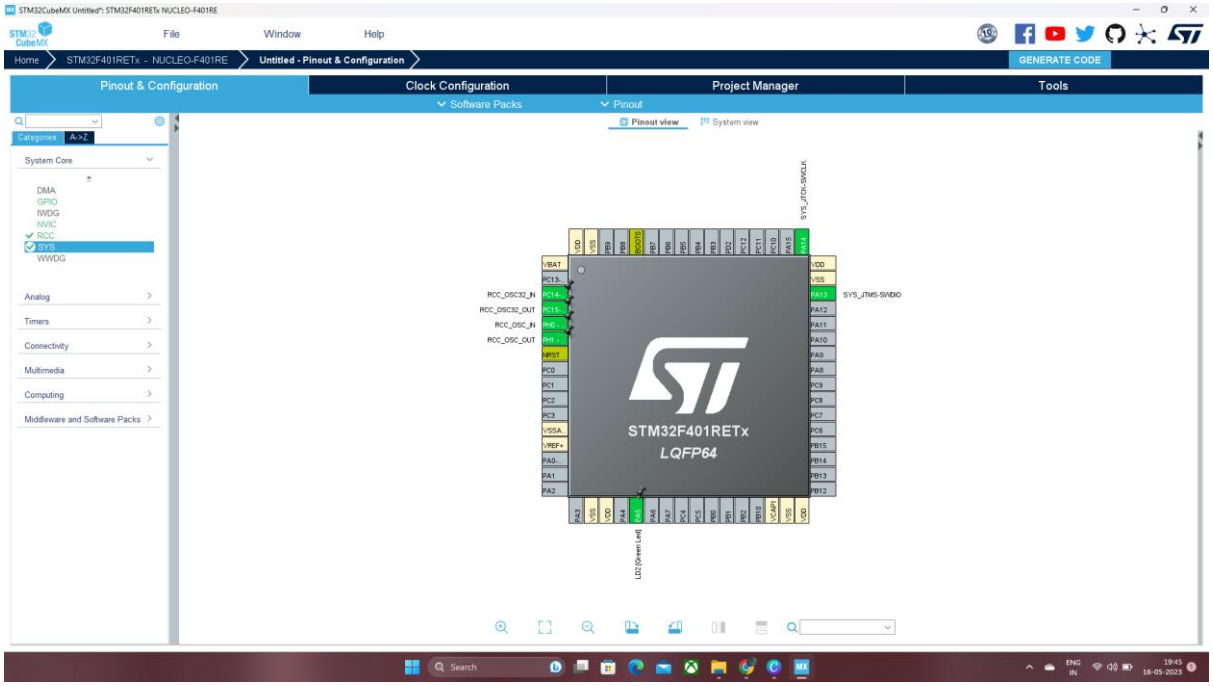
1) Open STM32CubeMx
2) Select new project, access to board selector
3) Select your board by double clicking it



