STM32 CubeIDE, CubeMx for beginners with Examples

Welcome to my STM32CubeIDE specifically designed for Beginners. In this, we will take a step-by-step approach to ensure that you gain a solid understanding of how to use **CudeIDE, CubeMx** for programming the **STM32** Microcontroller.

Hardware requirements:

1. STM32 blue pill board
2. ST-LINK/V2
3. Jumper wires
4. SSD1306 Oled Display Module
5. HC-SR04 Ultrasonic Sensor
6. Push Button
7. Led, resistors

Software requirements:

1. Stm32 cubeide
2. Stm32 cubemx

Installation procedure: https://www.youtube.com/watch?v=qe8ToTBv7jY

I will be using the **STM32F103C6T6A** microcontroller board which is also known as Blue Pill, and for uploading the programming I will use the **ST-Link/V2**.

STM32F103C6T6A MICROCONTROLLER OVERVIEW

The STM32 Blue Pill development board is based on the STM32F103C6T6A microcontroller from the STM32F1 series by STMicroelectronics. It uses the ARM Cortex-M3 core and is widely used for embedded system learning and development due to its low cost and rich peripherals.

MICROCONTROLLER CORE

The STM32F103C6T6A is based on the ARM Cortex-M3 32-bit RISC processor. It provides a good balance between performance, power efficiency, and ease of programming, making it suitable for real-time embedded applications.

CLOCK SPEED

The microcontroller operates at a maximum clock frequency of 72 MHz, which allows fast execution of instructions and efficient peripheral handling.

MEMORY

The STM32F103C6T6A contains 32 KB of Flash memory used to store the program code and 10 KB of SRAM used for data storage during program execution.

OPERATING VOLTAGE

The operating voltage range of the STM32F103C6T6A is from 2.0 V to 3.6 V, with 3.3 V as the typical operating voltage. This makes it suitable for low-power applications.

GPIO PINS

The microcontroller provides approximately 32 GPIO pins. These pins are divided into Port A, Port B, and Port C and can be configured as input or output for interfacing with external devices such as LEDs, switches, sensors, and modules.

ANALOG PINS

The STM32F103C6T6A has 10 analog input pins connected to a built-in 12-bit Analog-to-Digital Converter (ADC). These pins are used to measure analog signals from sensors.

COMMUNICATION INTERFACES

The microcontroller supports multiple communication protocols. It has three USART interfaces for serial communication, two I2C interfaces for sensor and module communication, SPI for high-speed data transfer, and CAN for industrial communication.

PWM AND TIMERS

Multiple timers are available in the STM32F103C6T6A. These timers are used to generate PWM signals, perform time measurements, and control motors and other real-time operations.

POWER PINS ON BLUE PILL BOARD

The Blue Pill board provides two 3.3 V pins from the onboard voltage regulator. These pins can be used to power external sensors. The 5 V pin is used as an input power pin, and the onboard regulator converts it to 3.3 V. The board also has three common ground pins.

PROGRAMMING AND DEBUGGIN

The STM32F103C6T6A supports programming and debugging through SWD (Serial Wire Debug) pins. An ST-Link V2 programmer is commonly used for flashing and debugging the microcontroller.

VOLTAGE TOLERANCE

Most GPIO pins on the board are 5 V tolerant, allowing safe interfacing with 5 V devices. However, the PB5 pin is not 5 V tolerant and should only be used with 3.3 V signals.

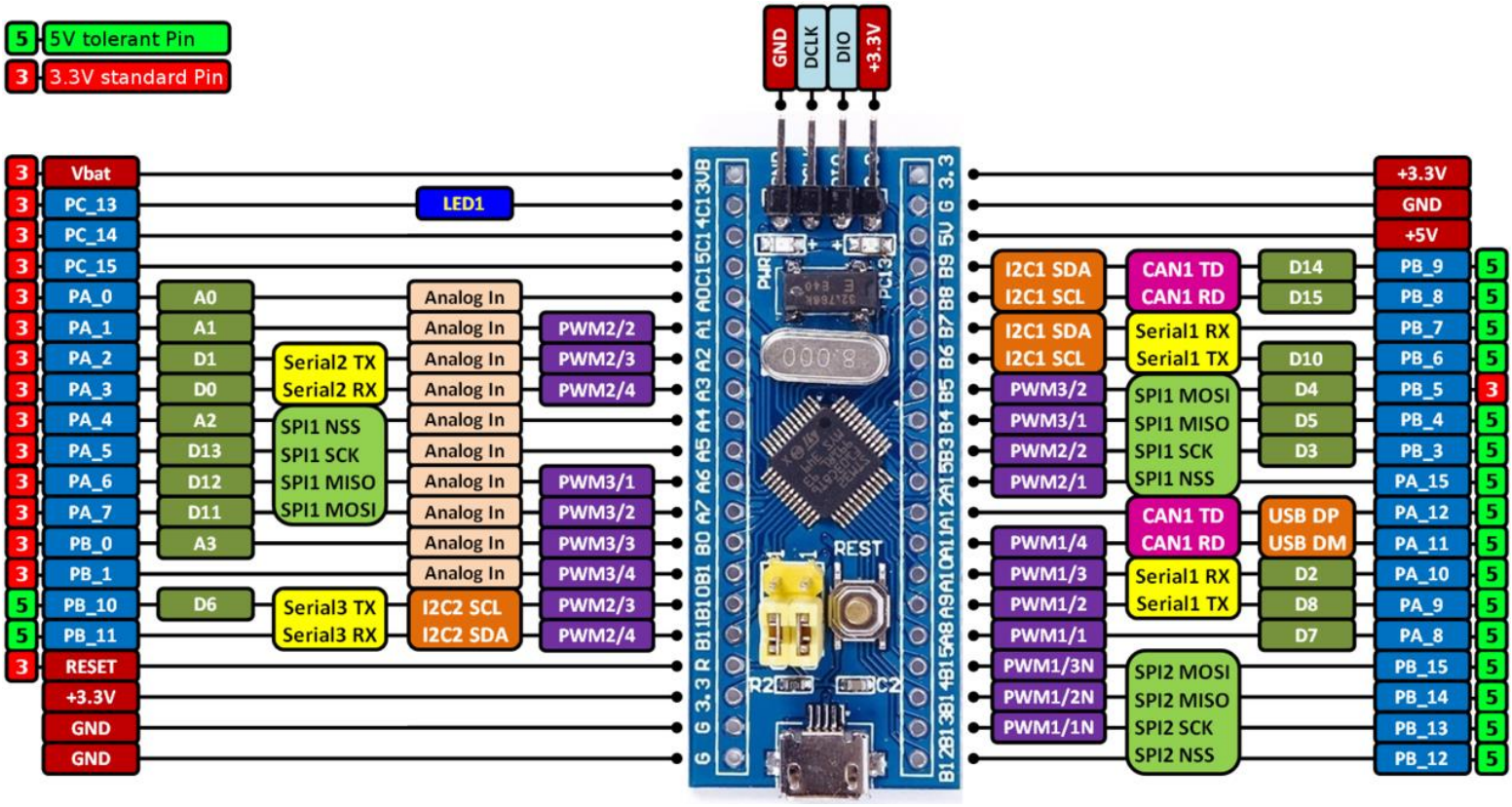
DEVELOPMENT ENVIRONMENT

STM32CubeIDE is the official integrated development environment used for programming the STM32F103C6T6A. Programming is typically done using C or C++.

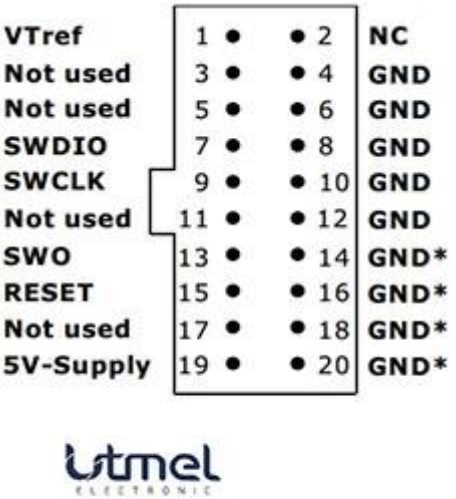
APPLICATIONS

The STM32F103C6T6A is widely used in industrial automation, robotics, IoT devices, motor control systems, and embedded system learning projects.

STM32 Blue Pill Board Pinout:



ST-Link/V2 Pinout:



Connection:

ST-Link/V2 Pin	Signal Name	STM32 Blue Pill Pin
Pin 7	SWDIO	SWDIO
Pin 9	SWCLK	SWCLK
Pin 20	GND	GND
Pin 1	VREF (3.3V)	3.3V

Project-1: Blink on-board LED

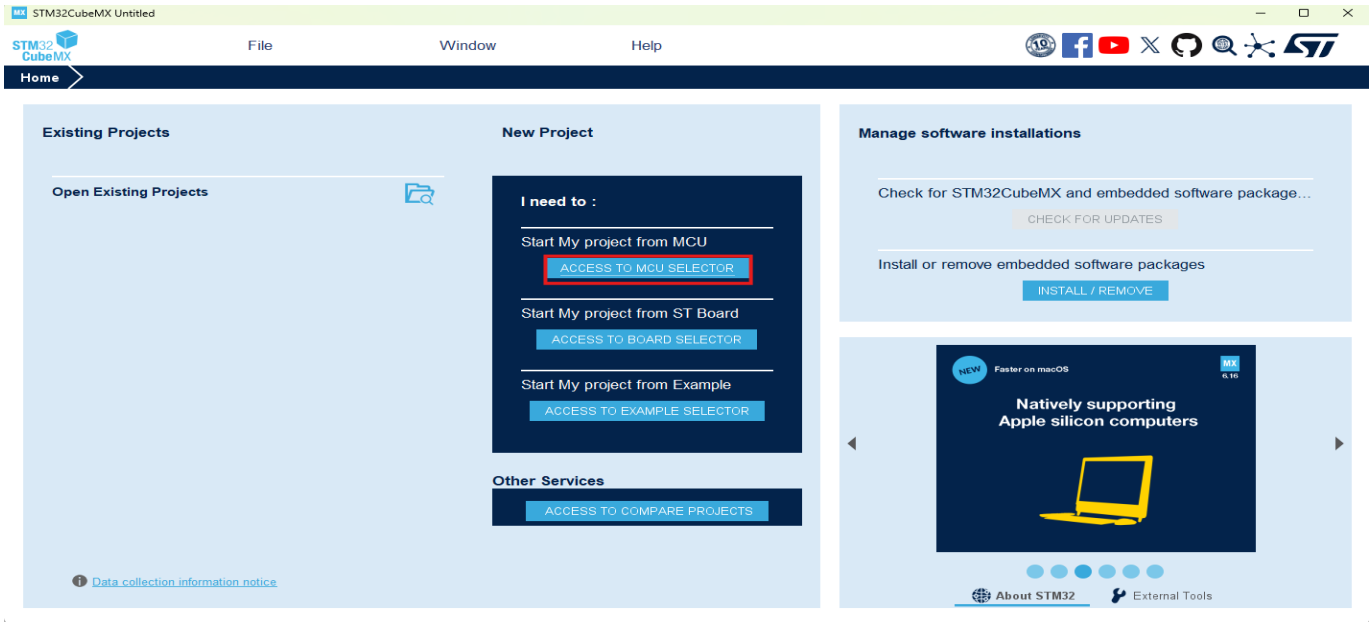
Objective: To understand basic GPIO configuration and control by blinking an on-board LED using the STM32 microcontroller.