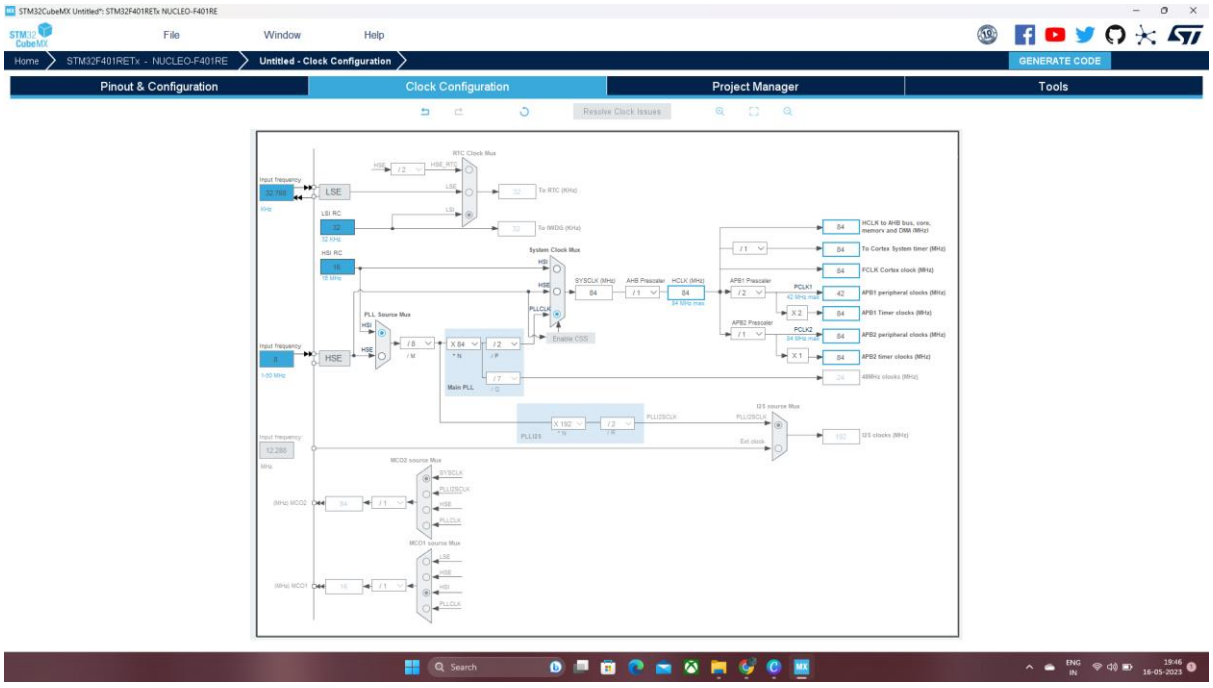

4) Clear the pinout (ctrl + p)



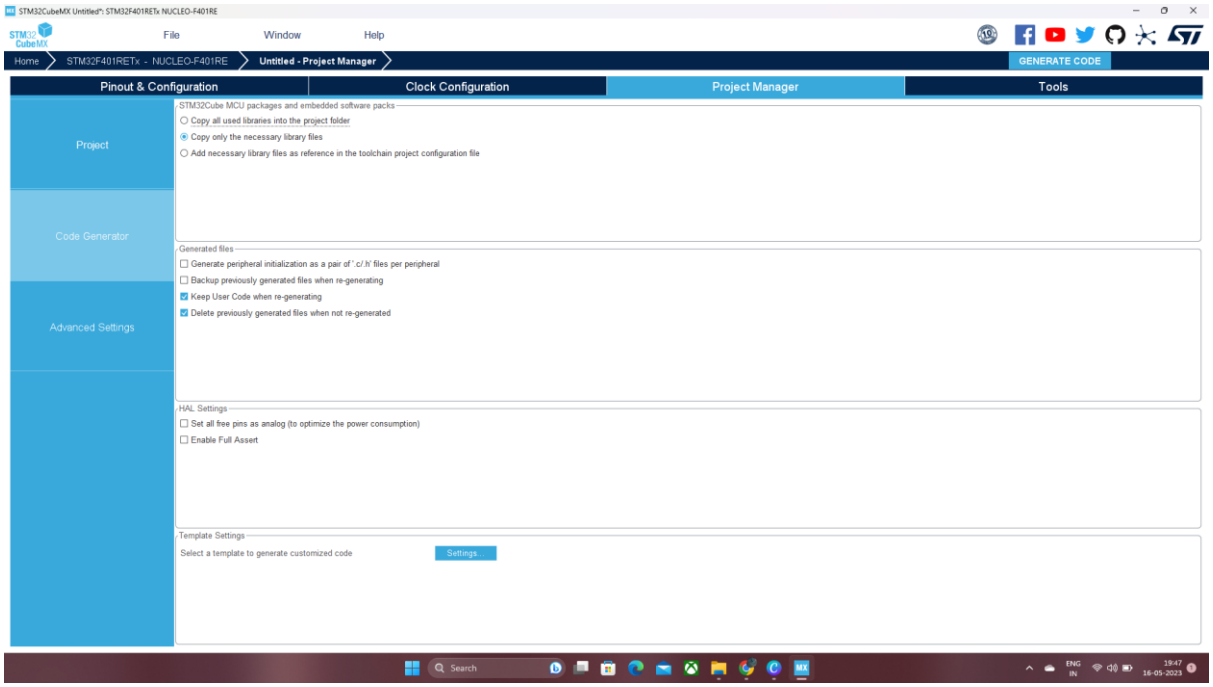
5) Select the led pin and mark it as output also select the RCC as crystal oscillator

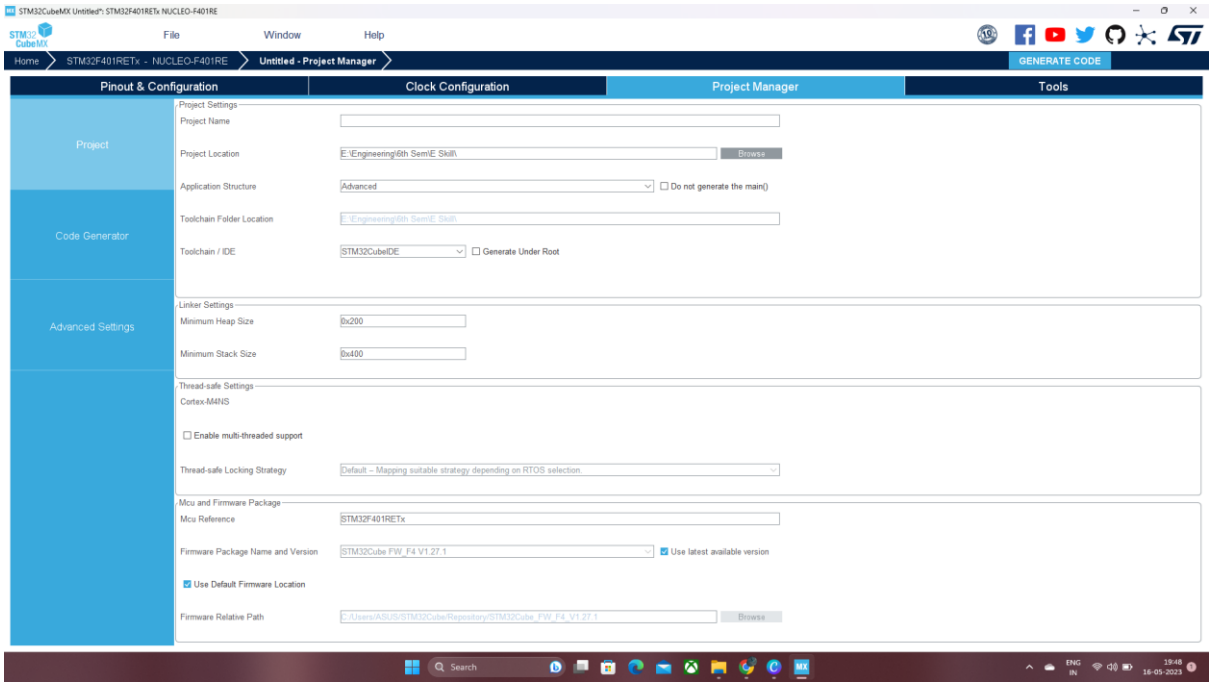



6) Select the clock configuration to 84 MHz as shown below

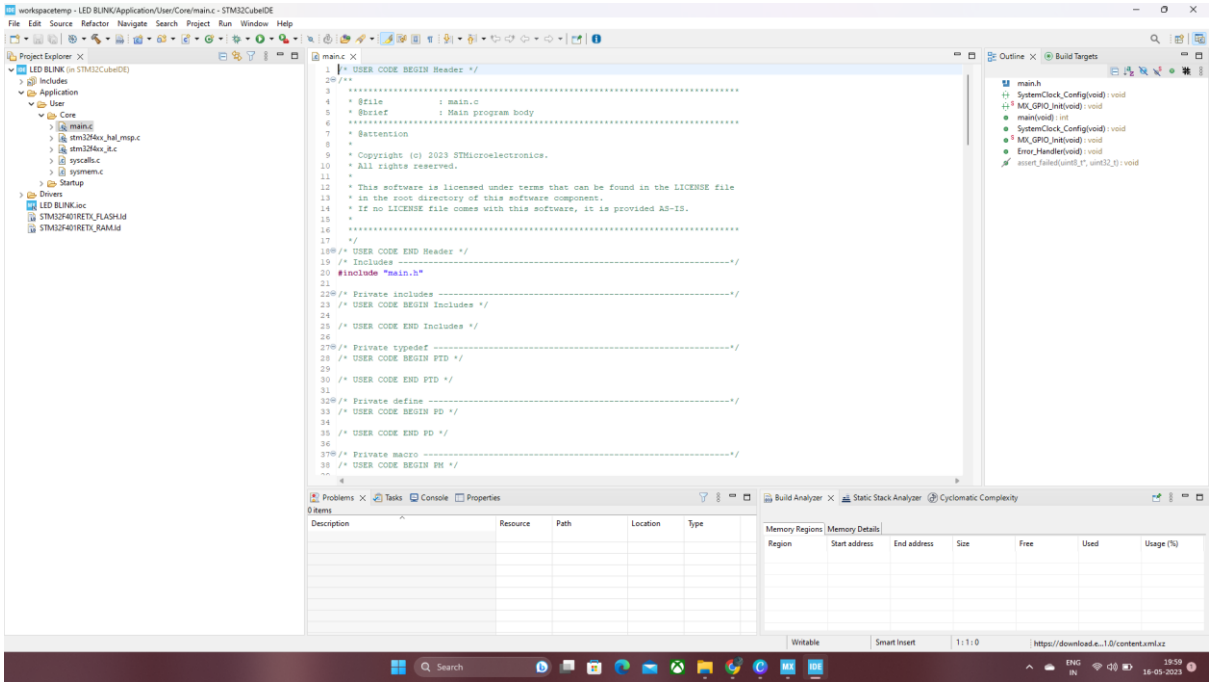


7) Click generate project and follow along

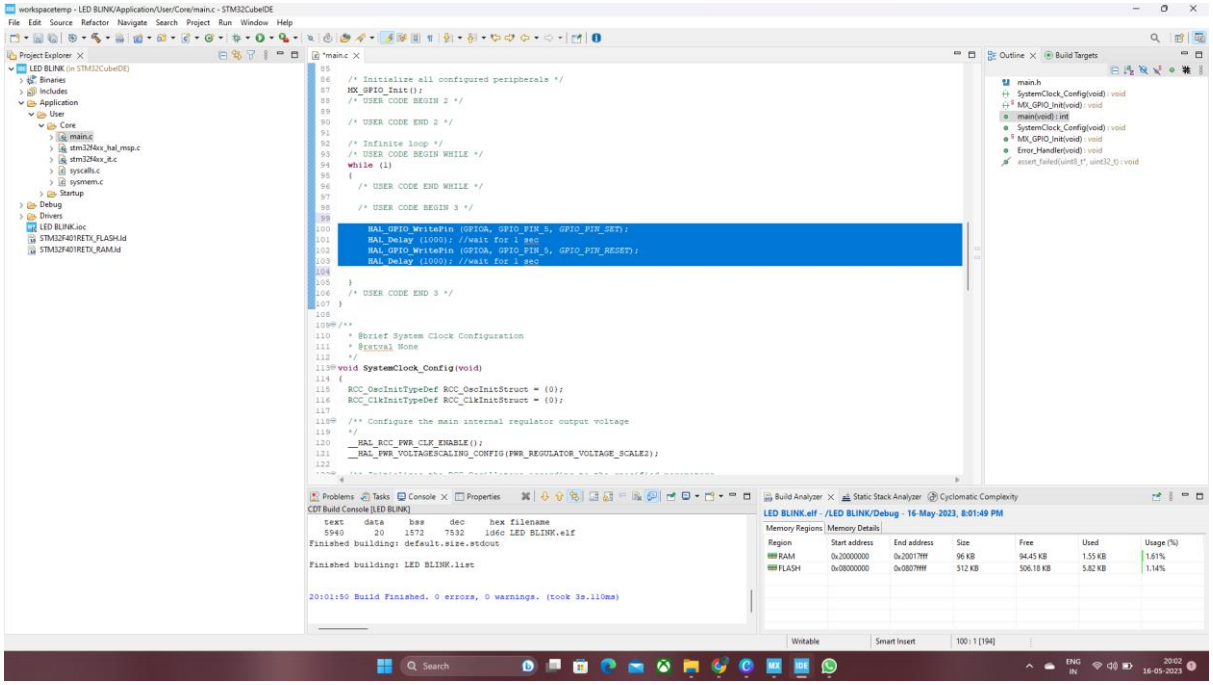