Learning Outcomes: Configure GPIO pins as output using STM32CubeIDE. Generate delays and control LED ON/OFF through embedded C code

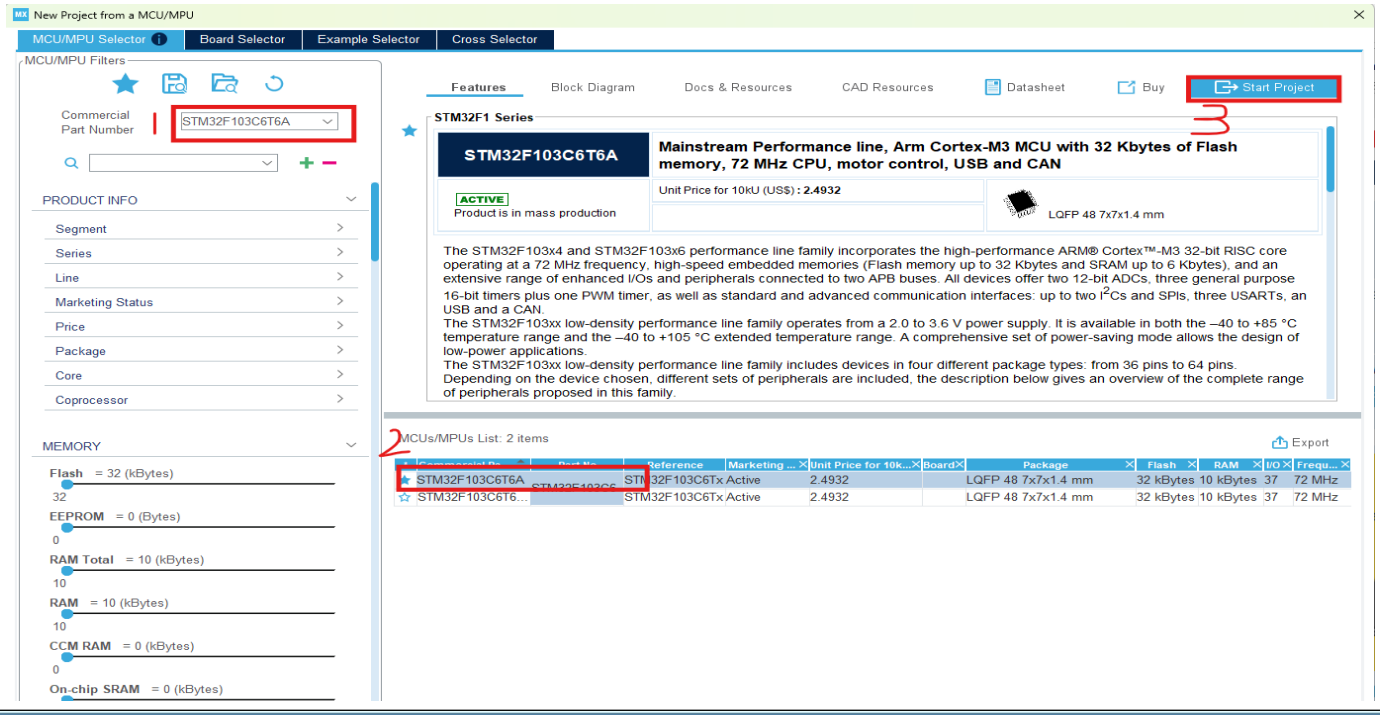
Procedure/Setup:

Step-1: Open stm32Cubemx

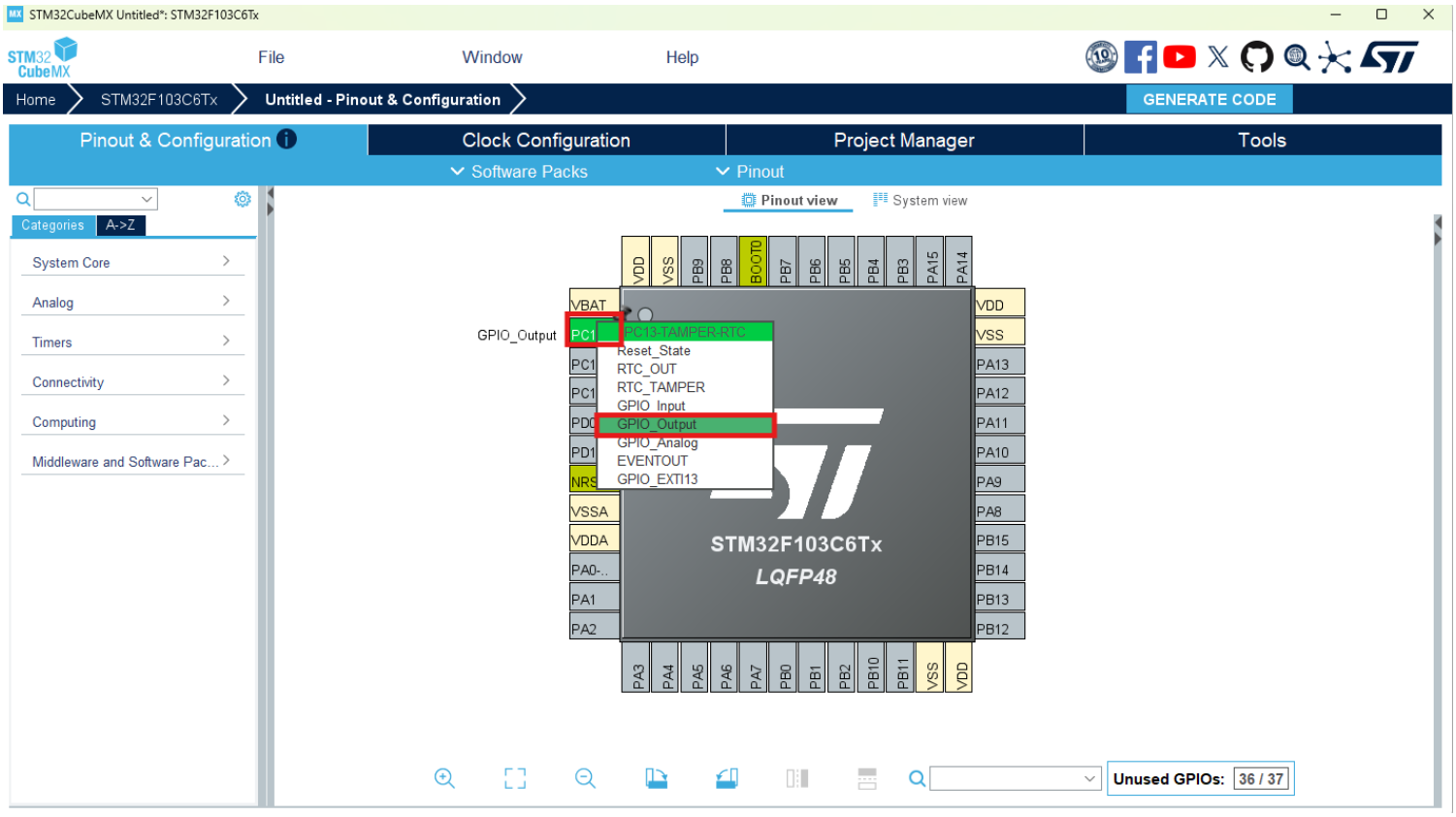
Step-2:



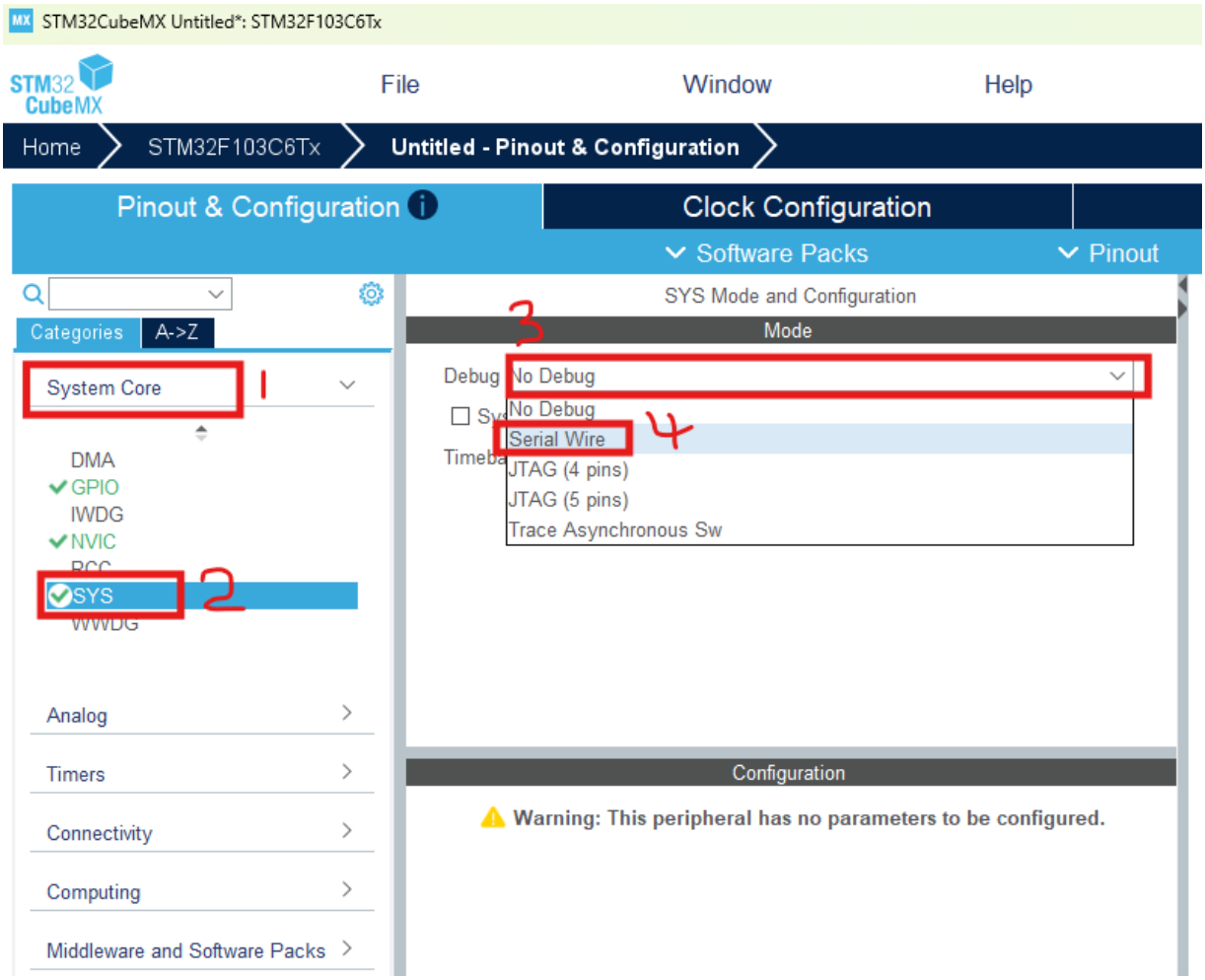
Step-3:



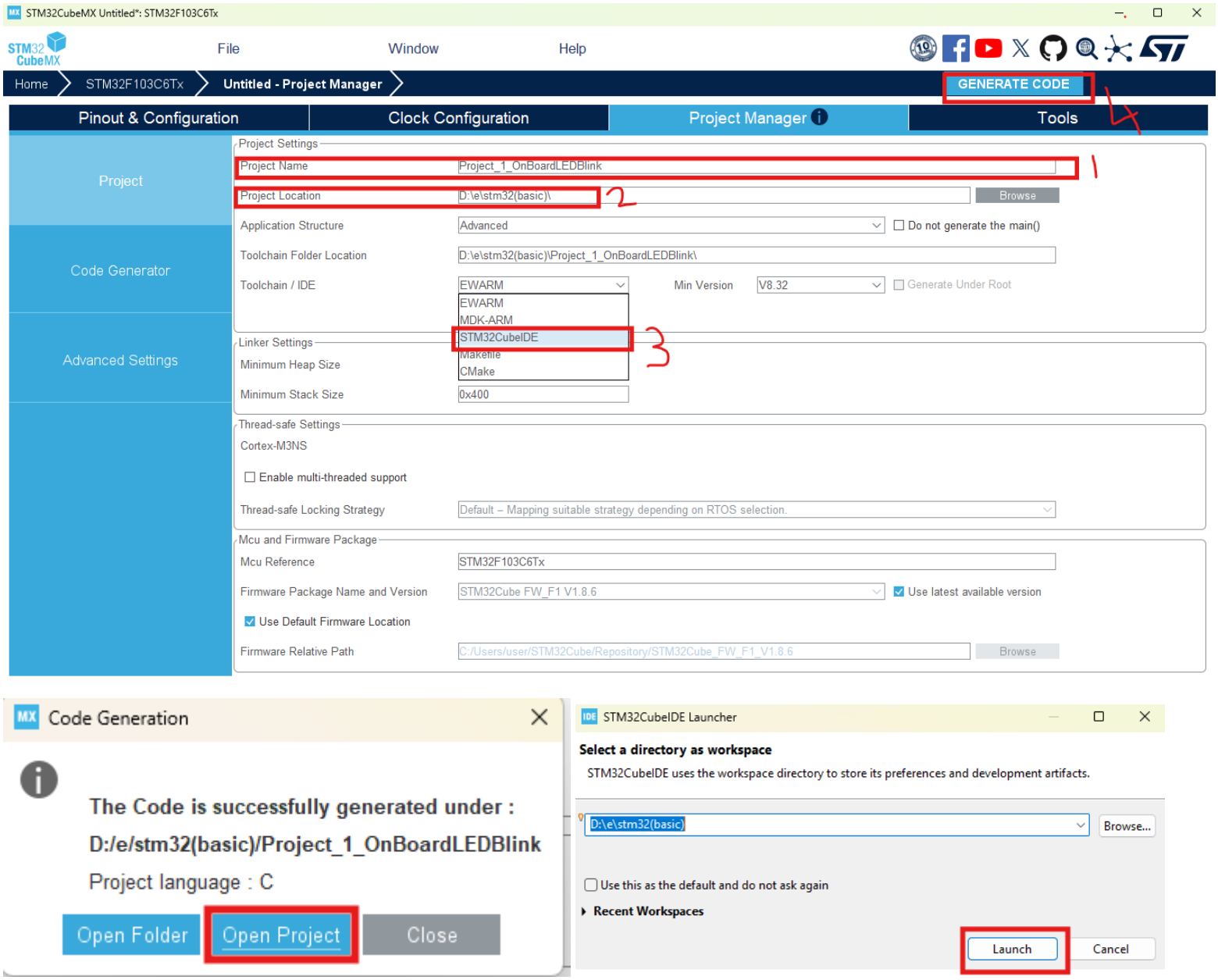
Step-4:


STM32CubeMX Untitled*: STM32F103C6Tx
File
Window
Help
Home
STM32F103C6Tx
Untitled - Pinout & Configuration
GENERATE CODE
Pinout & Configuration
Clock Configuration
Project Manager
Tools
Software Packs
Pinout
Categories
A->Z
System Core
Analog
Timers
Connectivity
Computing
Middleware and Software Pac...
Pinout view
System view
GPIO_Output
PC13-TAMPER-RTC
Reset_State
RTC_OUT
RTC_TAMPER
GPIO_Input
GPIO_Output
GPIO_Analog
EVENTOUT
GPIO_EXTI13
STM32F103C6Tx
LQFP48
Unused GPIOs: 36 / 37

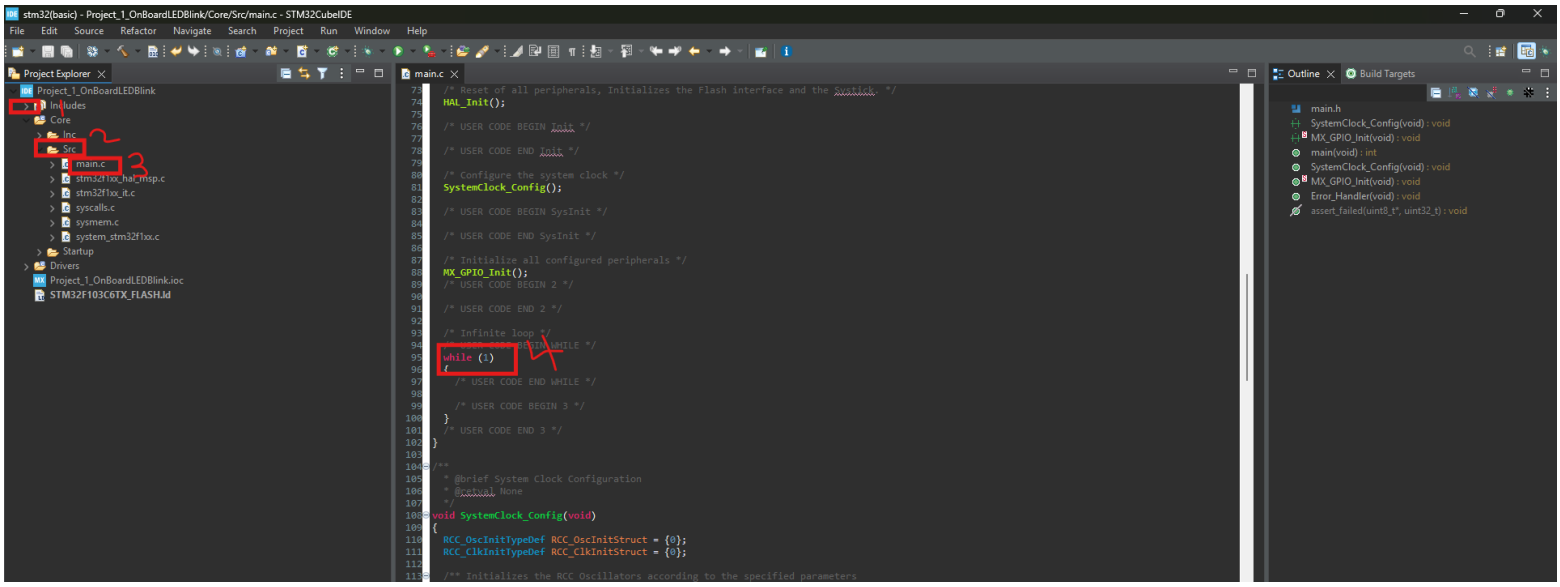
Step-5:


STM32CubeMX Untitled*: STM32F103C6Tx
File
Window
Help
Home
STM32F103C6Tx
Untitled - Pinout & Configuration
Pinout & Configuration
Clock Configuration
Software Packs
Pinout
Categories
A->Z
System Core
1
DMA
GPIO
IWDG
NVIC
RCC
SYS
2
WWDG
Analog
Timers
Connectivity
Computing
Middleware and Software Packs
SYS Mode and Configuration
3
Mode
Debug
No Debug
No Debug
Serial Wire
4
JTAG (4 pins)
JTAG (5 pins)
Trace Asynchronous Sw
Configuration
Warning: This peripheral has no parameters to be configured.

Step-6:



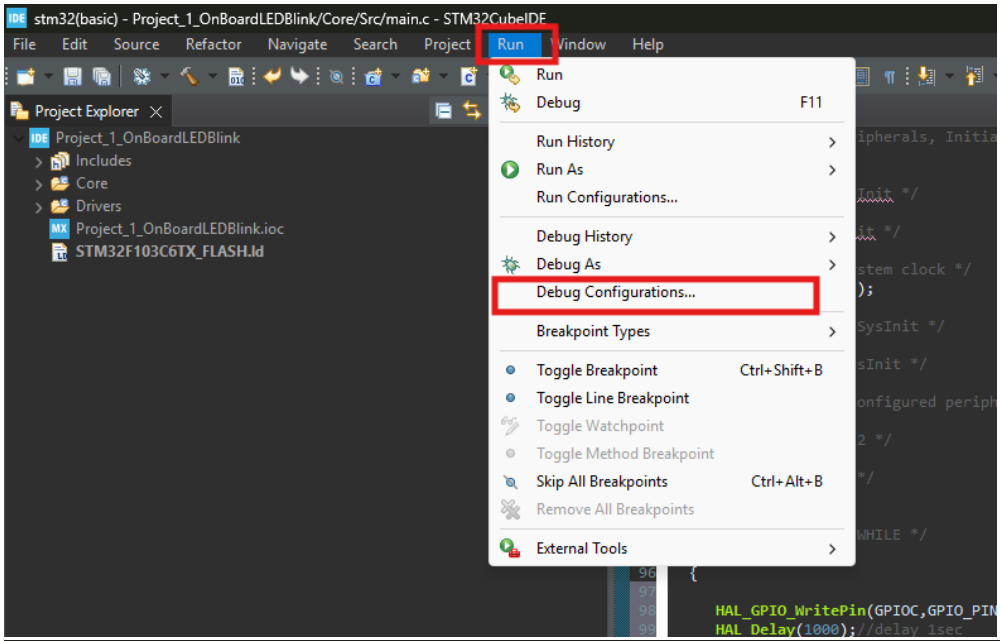
Step-7: Expand **Core** → Expand **src** → **main.c** → Scroll to **while(1)**

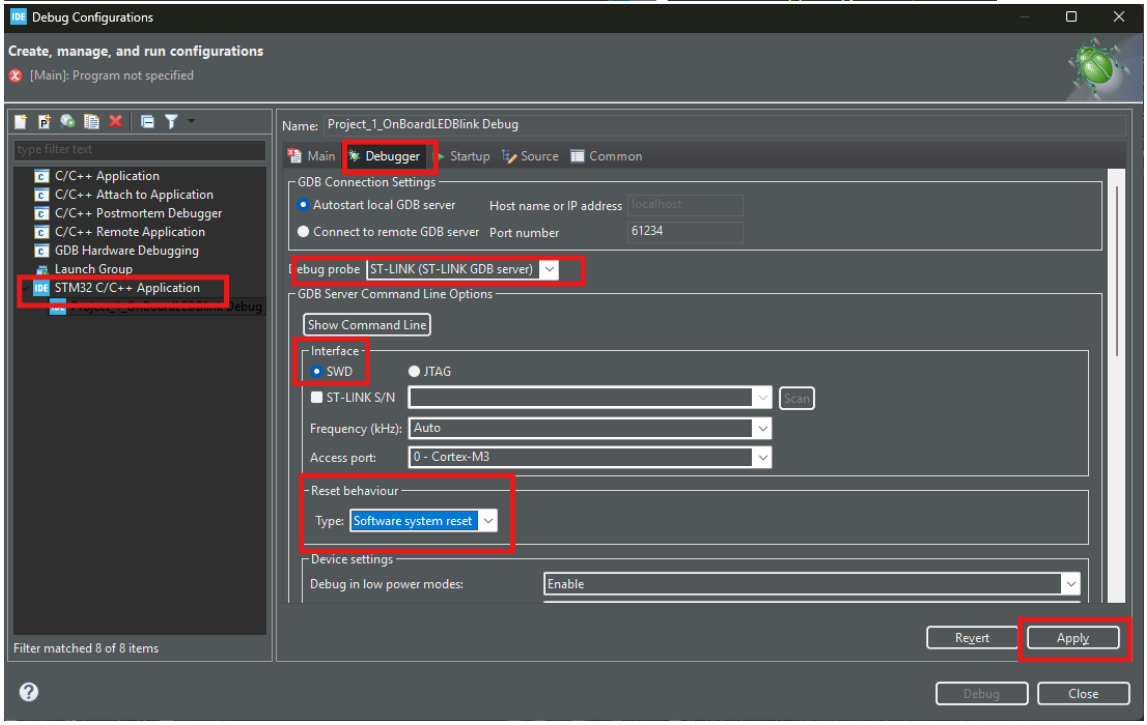


Code:

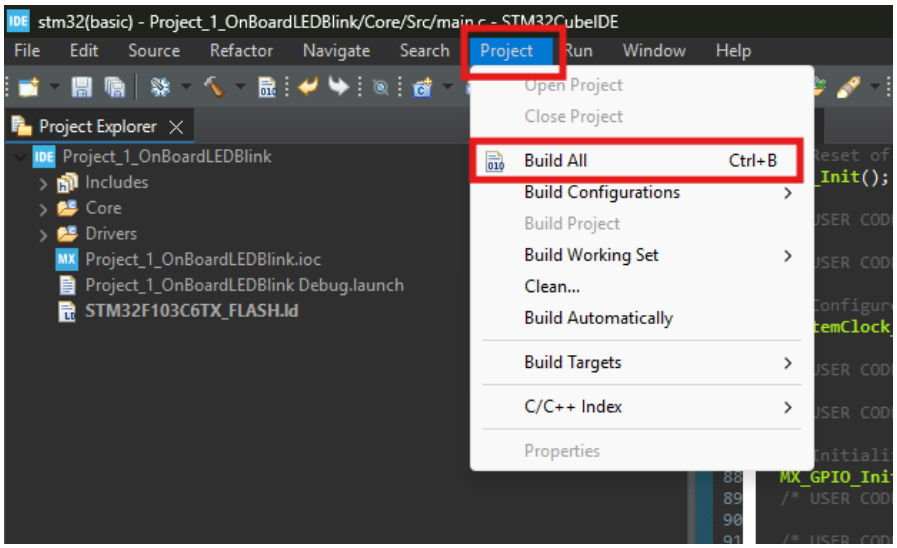
```
while (1)
{

    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,1);
    HAL_Delay(1000);//delay 1sec
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,0);
    HAL_Delay(1000);
  /* USER CODE END WHILE */

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

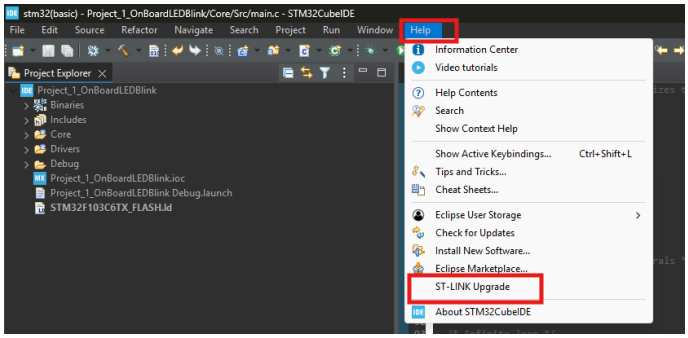
Step-8:

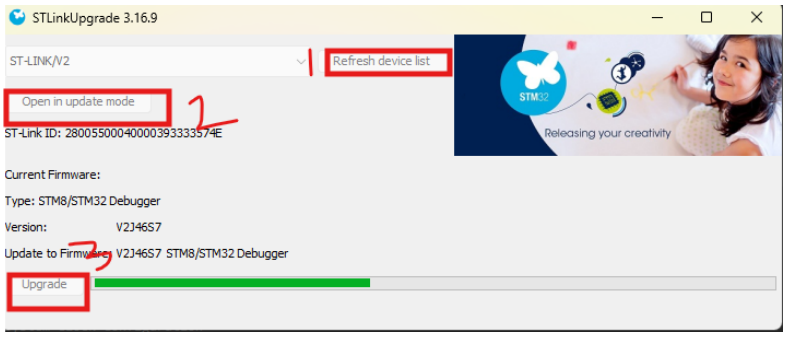




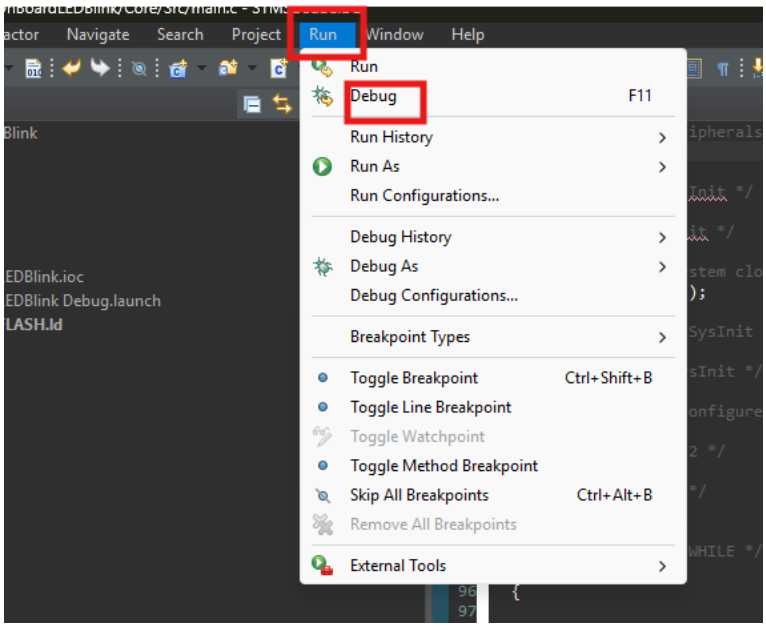
Step-9:

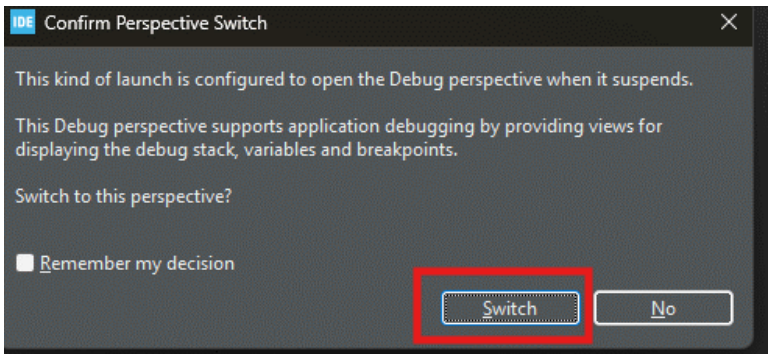




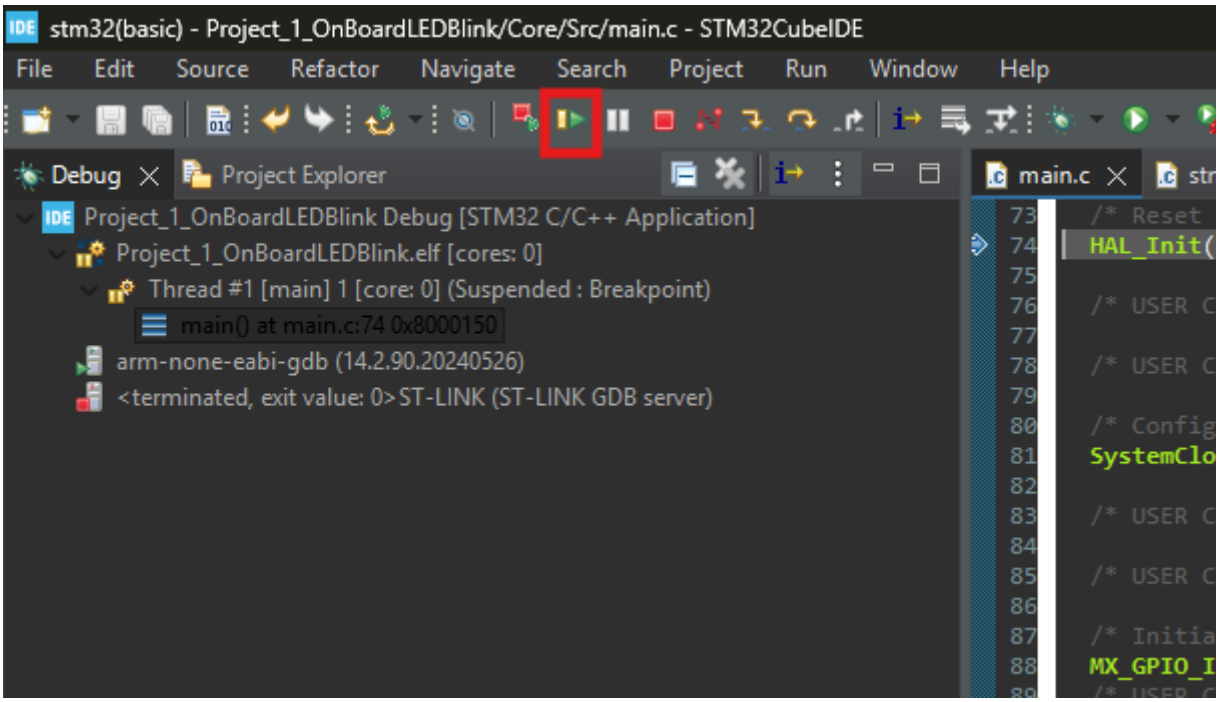


Step-10:

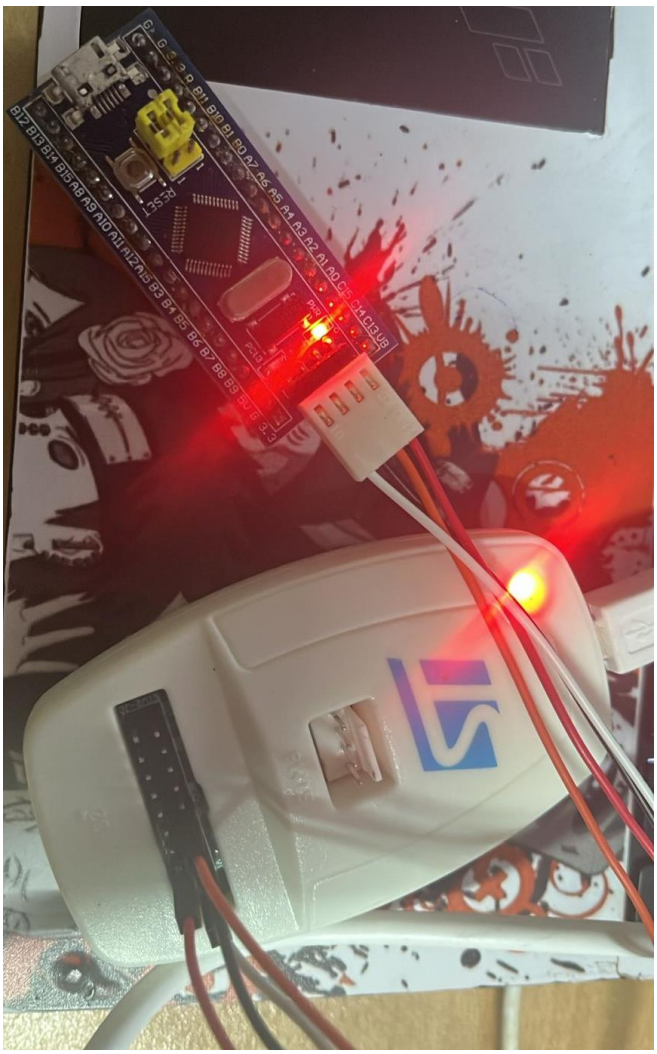




Step-11:



Output:



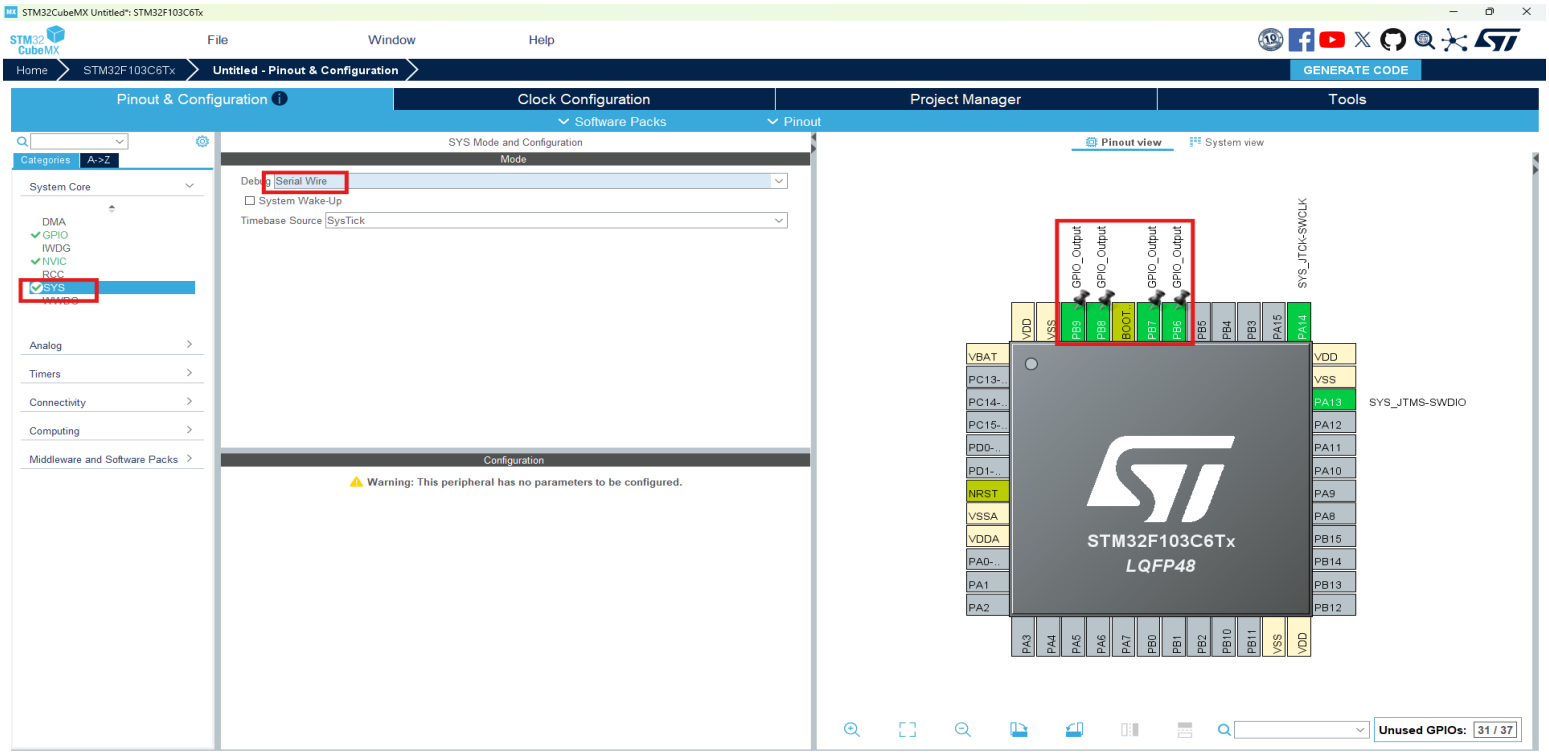
Project-2: Blink External LED

Objective: To interface and control external LEDs using GPIO pins of the STM32 microcontroller and generate a blinking pattern.

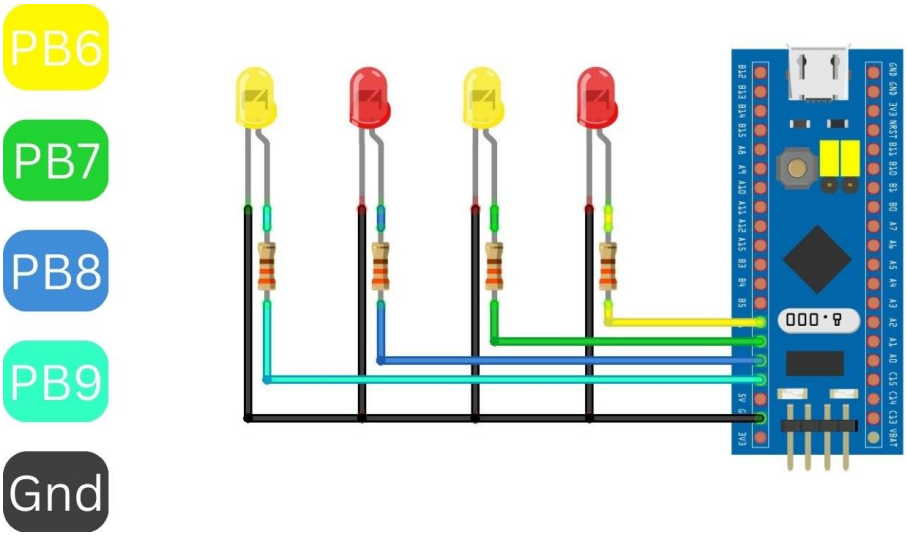
Learning Outcomes: Interface external LEDs with proper current-limiting resistors. Configure GPIO pins as output and control external LEDs using STM32CubeIDE

Procedure/Setup:

Step-1:



Circuit diagram:

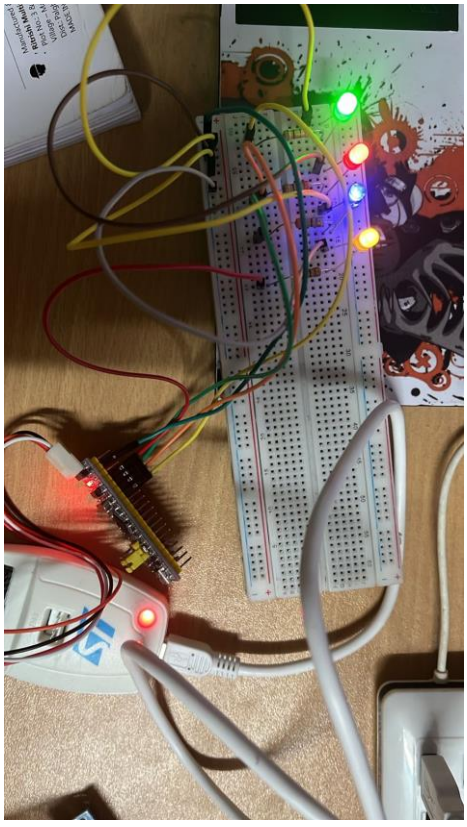


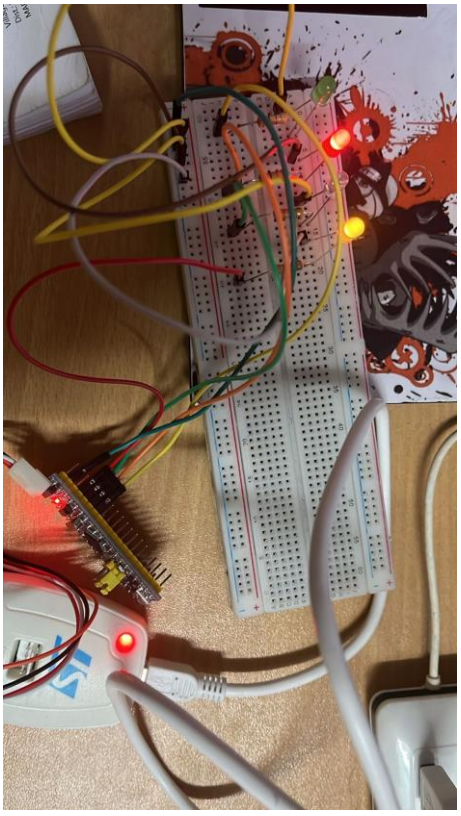
Code:

```
/* USER CODE BEGIN WHILE */
while (1)
{
    //Turn on all LEDs
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_9,1);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_8,1);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_7,1);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,1);
    HAL_Delay(1000);
    //Alternate-1
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_9,1);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_8,0);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_7,1);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,0);
    HAL_Delay(1000);
    //Alternate-2
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_9,0);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_8,1);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_7,0);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,1);
    HAL_Delay(1000);
    //Turn off all LEDs
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_9,0);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_8,0);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_7,0);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,0);
    HAL_Delay(1000);
  /* USER CODE END WHILE */

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

Close

Output:





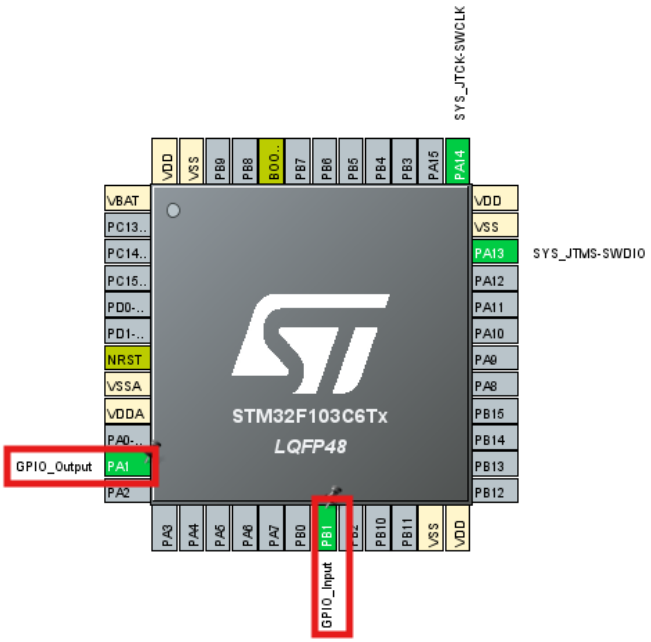
Project-3: Button controlled LED

Objective: To interface a push button with the STM32 microcontroller and control an external LED based on button input.