8) Now click ok and open the project. It will open the STM32CubeIDE.
9) Open main.c file in STM32CubeIDE

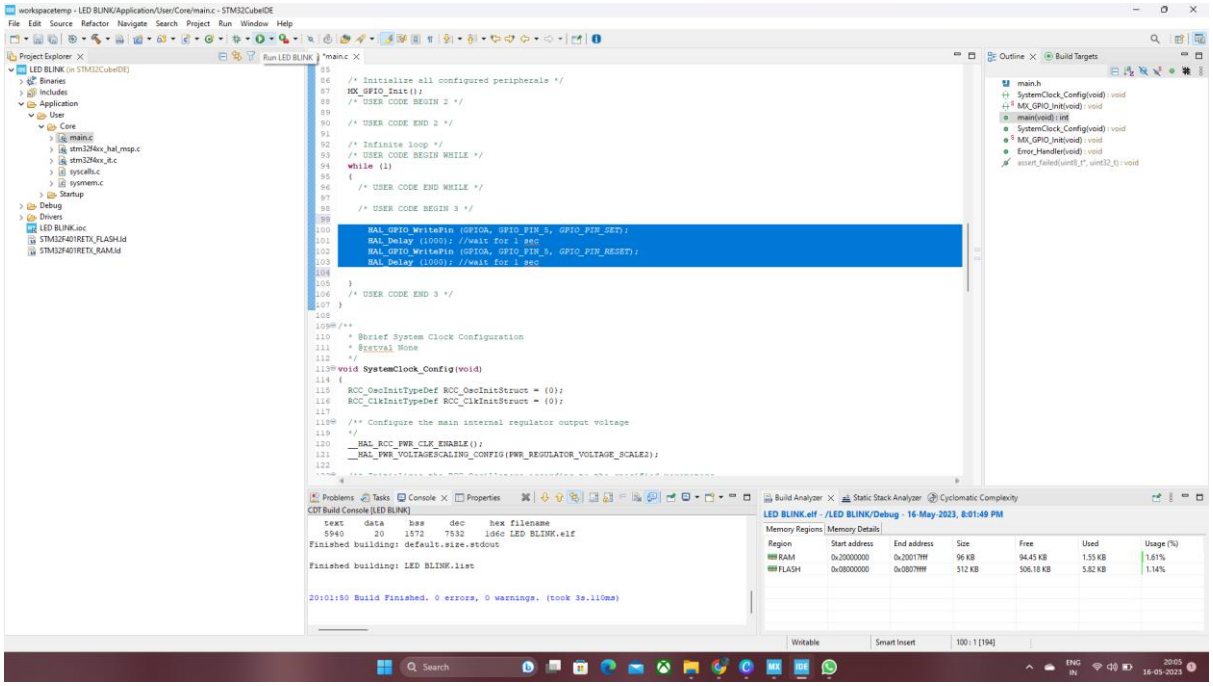


10) Write the program. You can check the available functions in the function tab HAL_Delay is in milliseconds.



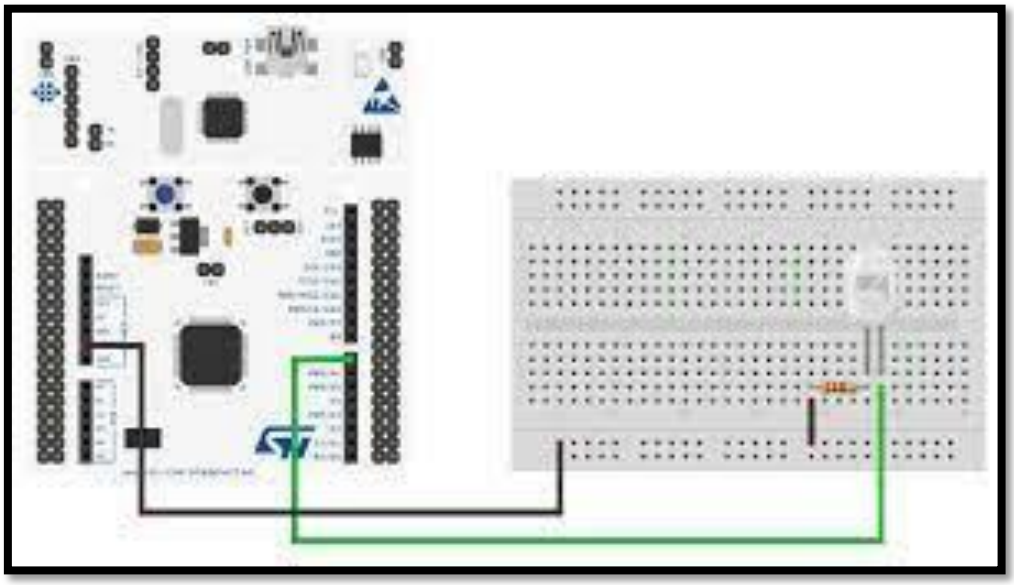
Make sure you write the code after "user code begin". This is because if you want to regenerate the same code with STM32CubeMx with some editing, it will not affect your program.

11) That's it. Now click build.
12) Now Click the RUN button and it will load the program to your board. Make sure you have installed all necessary drivers before doing this or else it will give an error. You can install the drivers from the device manager.

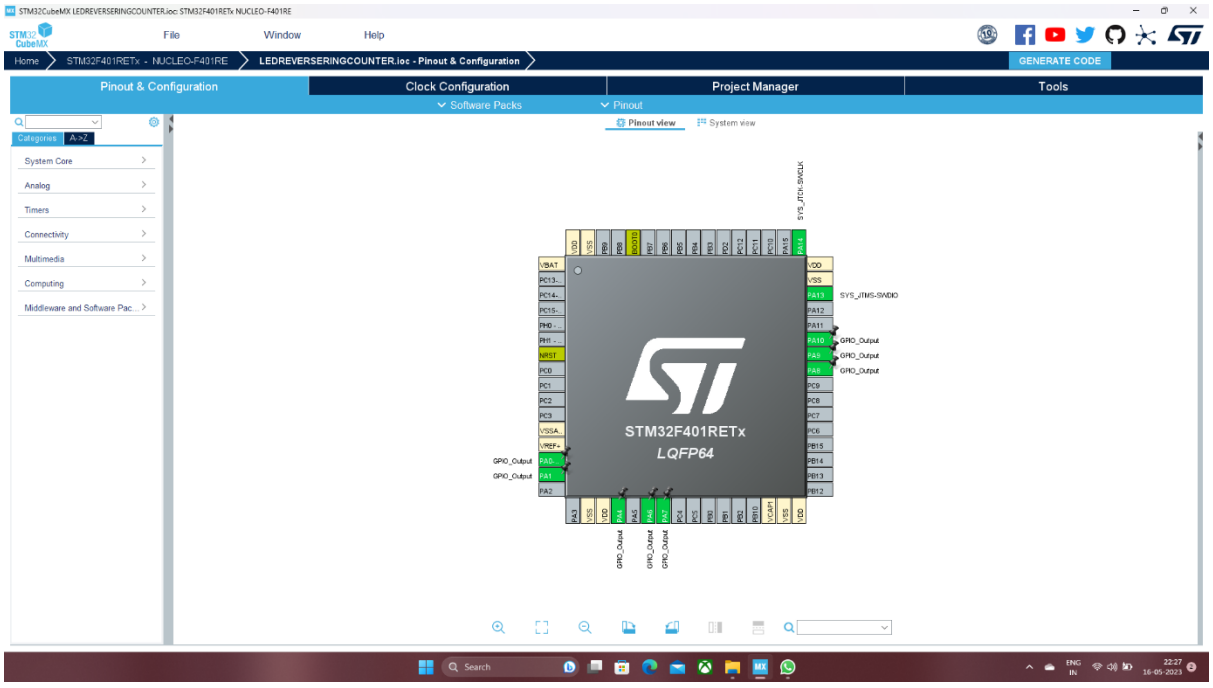



13) Click reset on board, and you can see the led blinking every 1 second.