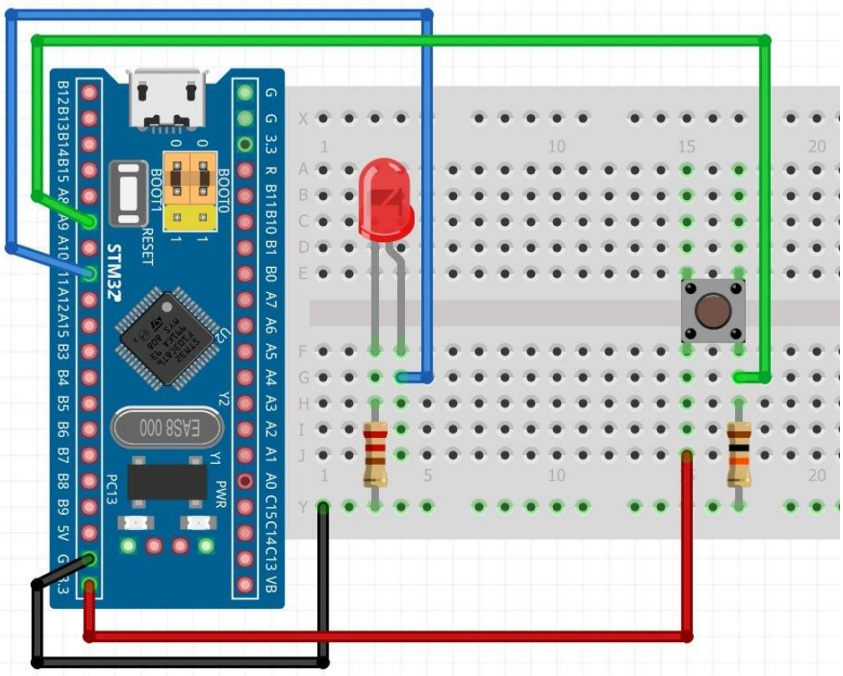
Learning Outcomes: Configure GPIO pins as input (button) and output (LED). Read button status and control an LED using embedded C in STM32CubeIDE

Procedure/Setup:

Step-1:



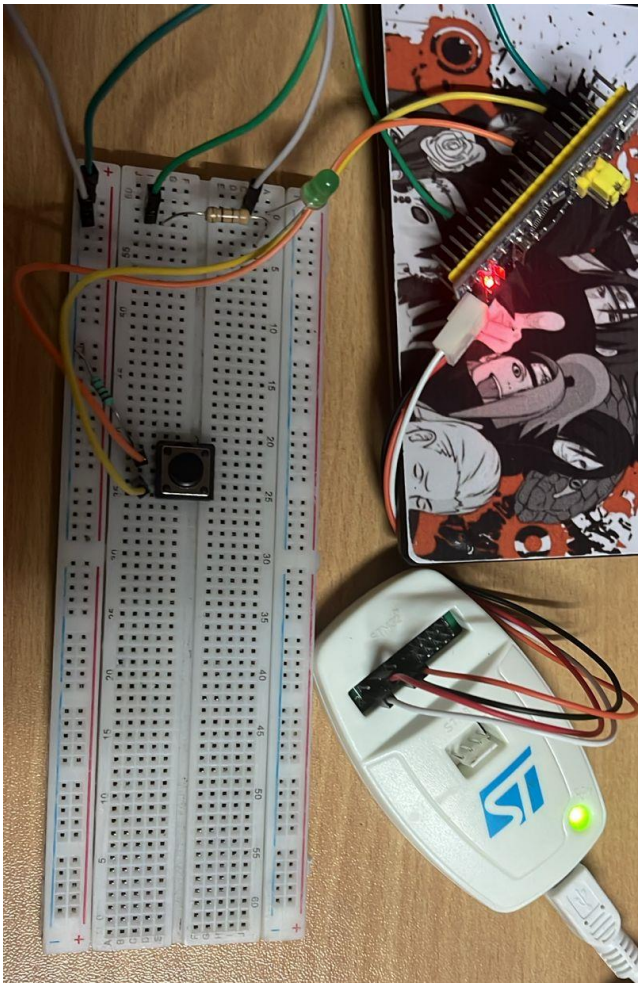
Circuit Diagram:

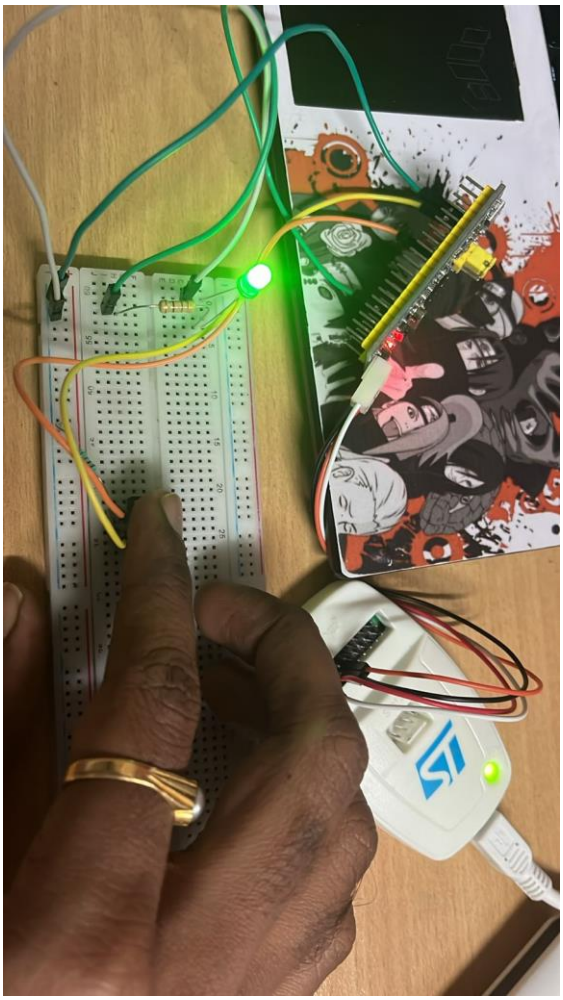


Code:

```
  while (1)
  {

    // Read the current button state
      if (HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_1) == GPIO_PIN_SET)
          {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);   // LED ON
          }
      else
          {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET); // LED OFF
          }
    /* USER CODE END WHILE */

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

Output:





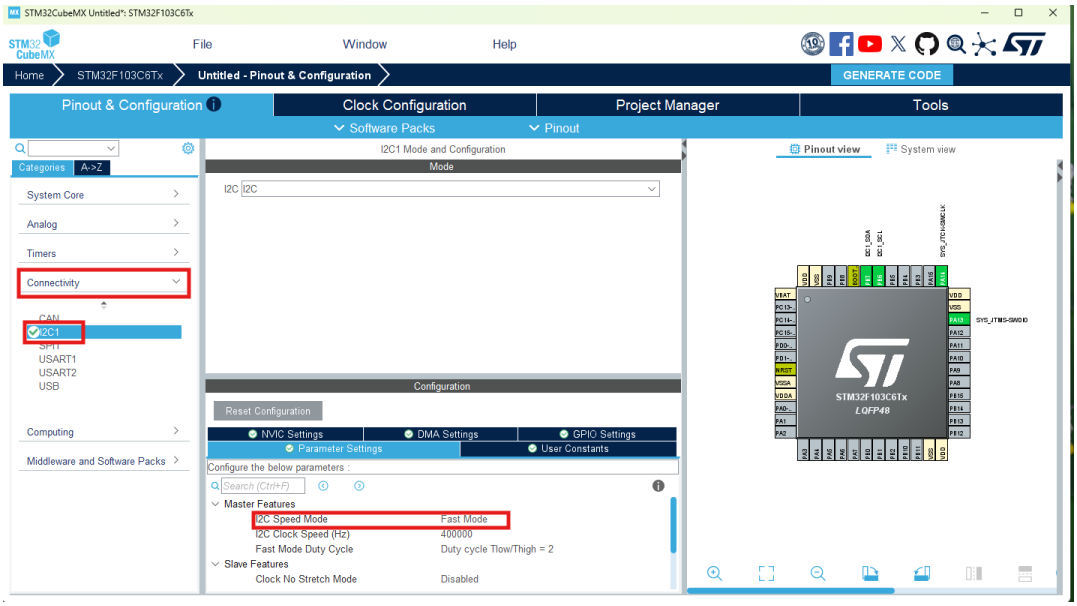
Project-4: Interfacing OLED

Objective: To interface an OLED display with the STM32 microcontroller and display basic text or graphics using a communication protocol.

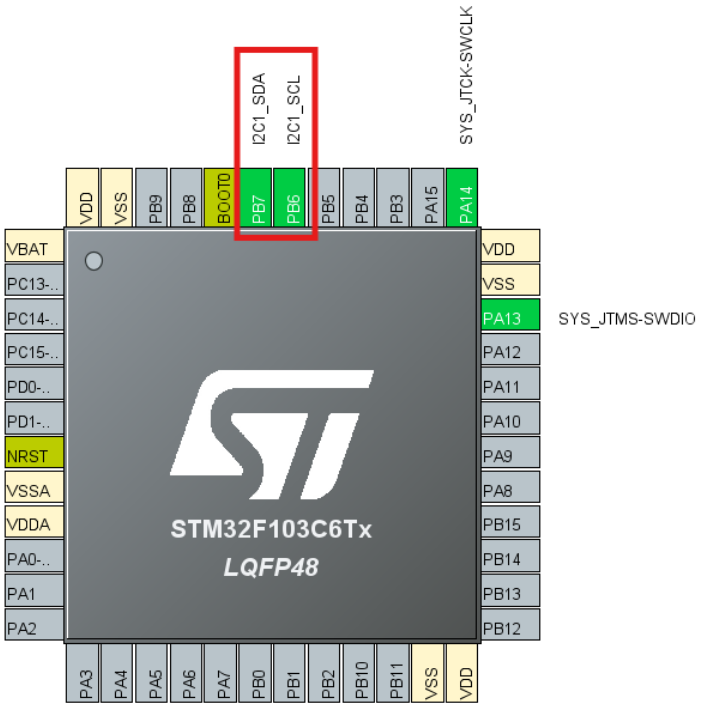
Learning Outcomes: Interface an OLED display using I2C protocol. Initialize the display and send data/commands using STM32CubeIDE.

Setup/Procedure:

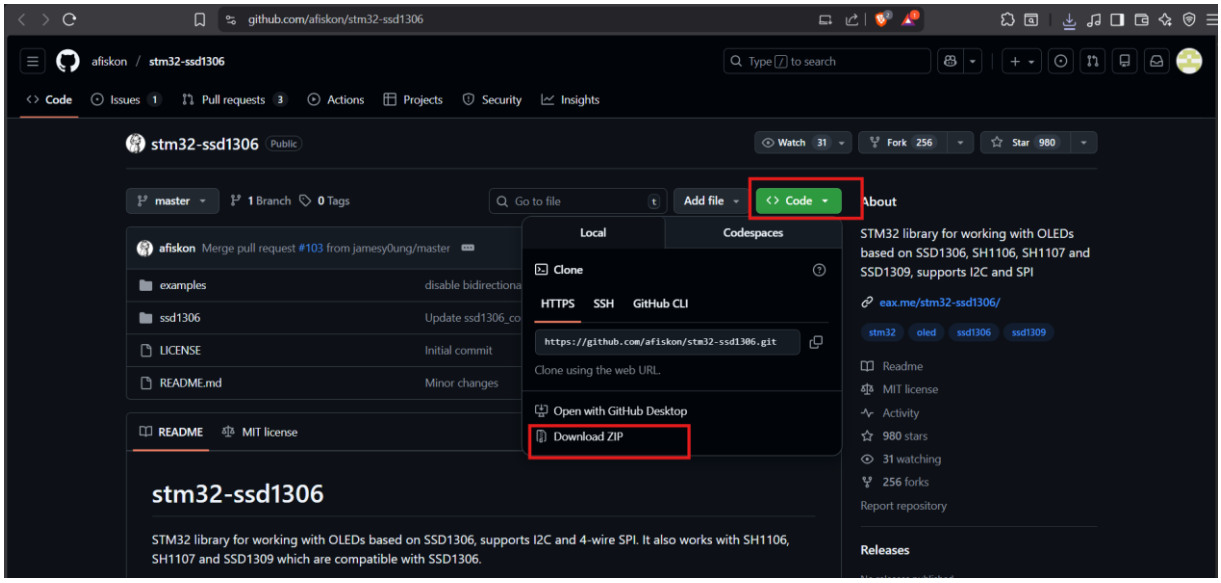
Step-1:



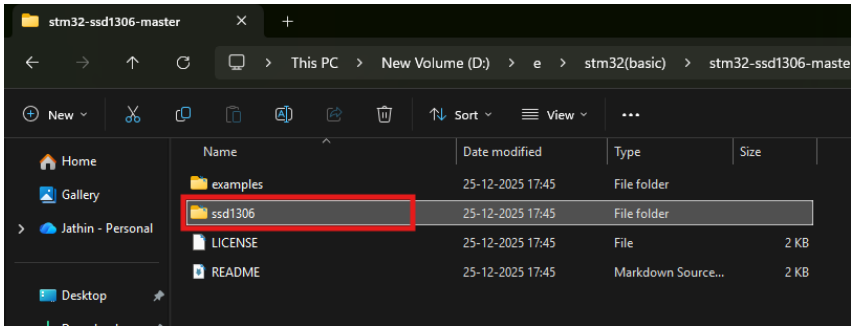
Step-2:

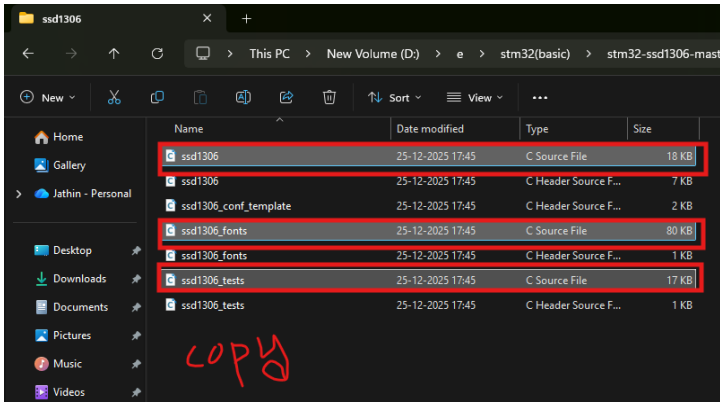


Step-3: visit https://github.com/afiskon/stm32-ssd1306

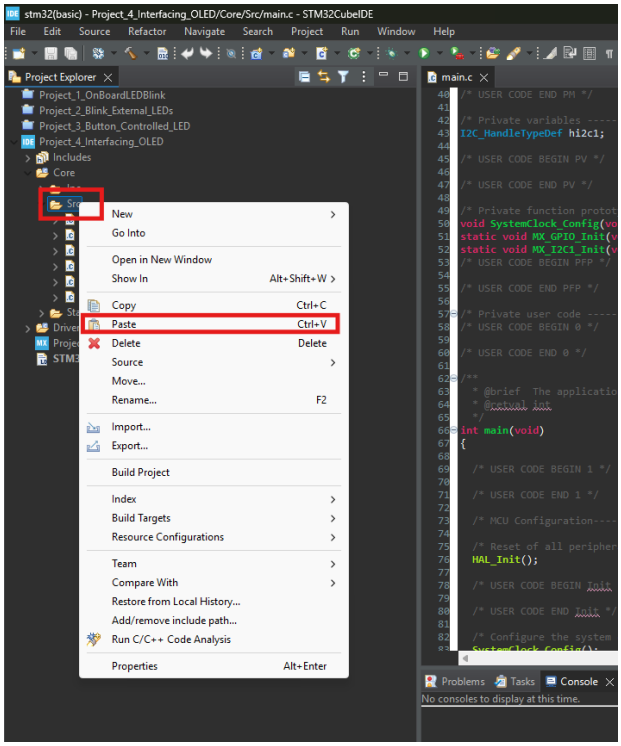


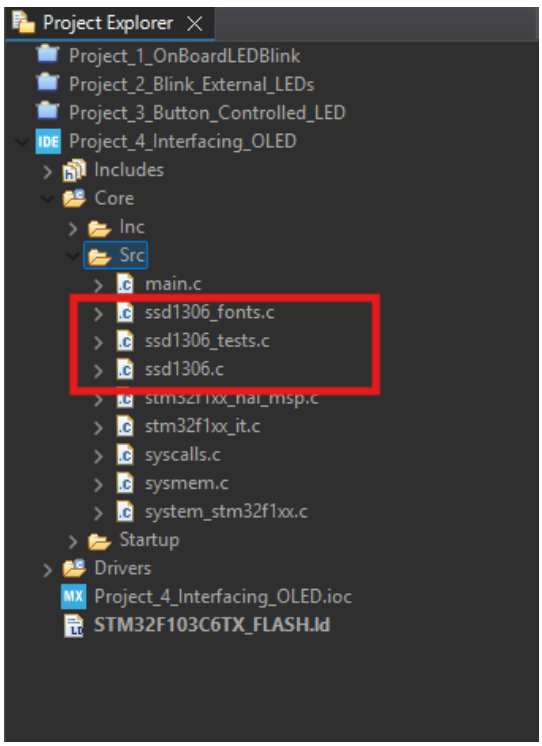
Step-4:



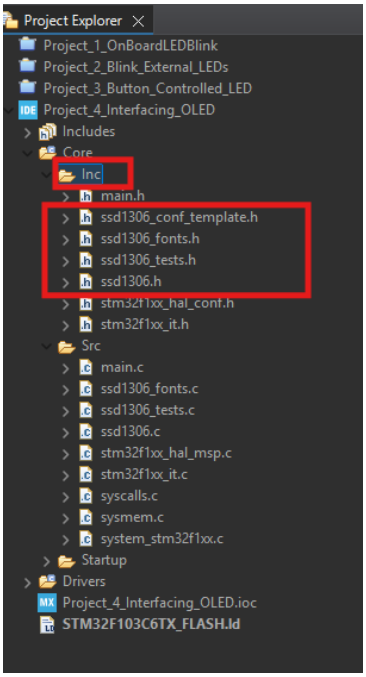


Step-5:

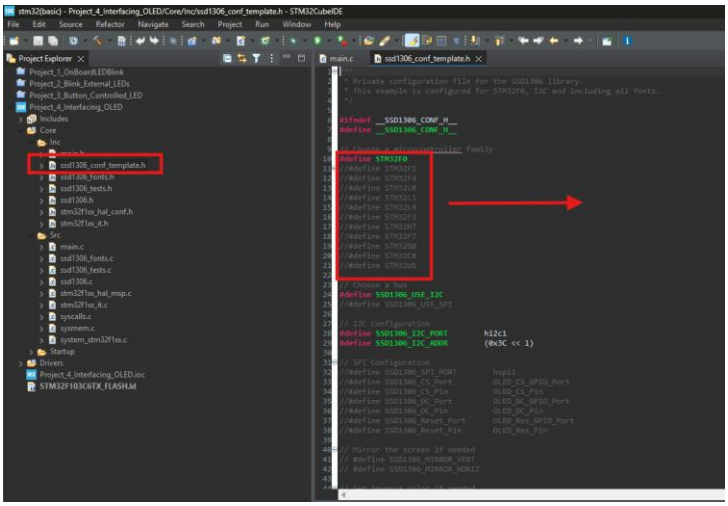


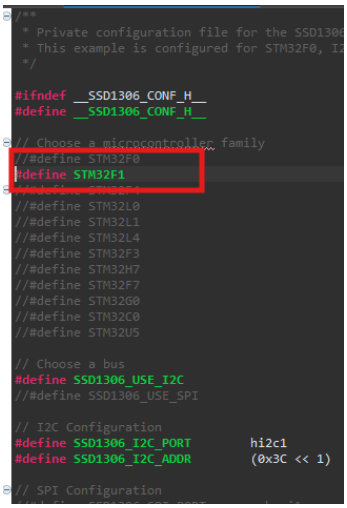


Step-6: Repeat the same process with header source file and paste them in Inc.

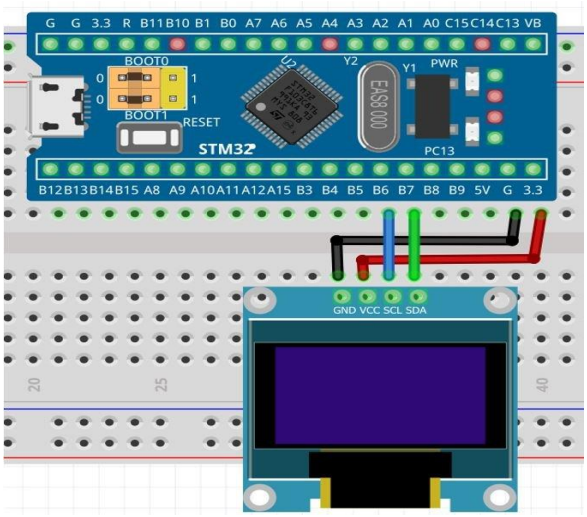


Step-7:





Circuit diagram:

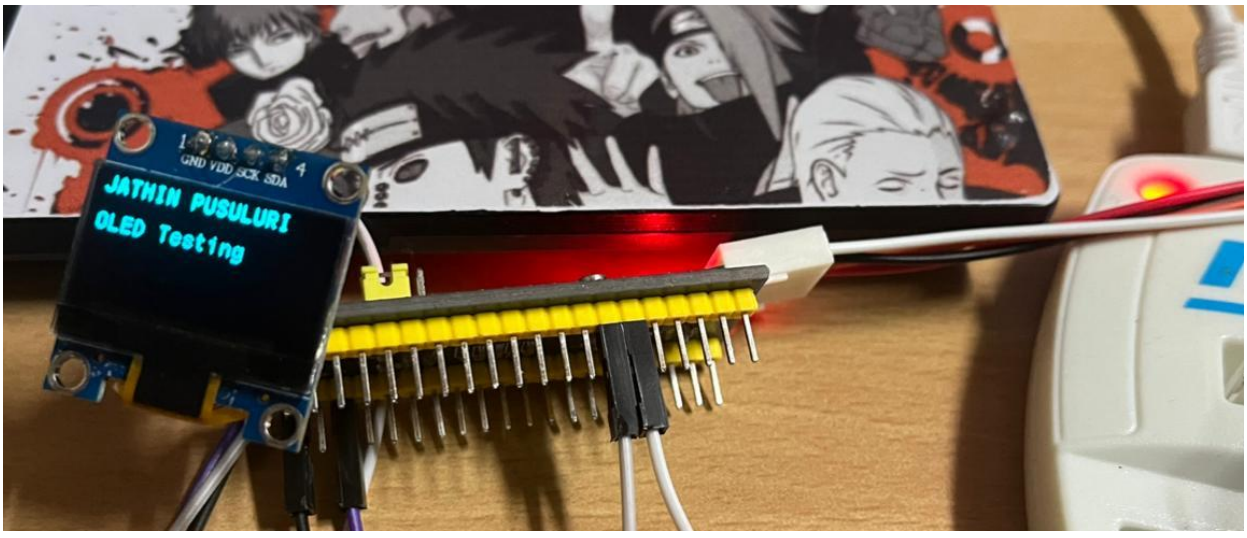


Code:

```c
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "ssd1306.h"
#include "ssd1306_tests.h"
#include "ssd1306_fonts.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */
```

```c
/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_I2C1_Init();

ssd1306_Init();
ssd1306_Fill(Black);
ssd1306_SetCursor(0, 0);
ssd1306_WriteString("JATHIN PUSULURI", Font_7x10, White);
ssd1306_SetCursor(0, 20);
ssd1306_WriteString("OLED Testing", Font_7x10, White);
ssd1306_UpdateScreen();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

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

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

Output:



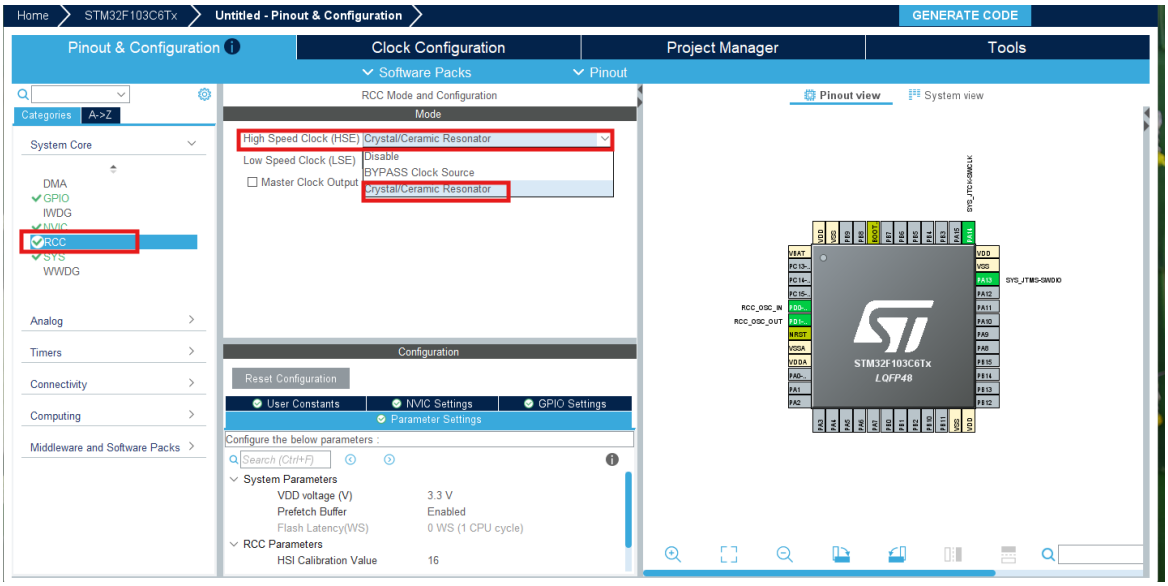
Project-5: Distance Measurement using Ultrasonic sensor

Objective: To measure distance using an ultrasonic sensor interfaced with the STM32 microcontroller by calculating time of echo signal.