# Interfacing with LED:



## LED Pinout & Configuration.



## STM32CubeMx Settings

Set PA0, PA1, PA4, PA6, PA7, PA8, PA9 and PA10 to GPIO_Output

## Additional code on top of STM32CubeIDE generated code

**1ST Pattern-**

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

        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec

        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_RESET);
        HAL_Delay (1000); //wait for 1 sec

   /* USER CODE END WHILE */
```

Now click BUILD and then click RUN and it will load the program to your board.

**2nd Pattern-**

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

```
  {

        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_RESET);
        HAL_Delay (1000); //wait for 1 sec


   /* USER CODE END WHILE */
```

**3rd Pattern-**

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

        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);
```

```
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_RESET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec


    /* USER CODE END WHILE */
```

**4th Pattern-**

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

        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
        HAL_Delay (1000); //wait for 1 sec
```

```
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_RESET);
        HAL_Delay (1000); //wait for 1 sec

    /* USER CODE END WHILE */
```

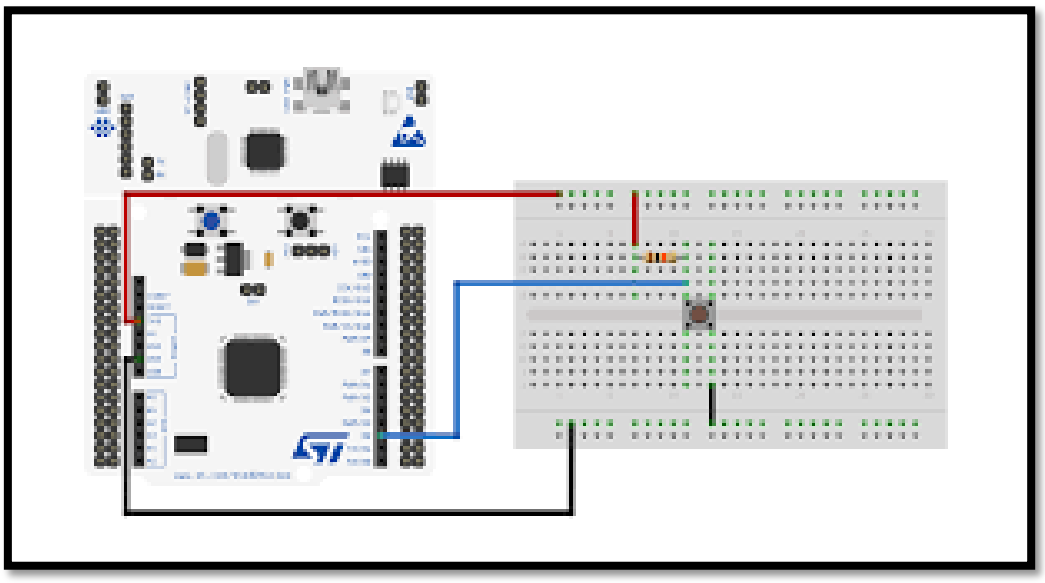
**5th Pattern-**

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

        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
        HAL_Delay (1000); //wait for 1 sec

        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
```

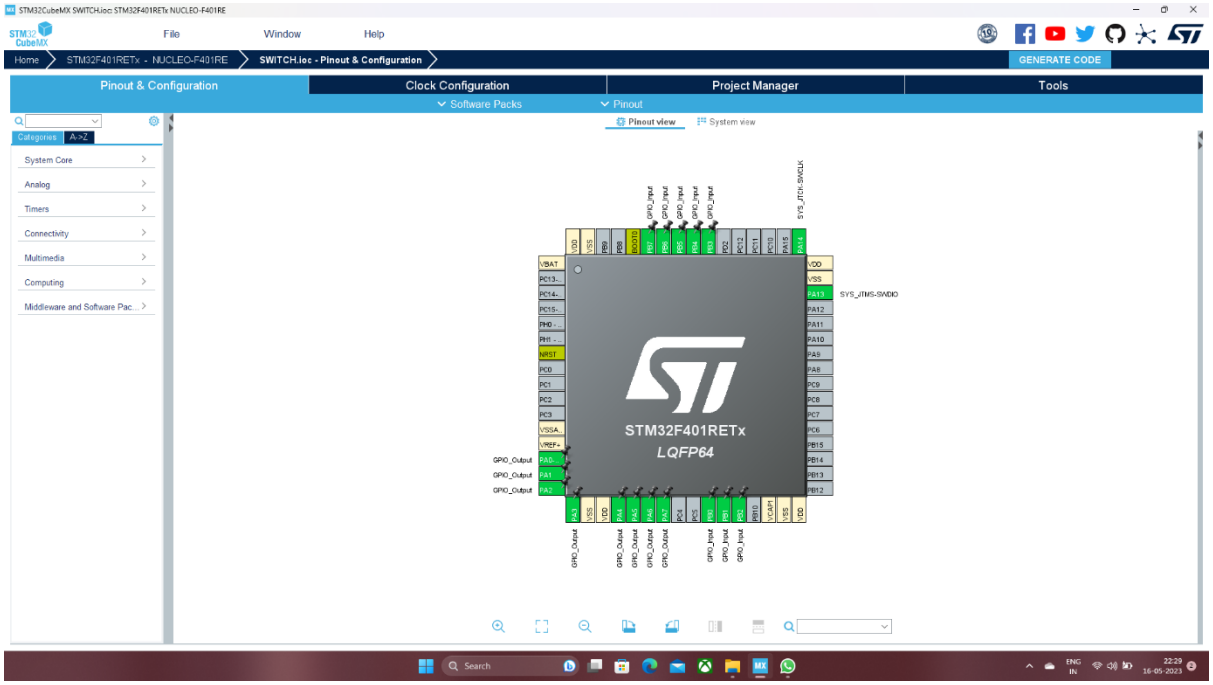
```
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
HAL_Delay (1000); //wait for 1 sec

HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
HAL_Delay (1000); //wait for 1 sec

HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
HAL_Delay (1000); //wait for 1 sec

HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
HAL_Delay (1000); //wait for 1 sec
```

```
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
HAL_Delay (1000); //wait for 1 sec

HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
HAL_Delay (1000); //wait for 1 sec

HAL_GPIO_WritePin (GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
HAL_GPIO_WritePin (GPIOA, GPIO_PIN_10, GPIO_PIN_RESET);
HAL_Delay (1000); //wait for 1 sec


/* USER CODE END WHILE *
```

# Interfacing with Switch:



## Switch Pinout & Configuration.



## STM32CubeMx Settings

Set PA0, PA1, PA2, PA3, PA4, PA5, PA6 and PA7 to GPIO_Output
Set PB0, PB1, PB2, PB3, PB4, PB5, PB6 and PB7 to GPIO_Input

## Additional code on top of STM32CubeIDE generated code

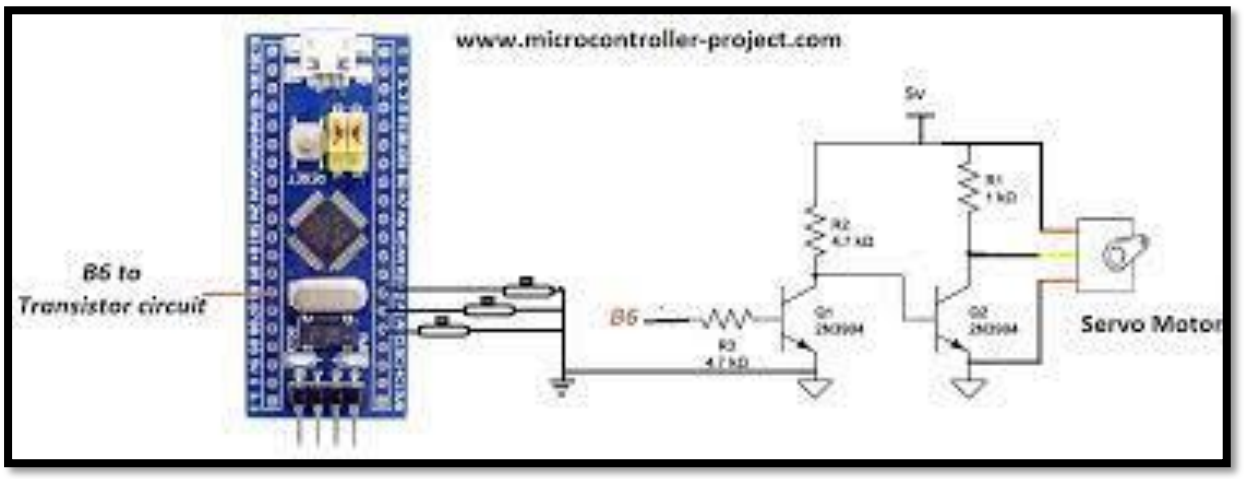
```
/* USER CODE BEGIN WHILE */
  while (1)
  {

		if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_0) ==
GPIO_PIN_RESET)
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, 1);
		}
		else
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, 0);
		}

		if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_1) ==
GPIO_PIN_RESET)
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 1);
		}
		else
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 0);
		}

		if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_2) ==
GPIO_PIN_RESET)
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 1);
		}
		else
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 0);
		}
```

```
		if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_3) ==
GPIO_PIN_RESET)
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 1);
		}
		else
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 0);
		}

		if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_4) ==
GPIO_PIN_RESET)
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 1);
		}
		else
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 0);
		}

		if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_5) ==
GPIO_PIN_RESET)
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 1);
		}
		else
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 0);
		}

		if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_6) ==
GPIO_PIN_RESET)
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, 1);
		}
		else
		{
			HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, 0);
		}
```

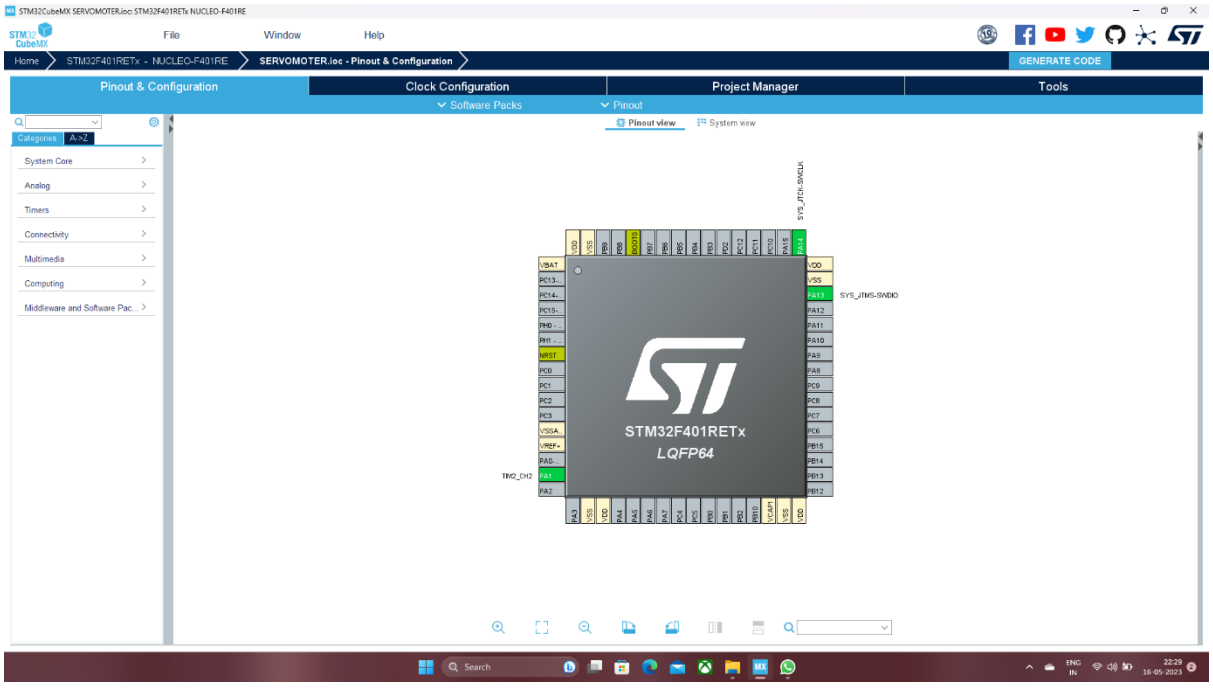
```
            if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_7) ==
GPIO_PIN_RESET)
            {
                  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 1);
            }
            else
            {
                  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 0);
            }


    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
```

Now click BUILD and then click RUN and it will load the program to your board.

# Interfacing with Servo Motor:




## Servo Motor Pinout & Configuration.



## STM32CubeMx Settings

Click Timer → Click TIM2 →
Clock Source set to Internal Clock
Channel2 set to PWM Generation CH2
Configuration → Parameter Settings →
Prescaler set to 16-1
Counter Period 10000

## Additional code on top of STM32CubeIDE generated code

```
/* USER CODE BEGIN 2 */
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);
  /* USER CODE END 2 */

  /* USER CODE BEGIN WHILE */
  while (1)
  {
    __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, 850);
    HAL_Delay(2000);
    __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, 950);
    HAL_Delay(2000);
    __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, 1050);
    HAL_Delay(2000);
    __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, 1250);
    HAL_Delay(2000);
    __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, 750);
    HAL_Delay(1000);
    __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, 650);
    HAL_Delay(2000);
    __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, 550);
    HAL_Delay(2000);
    __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, 450);
    HAL_Delay(2000);
    __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, 250);
    HAL_Delay(2000);
    __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, 750);
    HAL_Delay(1000);
    /* USER CODE END WHILE */
```

```
  /* USER CODE BEGIN 3 */
 }
 /* USER CODE END 3 */
```

Now click BUILD and then click RUN and it will load the program to your board.

## Conclusion-

The interfacing of LEDs, switches, and a servo motor using the STM32 Nucleo board allows for control and manipulation of these components through the microcontroller. This system enables various applications, such as creating simple digital output displays with LEDs, reading input states from switches, and controlling the position of a servo motor. In conclusion, the interfacing of LEDs, switches, and a servo motor using the STM32 Nucleo board opens up opportunities for creating interactive and dynamic projects, allowing you to control and manipulate these components with the power of the microcontroller.