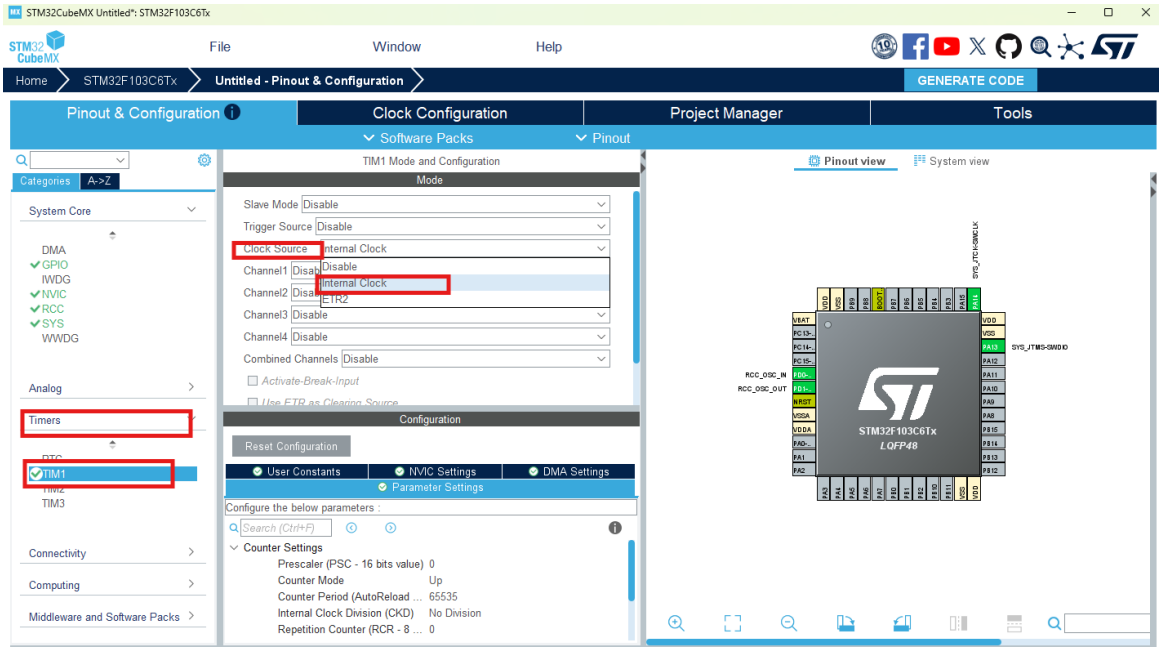
Learning Outcomes: Interface an ultrasonic sensor using GPIO and timer/delay functions. Calculate distance based o ultrasonic pulse timing using embedded C.

Setup/Procedure:

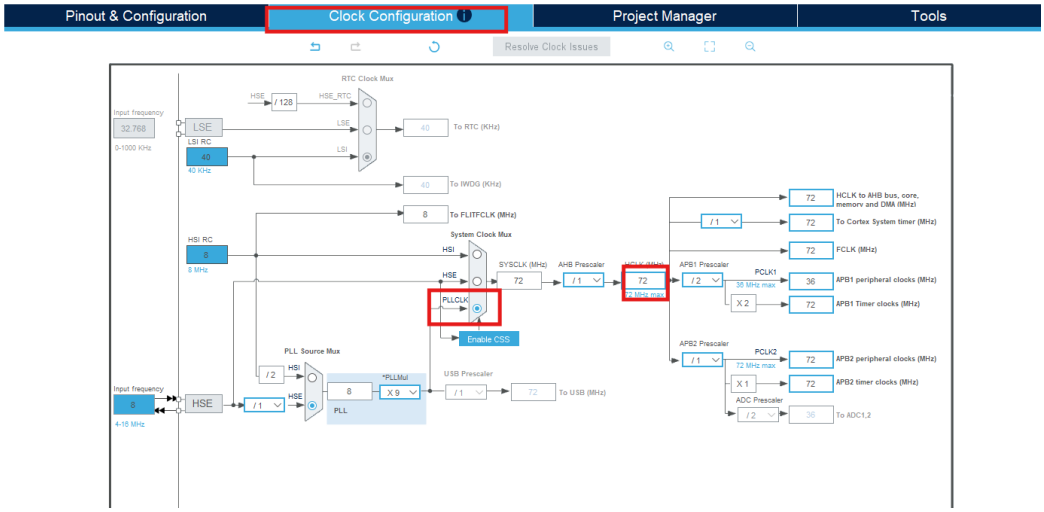
Step-1:



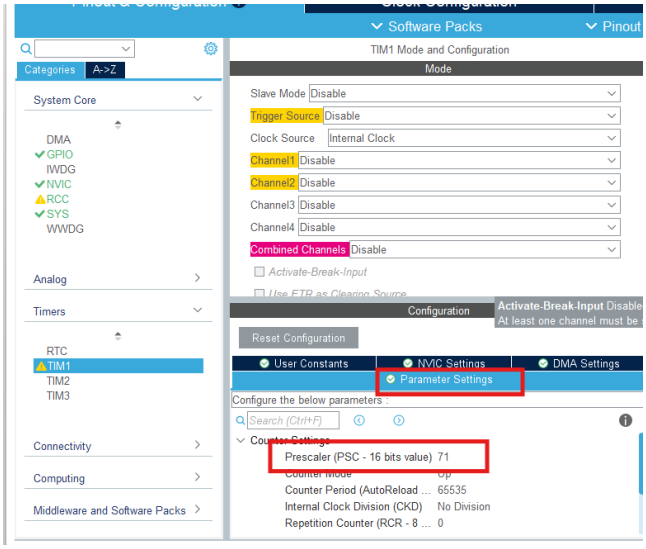
Step-2:



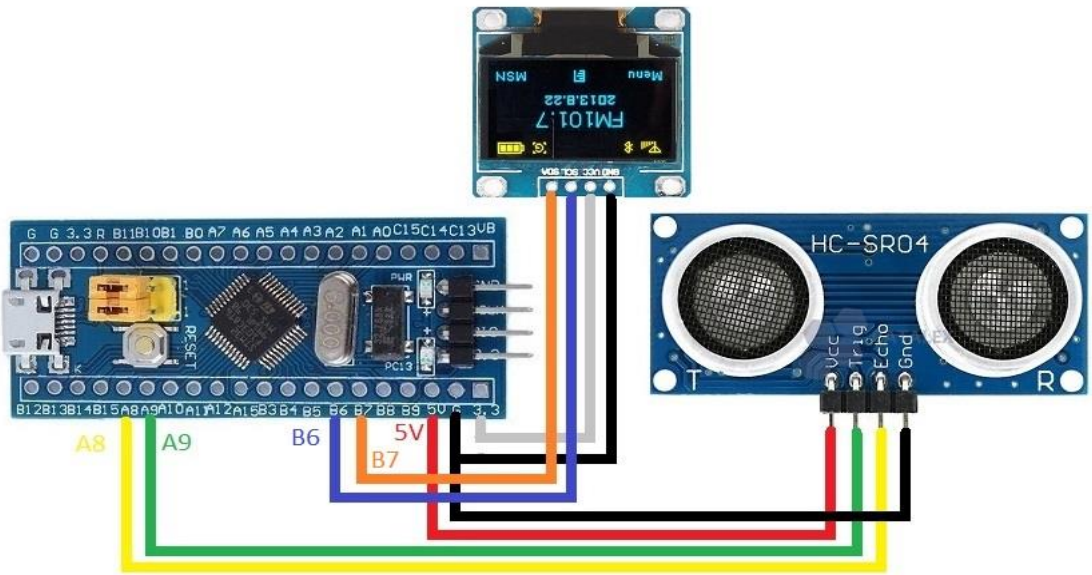
Step-3:



Step-4:



Circuit diagram:



Jathin Pusuluri
p.jathin021@gmail.com
Jathin021

Code:

```c
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;

TIM_HandleTypeDef htim1;

/* USER CODE BEGIN PV */

/* USER CODE END PV */
#define TRIG_PIN GPIO_PIN_9
#define TRIG_PORT GPIOA
#define ECHO_PIN GPIO_PIN_8
#define ECHO_PORT GPIOA
uint32_t pMillis;
uint32_t Value1 = 0;
uint32_t Value2 = 0;
uint16_t Distance  = 0;  // cm
char strCopy[15];

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_TIM1_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
```

```c
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "fonts.h"
#include "ssd1306.h"
#include "stdio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */
```

```c
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_TIM1_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start(&htim1);
  HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_RESET);  // pull the TRIG pin low
  SSD1306_Init();

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
        HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_SET);  // pull the TRIG pin HIGH
        __HAL_TIM_SET_COUNTER(&htim1, 0);
        while (__HAL_TIM_GET_COUNTER (&htim1) < 10);  // wait for 10 us
        HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_RESET);  // pull the TRIG pin low

        pMillis = HAL_GetTick(); // used this to avoid infinite while loop  (for timeout)
        // wait for the echo pin to go high
        while (!(HAL_GPIO_ReadPin (ECHO_PORT, ECHO_PIN)) && pMillis + 10 >  HAL_GetTick());
        Value1 = __HAL_TIM_GET_COUNTER (&htim1);

        pMillis = HAL_GetTick(); // used this to avoid infinite while loop (for timeout)
        // wait for the echo pin to go low
        while ((HAL_GPIO_ReadPin (ECHO_PORT, ECHO_PIN)) && pMillis + 50 > HAL_GetTick());
        Value2 = __HAL_TIM_GET_COUNTER (&htim1);

        Distance = (Value2-Value1)* 0.034/2;

        SSD1306_GotoXY (0, 0);
        SSD1306_Puts ("Distance:", &Font_11x18, 1);
        sprintf(strCopy,"%d    ", Distance);
        SSD1306_GotoXY (0, 30);
        SSD1306_Puts (strCopy, &Font_16x26, 1);
        SSD1306_UpdateScreen();
        HAL_Delay(50);

    /* USER CODE BEGIN 3 */
```

Output: