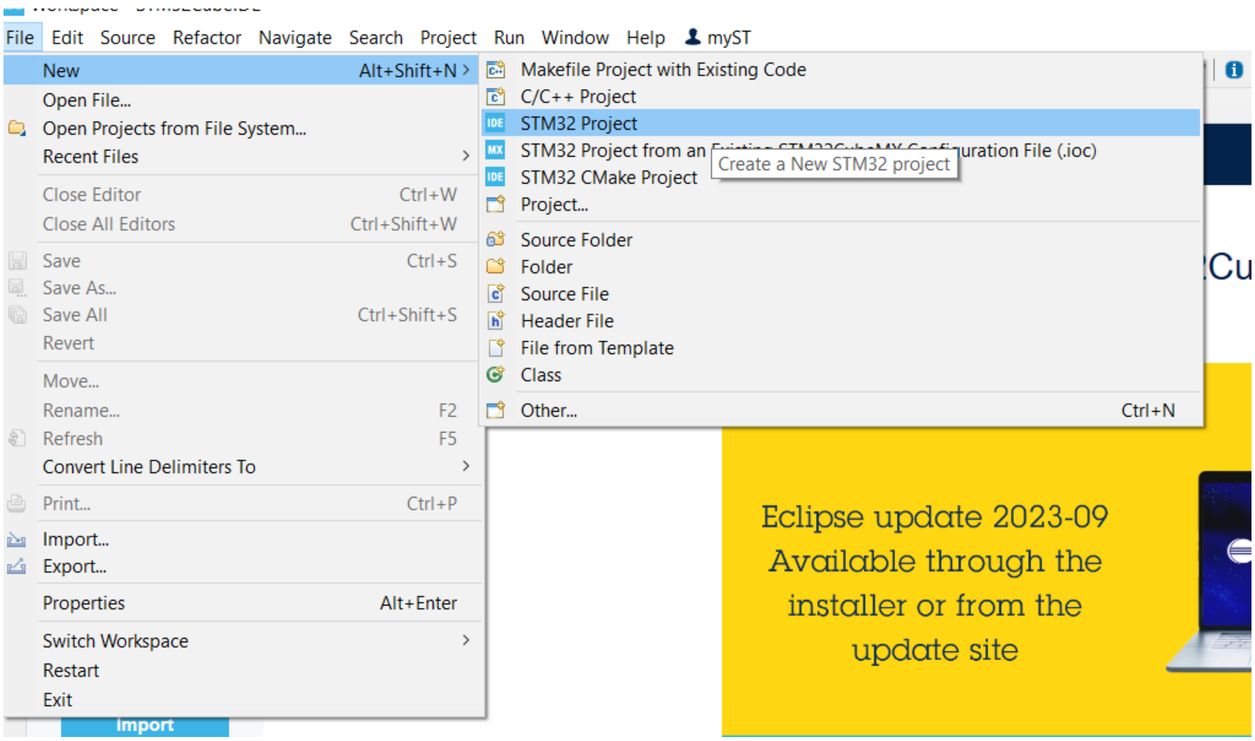
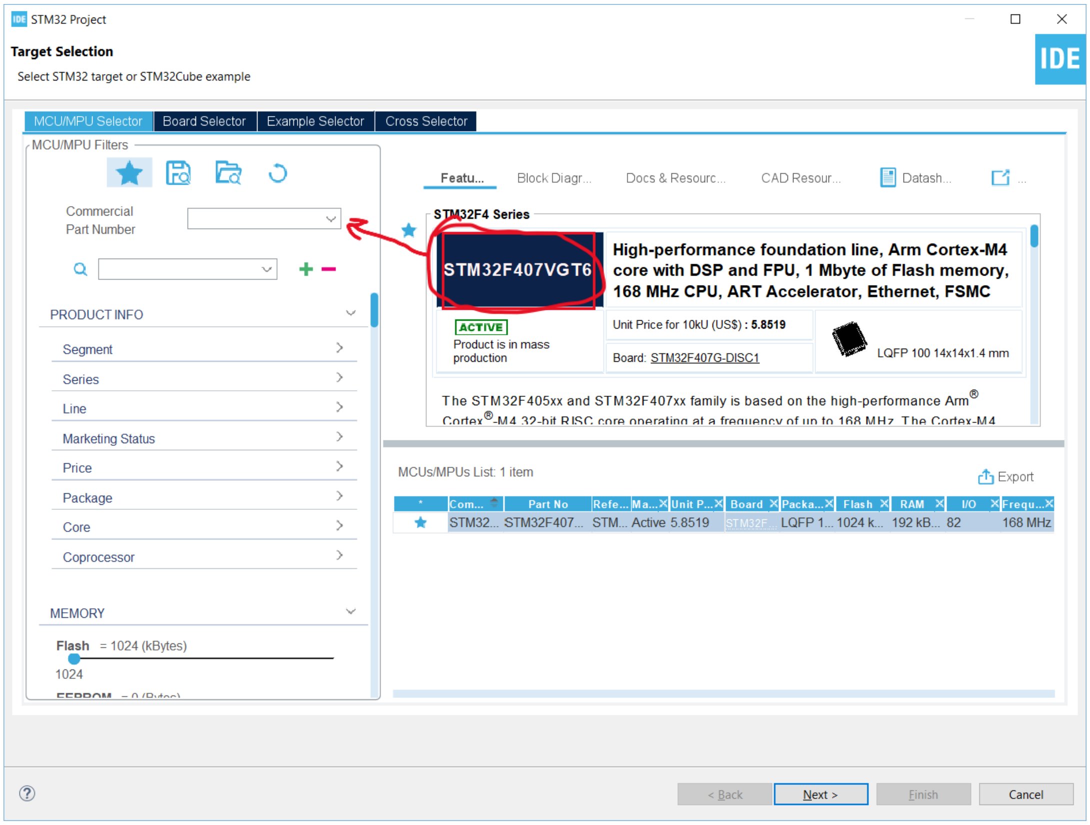

File Edit Source Refactor Navigate Search Project Run Window Help myST
New Alt+Shift+N
Open File...
Open Projects from File System...
Recent Files
Close Editor Ctrl+W
Close All Editors Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Import...
Export...
Properties Alt+Enter
Switch Workspace
Restart
Exit
Makefile Project with Existing Code
C/C++ Project
STM32 Project
STM32 Project from an Existing STM32CubeMX Configuration File (.ioc)
Create a New STM32 project
STM32 CMake Project
Project...
Source Folder
Folder
Source File
Header File
File from Template
Class
Other... Ctrl+N
Eclipse update 2023-09 Available through the installer or from the update site
Import

starting by making new project file .



Target Selection

Select STM32 target or STM32Cube example

STM32F4 Series

STM32F407VGT6

High-performance foundation line, Arm Cortex-M4 core with DSP and FPU, 1 Mbyte of Flash memory, 168 MHz CPU, ART Accelerator, Ethernet, FSMC

ACTIVE
Product is in mass production

Unit Price for 10kU (US$) : 5.8519

Board: STM32F407G-DISC1

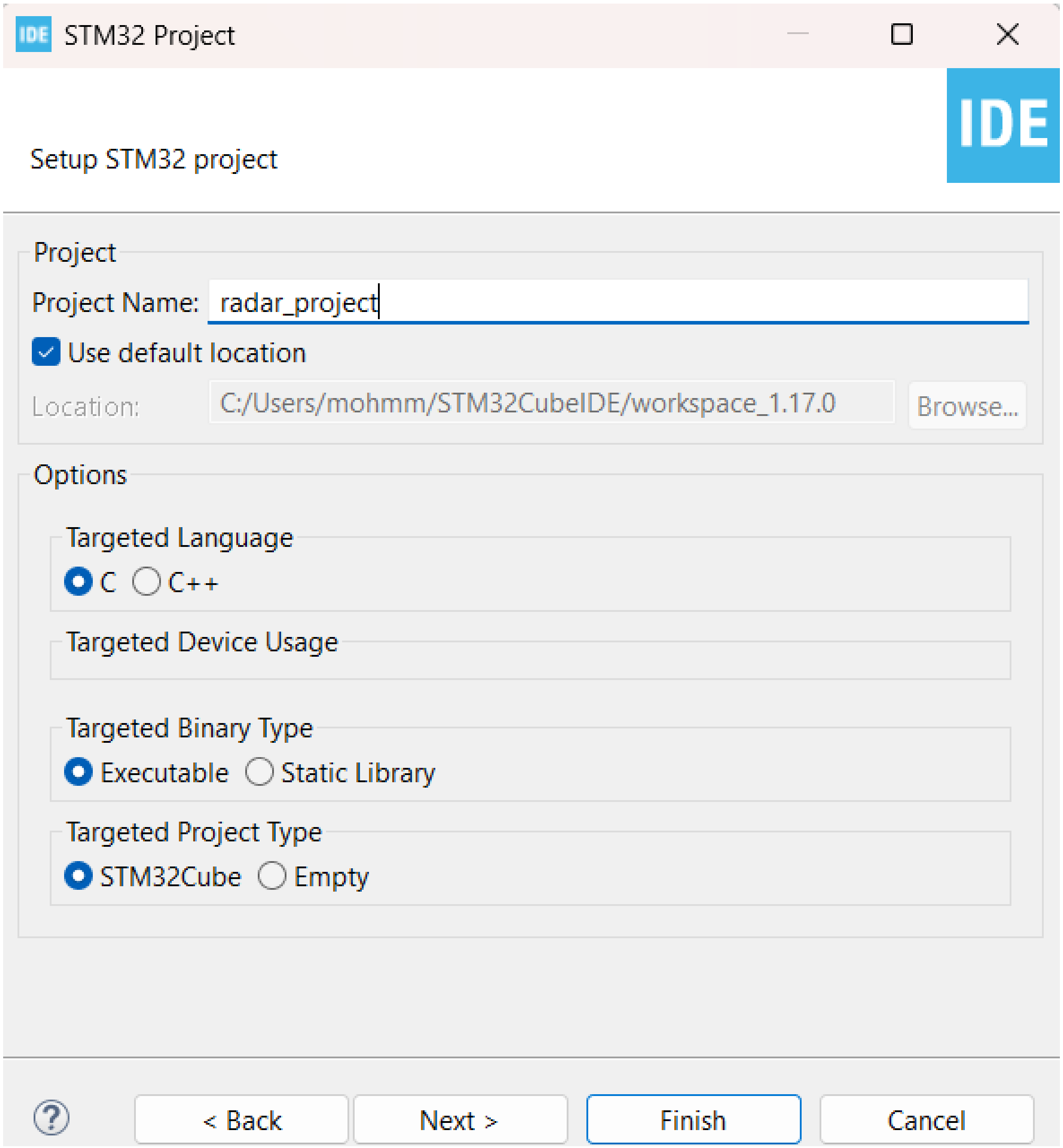
LQFP 100 14x14x1.4 mm

The STM32F405xx and STM32F407xx family is based on the high-performance Arm® Cortex®-M4 32-bit RISC core operating at a frequency of up to 168 MHz. The Cortex-M4

MCUs/MPUs List: 1 item

*	Com...	Part No	Refe...	Ma...	Unit P...	Board	Packa...	Flash	RAM	I/O	Frequ...
★	STM32...	STM32F407...	STM...	Active	5.8519	STM32F	LQFP 1...	1024 k...	192 kB...	82	168 MHz

Now you should chose the right stm32 which is stm32F407VGT6.



STM32 Project

Setup STM32 project

Project

Project Name: radar_project

☑ Use default location

Location: C:/Users/mohmm/STM32CubeIDE/workspace_1.17.0 Browse...

Options

Targeted Language

● C ○ C++

Targeted Device Usage

Targeted Binary Type

● Executable ○ Static Library

Targeted Project Type

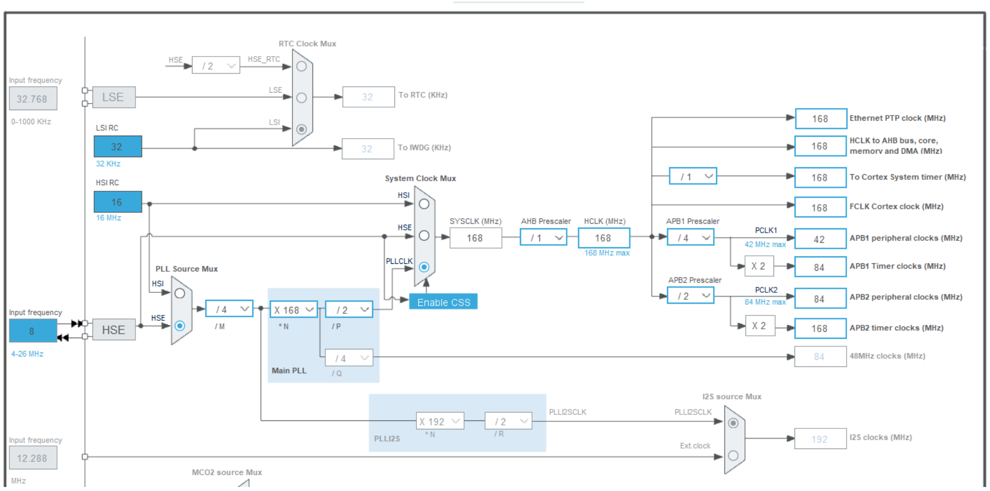
● STM32Cube ○ Empty

1. From Pinout and Configurations do the followings:

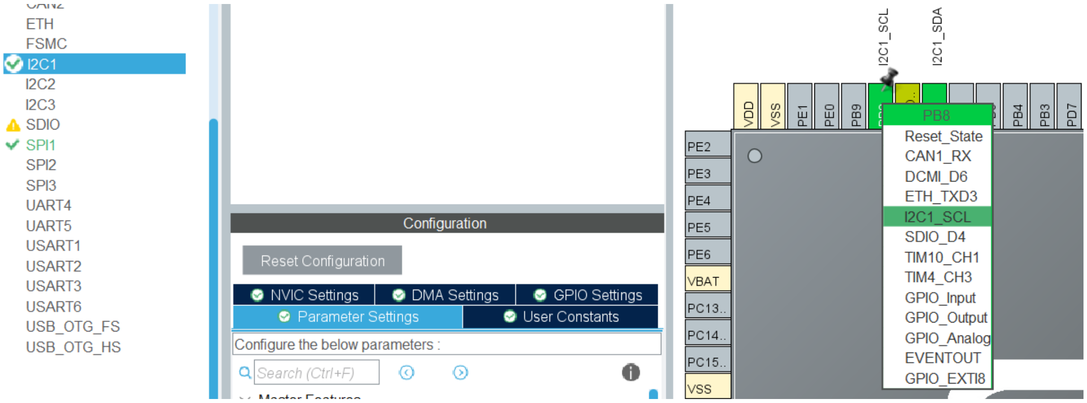
   a. System Core>RCC>High speed clock HSE: Select Crystal/Ceramic, we want to use external 8MHz clock.

   b. System Core>SYS>Debug : Select Serial wire in order to enable printing to the console window.

   c. Connectivity>I2C1>I2C: Select I2C in order to enable it. Instead of using PB6 we are going to use PB8 pin as clock pin, therefore you should configure PB8 as I2C1_SCL alternate mode.

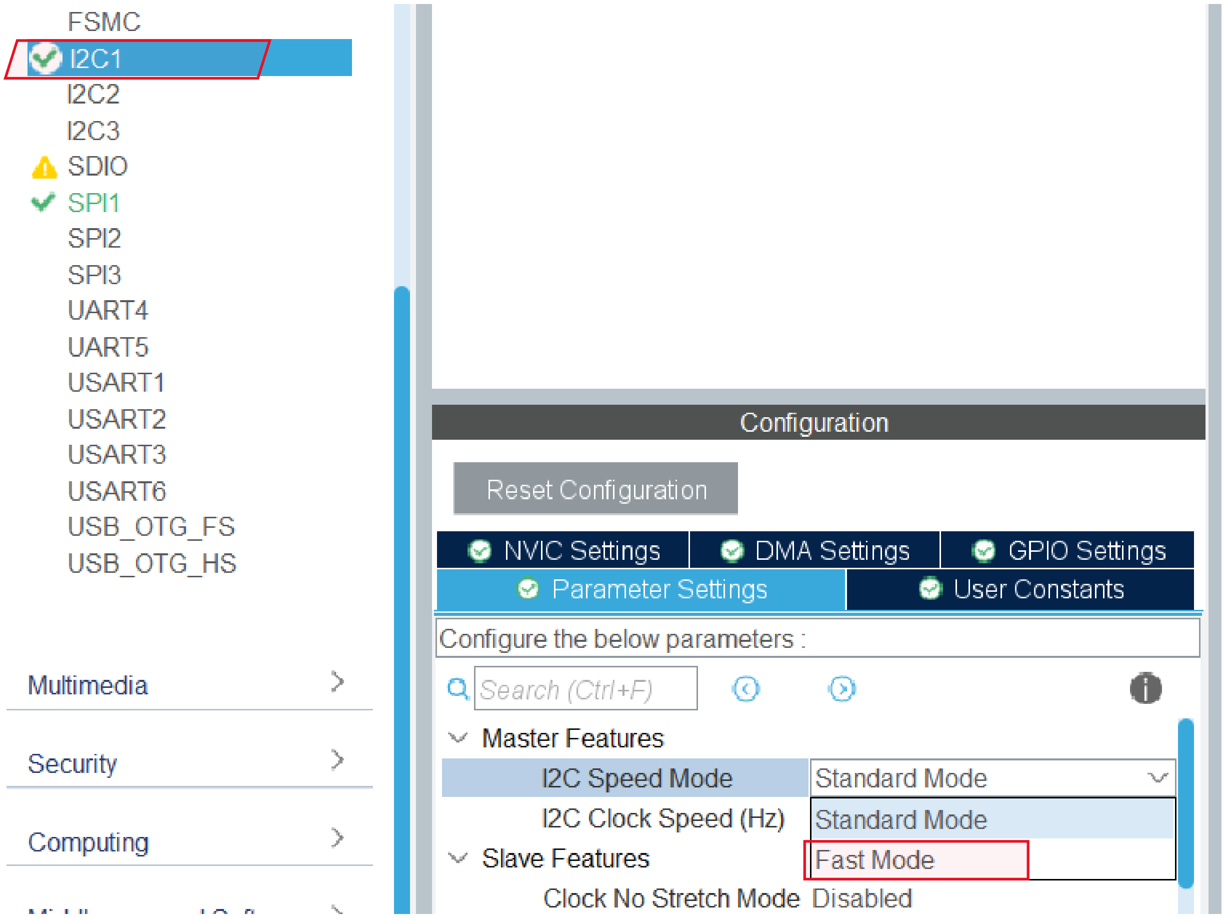


This is the final timer configuration which we use 84MHZ as frequently of our timers



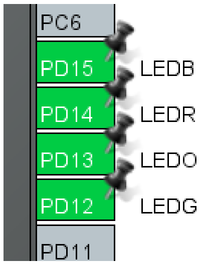
a. Connectivity>SPI1>Mode: Full duplex master, our OLED library contains a reference to SPI library. Also we are going to use it to get the accelerometer values.

Change the I2C1 speed to fast mode ▪



Configure the below parameters :

Master Features
I2C Speed Mode: Standard Mode
I2C Clock Speed (Hz)
Slave Features
Clock No Stretch Mode: Disabled



now you have to make from PD12 to PD15 as the photo LEDG, LEDO, LEDR, LEDB.

Now its time to set our timers we usually change the PSC value and the ARR value,

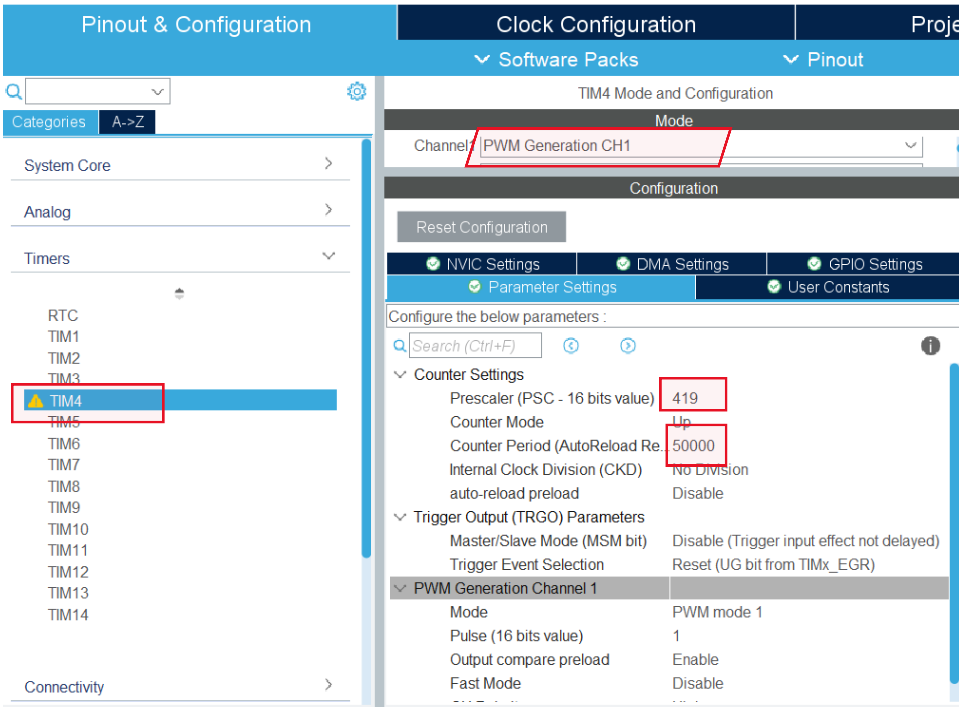
$$f_{\text{PWM}} = \frac{f_{\text{CLK}}}{(\text{PSC}+1)\cdot(\text{ARR}+1)}$$

 Ftm : The frequency of the timer's counting operation.

 Fclk : The clock frequency driving the timer (e.g., APB1/APB2 clock).

PSC : The value of the prescaler (register value)

Here we should make **timer 4** for the trig of our ultrasound and we make it in **channel 1** as **PWM Generation CH1** and the **PSC** should be **419** and the **ARR** should be **50000**.



TIM4 Mode and Configuration

Mode

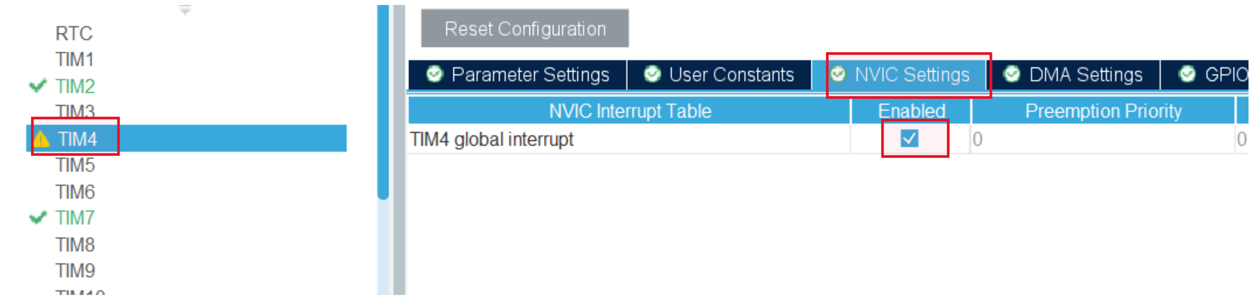
Channel1 PWM Generation CH1

Configuration

Configure the below parameters :

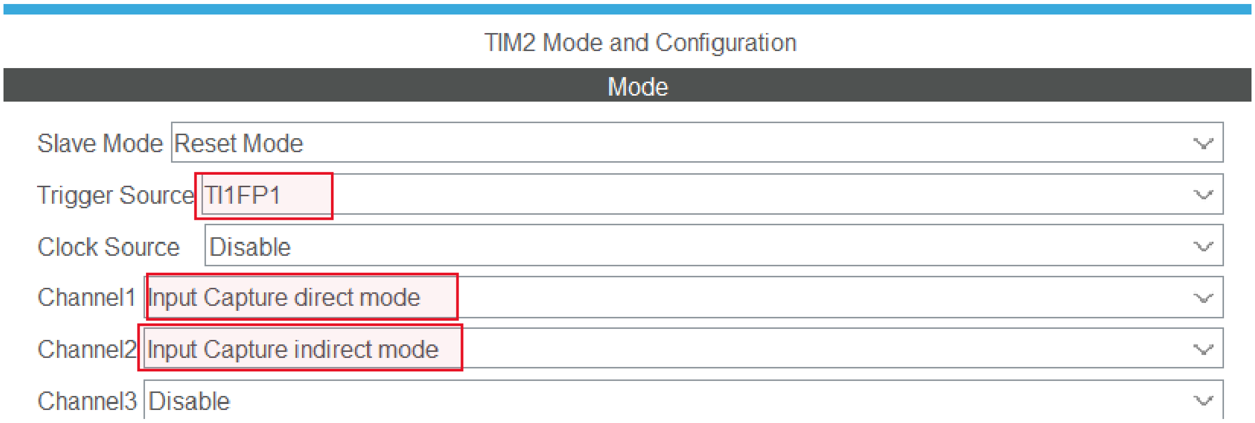
```
Counter Settings
    Prescaler (PSC - 16 bits value)       419
    Counter Mode                          Up
    Counter Period (AutoReload Re...      50000
    Internal Clock Division (CKD)         No Division
    auto-reload preload                   Disable
Trigger Output (TRGO) Parameters
    Master/Slave Mode (MSM bit)           Disable (Trigger input effect not delayed)
    Trigger Event Selection               Reset (UG bit from TIMx_EGR)
PWM Generation Channel 1
    Mode                                  PWM mode 1
    Pulse (16 bits value)                 1
    Output compare preload                Enable
    Fast Mode                             Disable
```

An here we should open the TIM4 global interrupt .



NVIC Interrupt Table	Enabled	Preemption Priority	
TIM4 global interrupt	☑	0	0

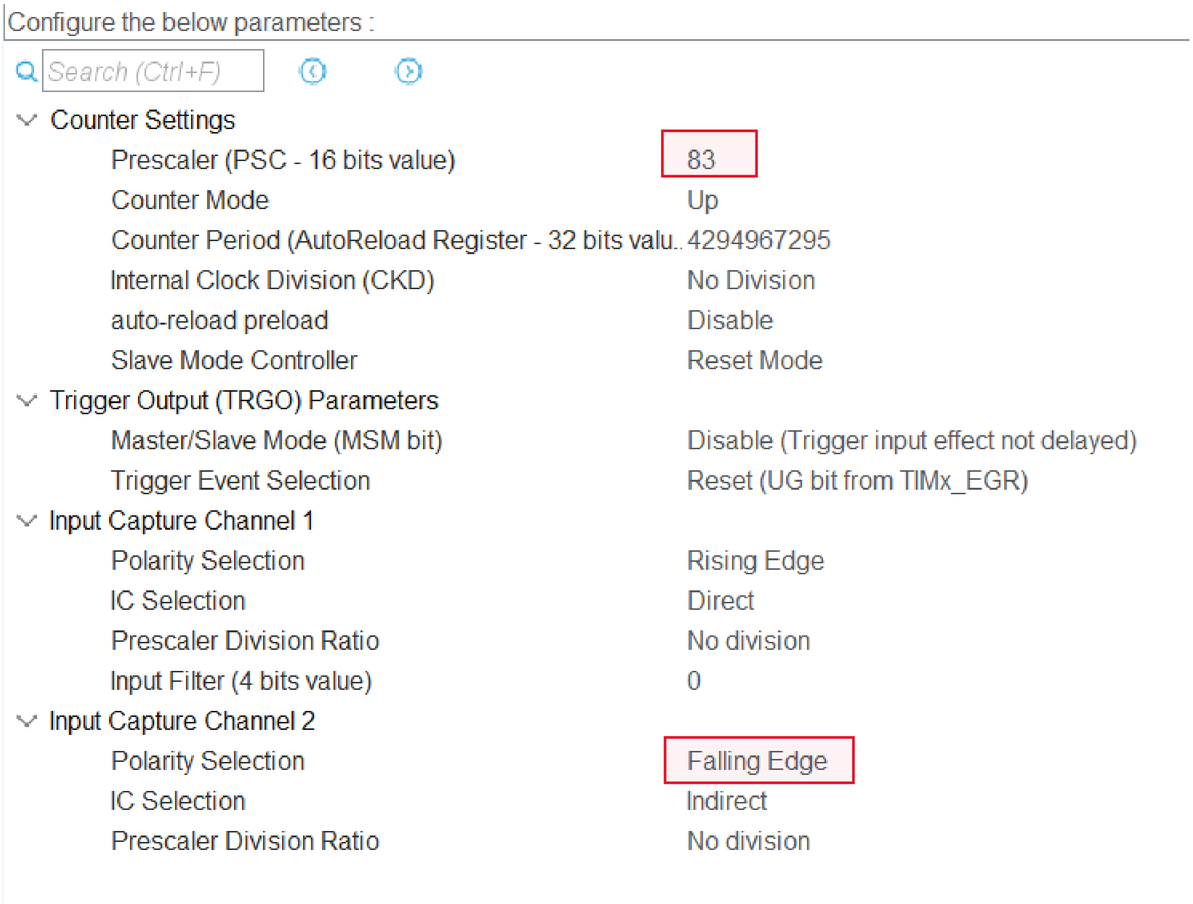
for the echo of the ultrasound we should make some changes .



TIM2 Mode and Configuration

Mode

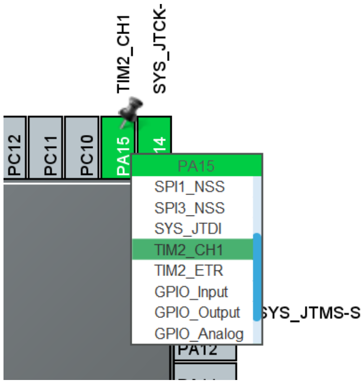
```
Slave Mode      Reset Mode
Trigger Source  TI1FP1
Clock Source    Disable
Channel1        Input Capture direct mode
Channel2        Input Capture indirect mode
Channel3        Disable
```



Configure the below parameters :

Search (Ctrl+F)

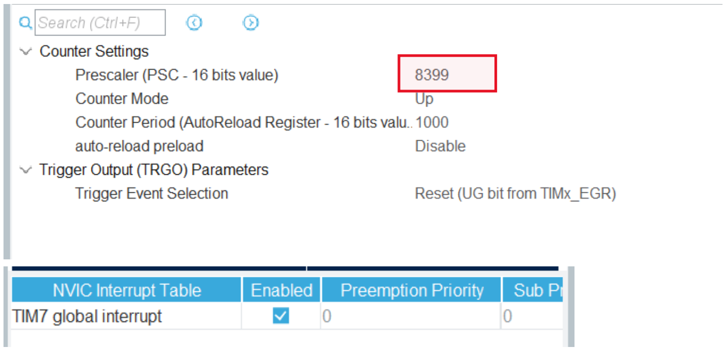
Counter Settings	
Prescaler (PSC - 16 bits value)	83
Counter Mode	Up
Counter Period (AutoReload Register - 32 bits valu..	4294967295
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable
Slave Mode Controller	Reset Mode
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)
Input Capture Channel 1	
Polarity Selection	Rising Edge
IC Selection	Direct
Prescaler Division Ratio	No division
Input Filter (4 bits value)	0
Input Capture Channel 2	
Polarity Selection	Falling Edge
IC Selection	Indirect
Prescaler Division Ratio	No division



For the echo we set timer 4 so it capture the echo and set the time's **PSC** as 83 and we need to set **PA15** as an output of the timer

And now for updating the screen we set **TIMER 7** so it update the screen every 100ms

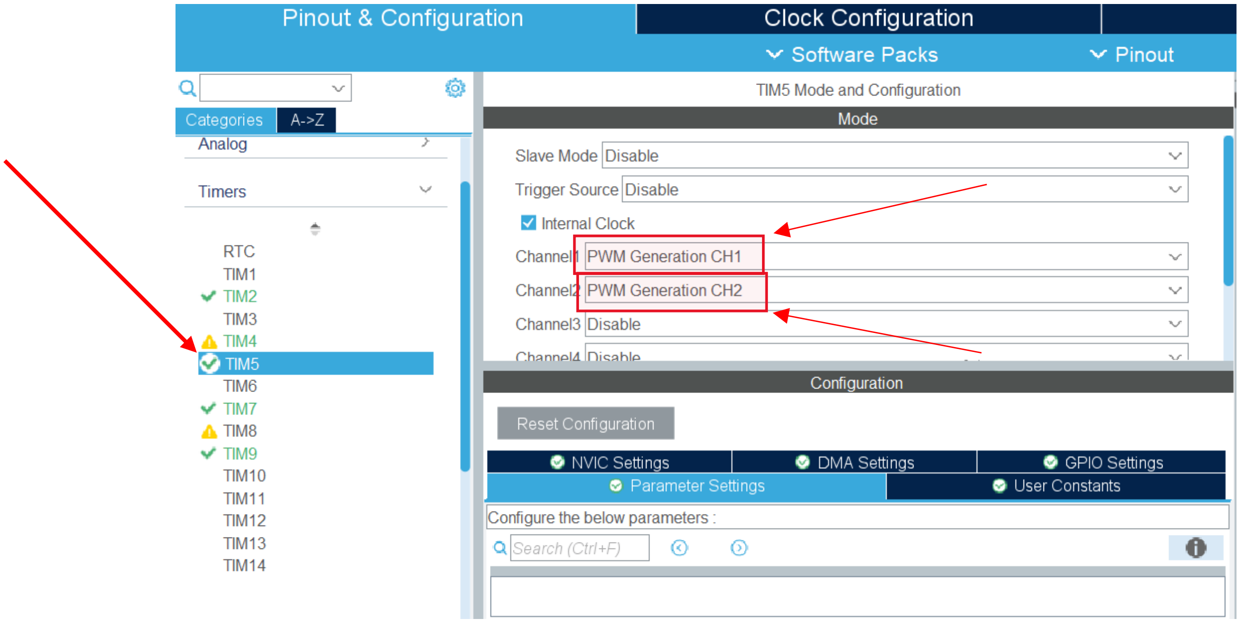
Set the timer7's **PSC** as **89399** and the TIM7 GLOBAL interrupt should be **enabled.**



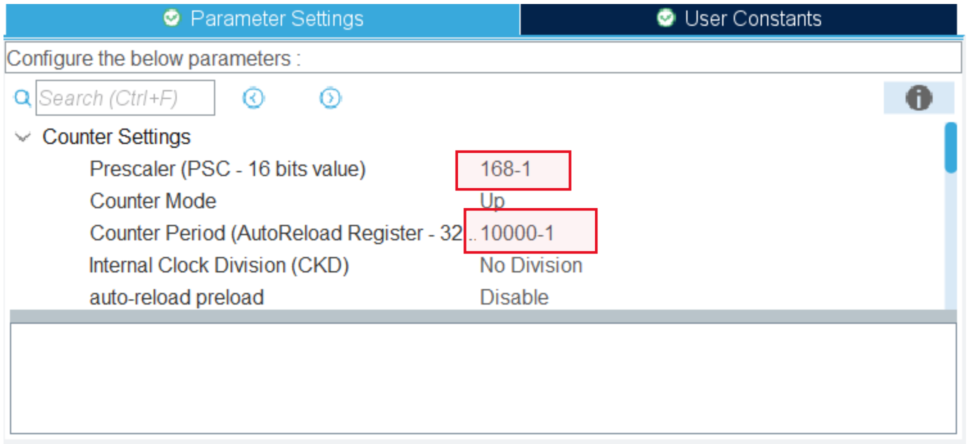
Counter Settings
Prescaler (PSC - 16 bits value) 8399
Counter Mode Up
Counter Period (AutoReload Register - 16 bits valu.. 1000
auto-reload preload Disable
Trigger Output (TRGO) Parameters
Trigger Event Selection Reset (UG bit from TIMx_EGR)

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Pr
TIM7 global interrupt	☑	0	0

now setting timer 5 **channel 2** for the scanning servo , and timer 5 **channel 1** for the laser servo and make the **PSC** 168-1 and the **ARR** as 1000-1.

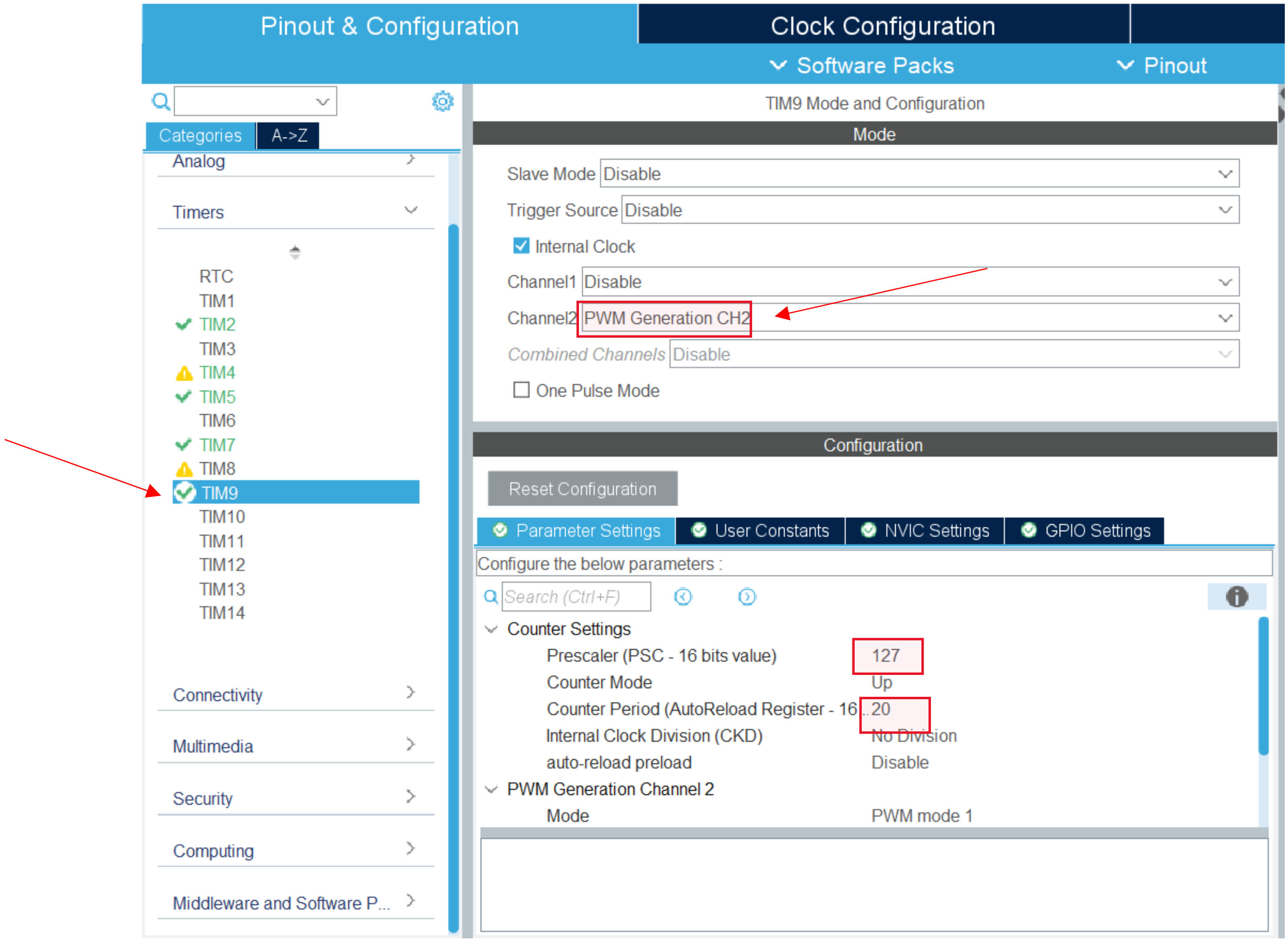


TIM5 Mode and Configuration
Mode
Slave Mode Disable
Trigger Source Disable
☑ Internal Clock
Channel1 PWM Generation CH1
Channel2 PWM Generation CH2
Channel3 Disable
Channel4 Disable
Configuration
Configure the below parameters :



Configure the below parameters :
Counter Settings
Prescaler (PSC - 16 bits value) 168-1
Counter Mode Up
Counter Period (AutoReload Register - 32.. 10000-1
Internal Clock Division (CKD) No Division
auto-reload preload Disable

now to make the laser and the buzzer works we need to make there own timer and if the timer voltage is not enough you can connect the buzzer and the laser with the npn transistor by connecting the timer withe the base of the transistor



TIM9 Mode and Configuration

Mode

Slave Mode Disable

Trigger Source Disable

☑ Internal Clock

Channel1 Disable

Channel2 PWM Generation CH2

Combined Channels Disable

□ One Pulse Mode

Configuration

Configure the below parameters :

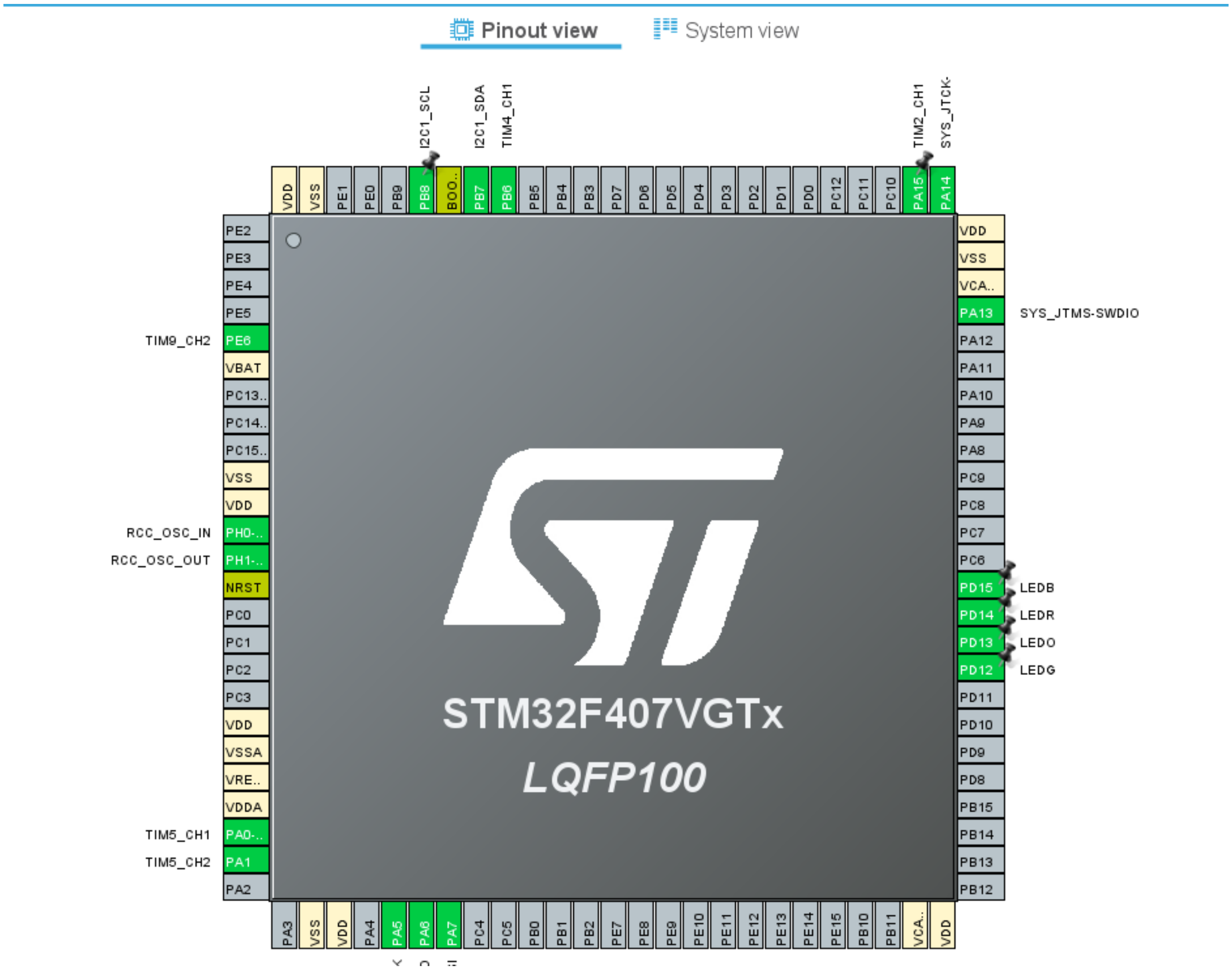
Counter Settings

Prescaler (PSC - 16 bits value)	127
Counter Mode	Up
Counter Period (AutoReload Register - 16...	20
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

PWM Generation Channel 2

Mode	PWM mode 1

The timer will be timer 9 channel 2 by setting the channel 2 as PWM Generation CH2 and makes PSC as 127 and the ARR as 20.



Now make sure the pins is like this and start to connect the **VCC** to 5v and the **GND** to the ground .

for the **LCD** :

The SDA should be connected to PB7 and the SCK to PB8 .

For the **ultrasound :**

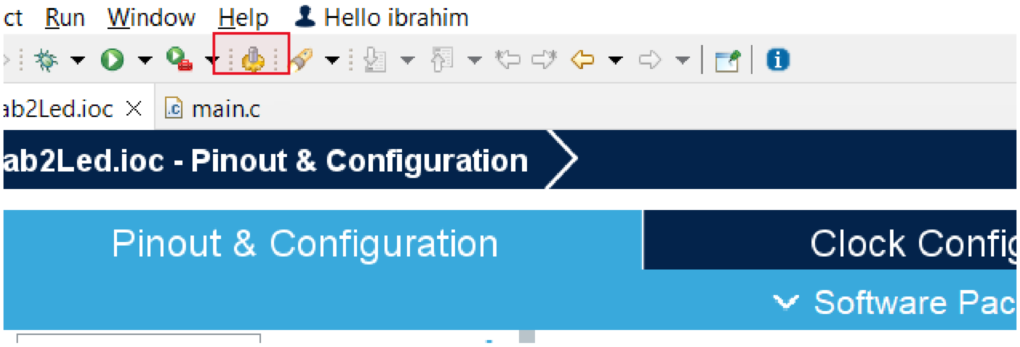
The echo should be connected to PA15 and the trig to PB6.

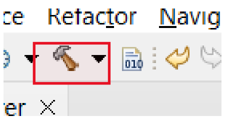
For the **scanning servo** :

connect the orange pin to PA1. For **the laser servo** use PA0.

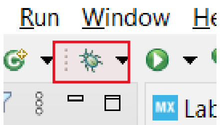
For the **laser** and the **buzzer** connect them to PE6.

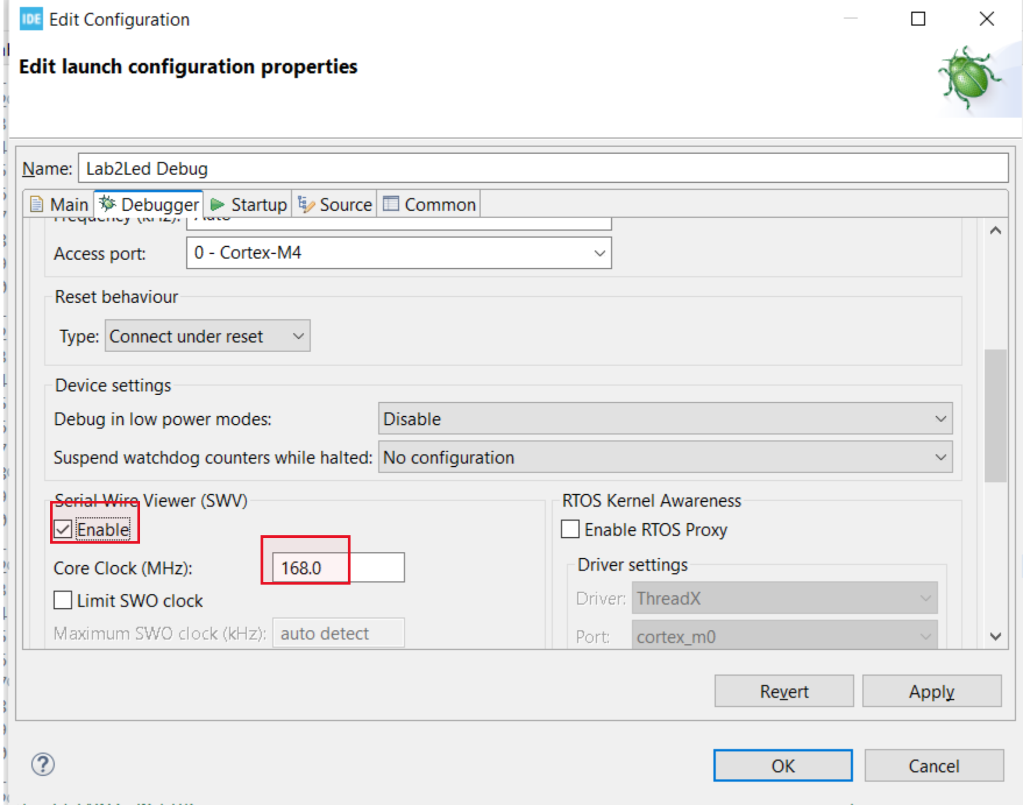
Now to save these changes and generate the main c for the code .





now for the debugging follow this steps .





Edit launch configuration properties

Name: Lab2Led Debug

Access port: 0 - Cortex-M4

Reset behaviour

Type: Connect under reset

Device settings

Debug in low power modes: Disable

Suspend watchdog counters while halted: No configuration

Serial Wire Viewer (SWV)

☑ Enable

Core Clock (MHz): 168.0

□ Limit SWO clock

Maximum SWO clock (kHz): auto detect

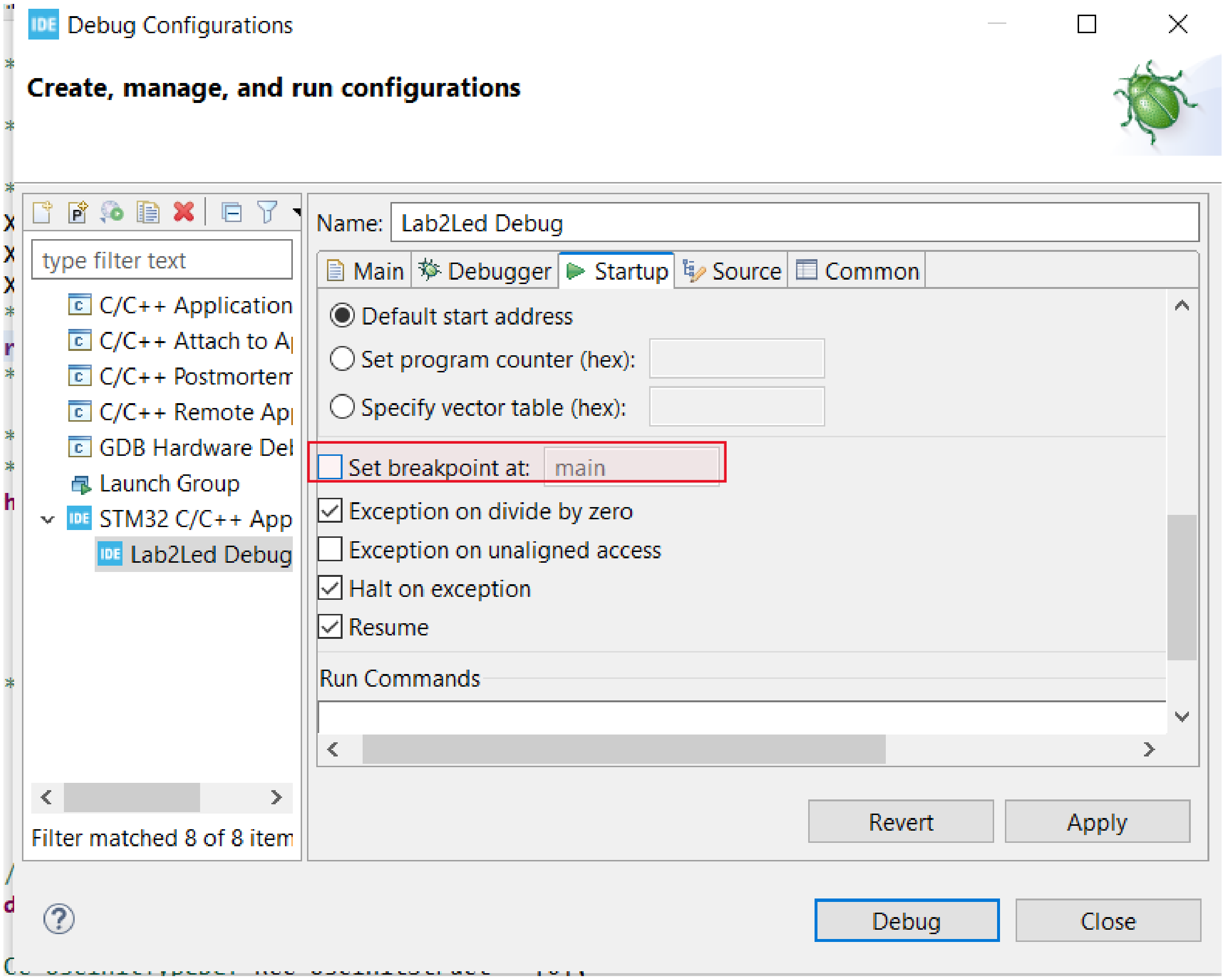
RTOS Kernel Awareness

□ Enable RTOS Proxy

Driver settings

Driver: ThreadX

Port: cortex_m0



Name: Lab2Led Debug

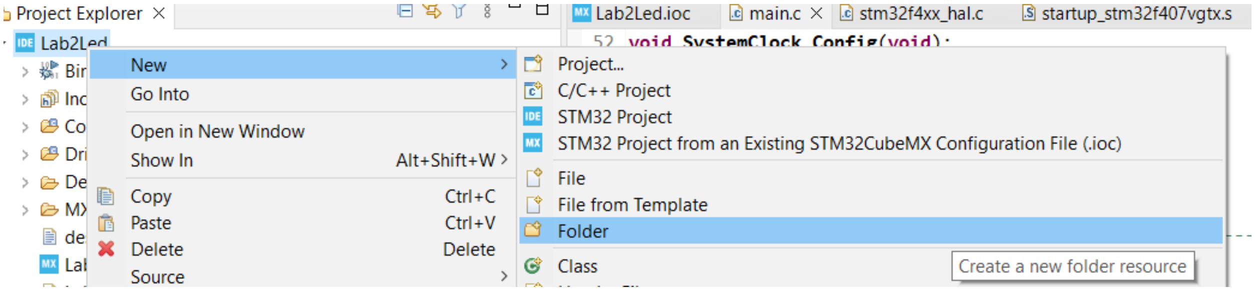
● Default start address
○ Set program counter (hex):
○ Specify vector table (hex):

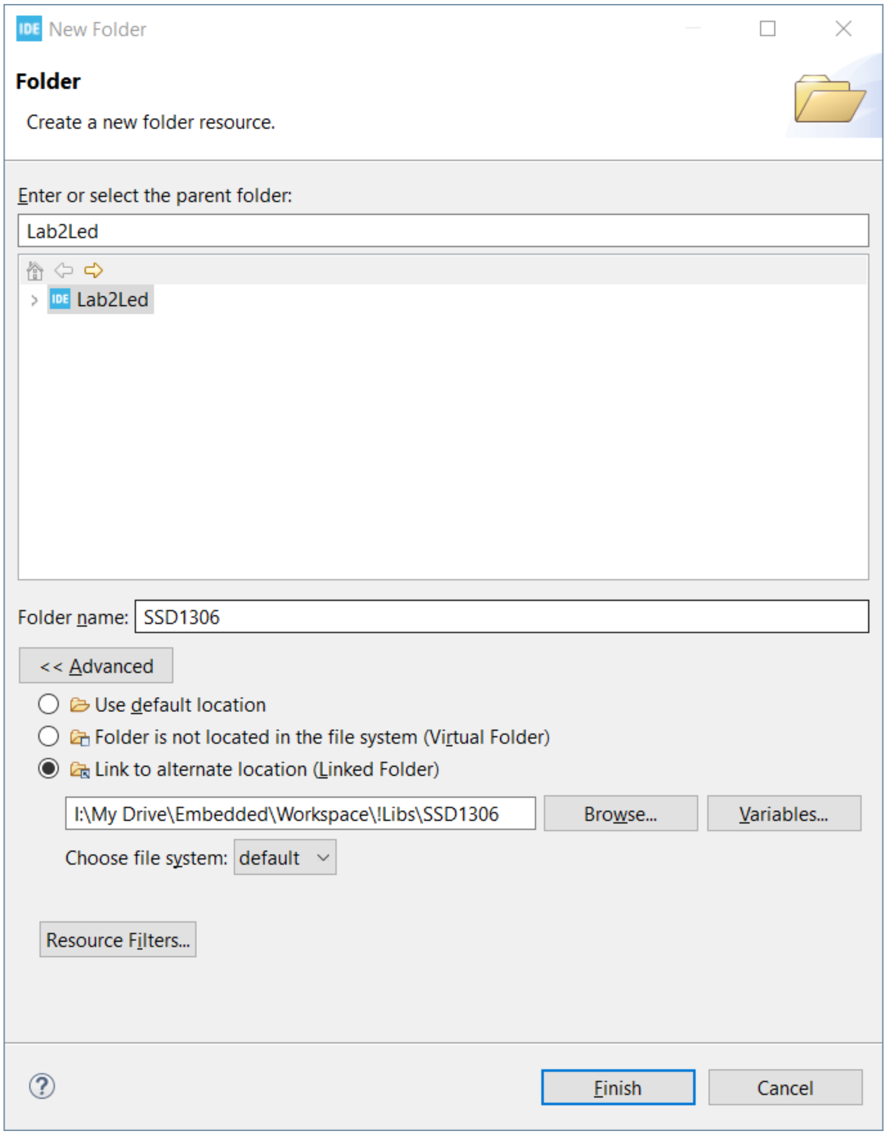
□ Set breakpoint at: main
☑ Exception on divide by zero
□ Exception on unaligned access
☑ Halt on exception
☑ Resume

Run Commands

now all the changes has been saved and the debugging file has been set .

before codding add the library of the lcd so you can use the code inside it .





New Folder

Folder

Create a new folder resource.

Enter or select the parent folder: Lab2Led

Lab2Led

Folder name: SSD1306

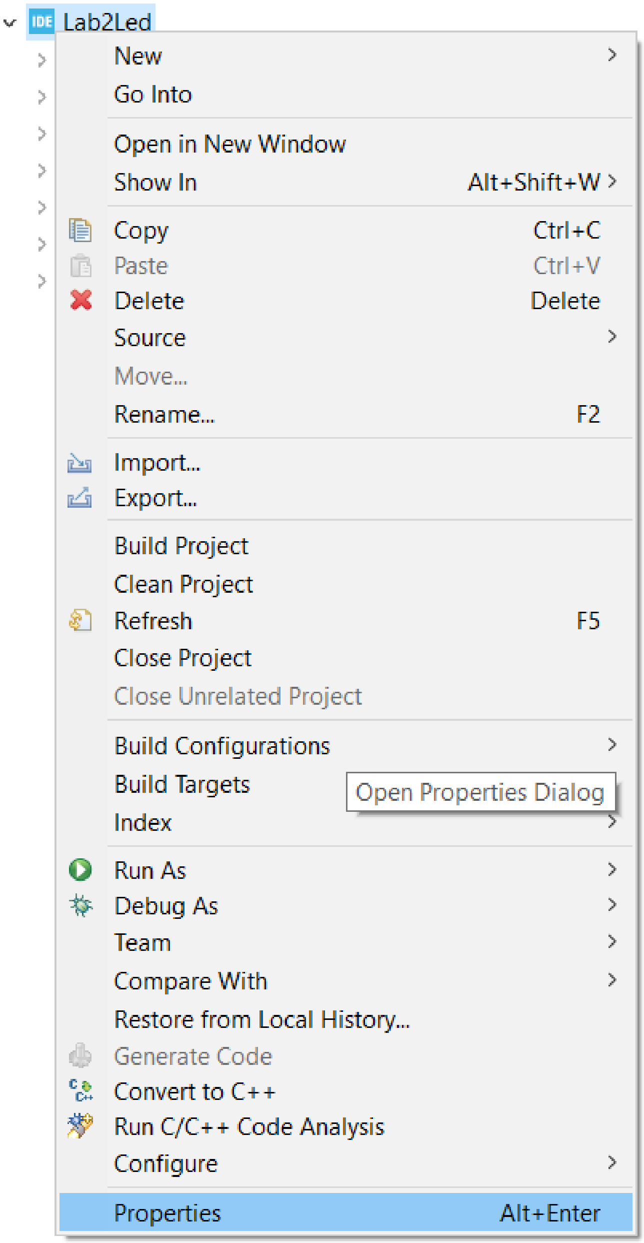
○ Use default location

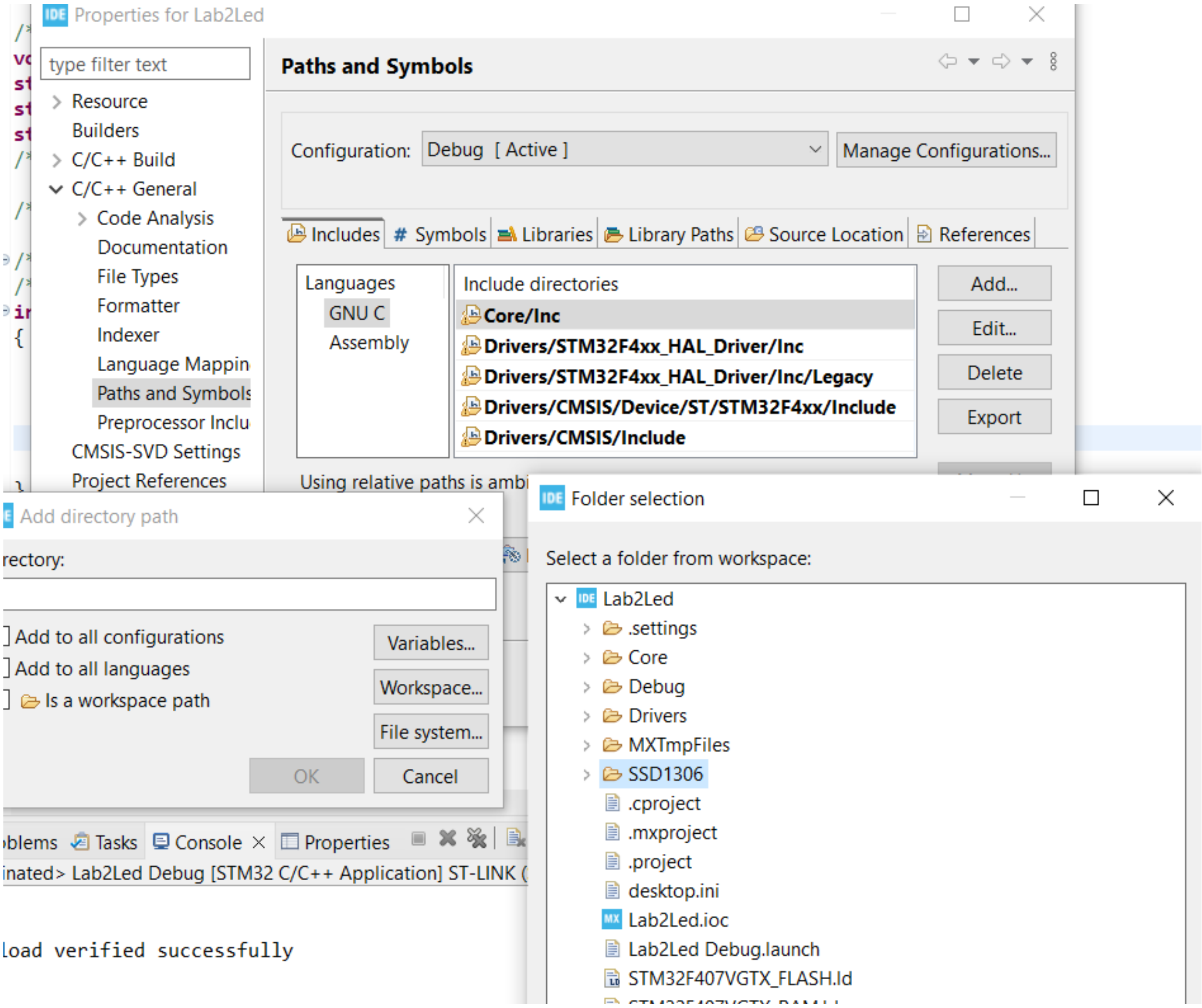
○ Folder is not located in the file system (Virtual Folder)

● Link to alternate location (Linked Folder)

I:\My Drive\Embedded\Workspace\!Libs\SSD1306

Choose file system: default


Lab2Led
New
Go Into
Open in New Window
Show In Alt+Shift+W
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Build Project
Clean Project
Refresh F5
Close Project
Close Unrelated Project
Build Configurations
Build Targets
Open Properties Dialog
Index
Run As
Debug As
Team
Compare With
Restore from Local History...
Generate Code
Convert to C++
Run C/C++ Code Analysis
Configure
Properties Alt+Enter



Paths and Symbols

Configuration: Debug [Active]

Languages	Include directories
GNU C	Core/Inc
Assembly	Drivers/STM32F4xx_HAL_Driver/Inc
	Drivers/STM32F4xx_HAL_Driver/Inc/Legacy
	Drivers/CMSIS/Device/ST/STM32F4xx/Include
	Drivers/CMSIS/Include

Using relative paths is ambi

Add directory path

rectory:

] Add to all configurations

] Add to all languages

] Is a workspace path

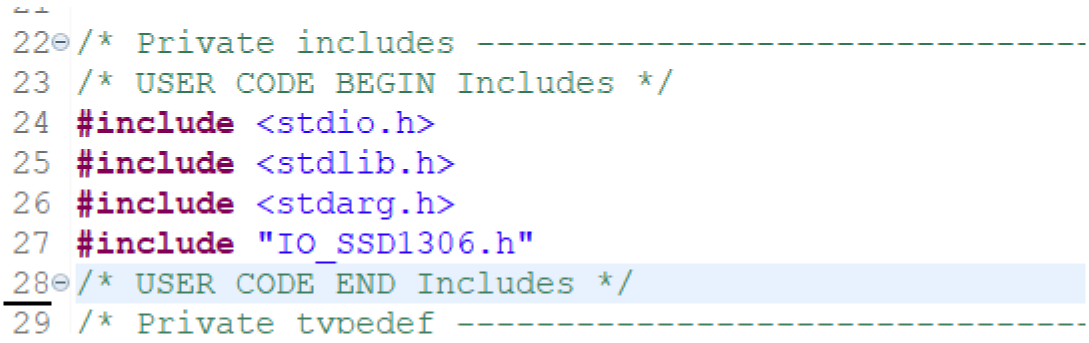
Select a folder from workspace:

```
Lab2Led
  .settings
  Core
  Debug
  Drivers
  MXTmpFiles
  SSD1306
  .cproject
  .mxproject
  .project
  desktop.ini
  Lab2Led.ioc
  Lab2Led Debug.launch
  STM32F407VGTX_FLASH.ld
```

```
inated> Lab2Led Debug [STM32 C/C++ Application] ST-LINK (

load verified successfully
```

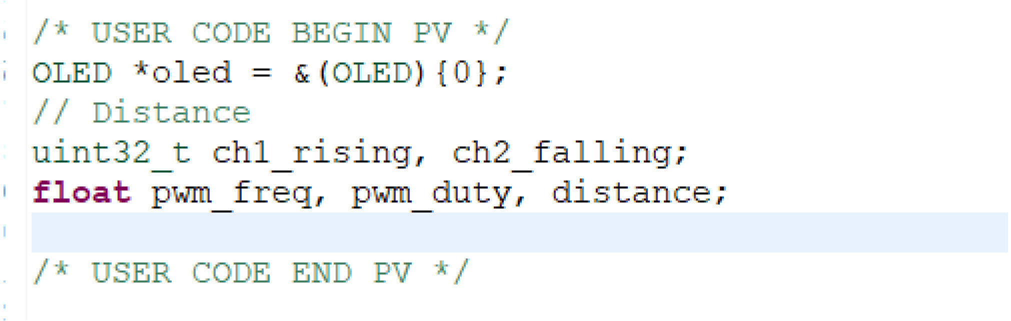
The library of the lcd is ready to be used now go to the main c.

Main.c



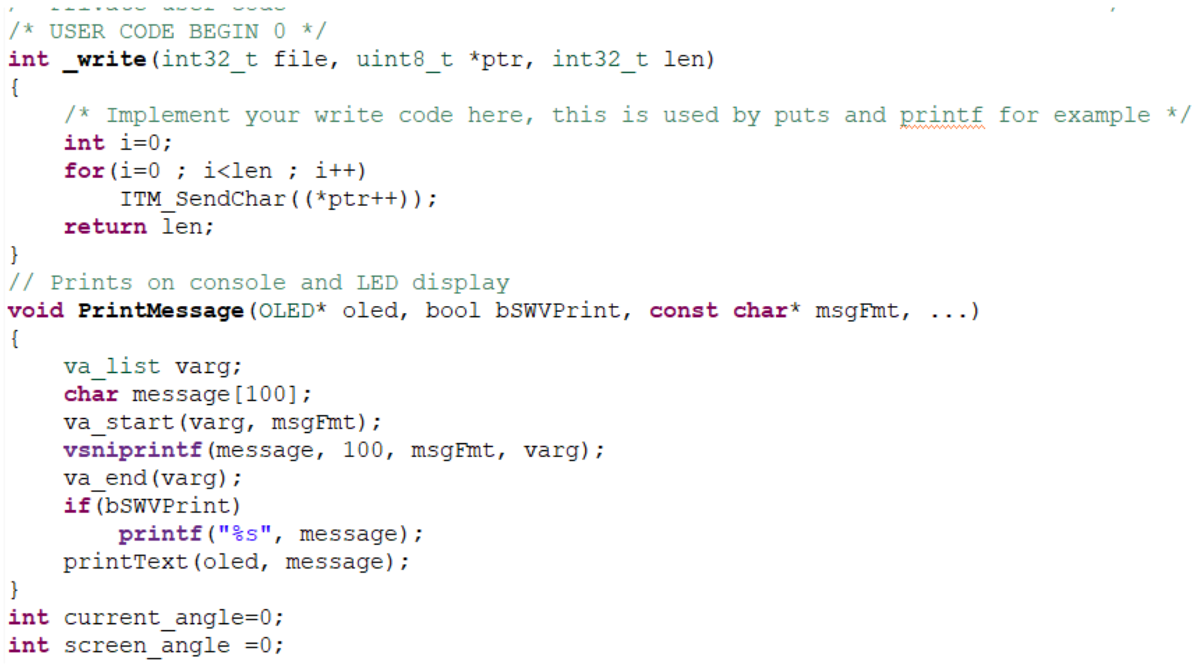
```
/* Private includes ----------------------------------
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "IO_SSD1306.h"
/* USER CODE END Includes */
/* Private typedef -----------------------------------
```

This is the includes that is needed for this project .



```
/* USER CODE BEGIN PV */
OLED *oled = &(OLED){0};
// Distance
uint32_t ch1_rising, ch2_falling;
float pwm_freq, pwm_duty, distance;

/* USER CODE END PV */
```

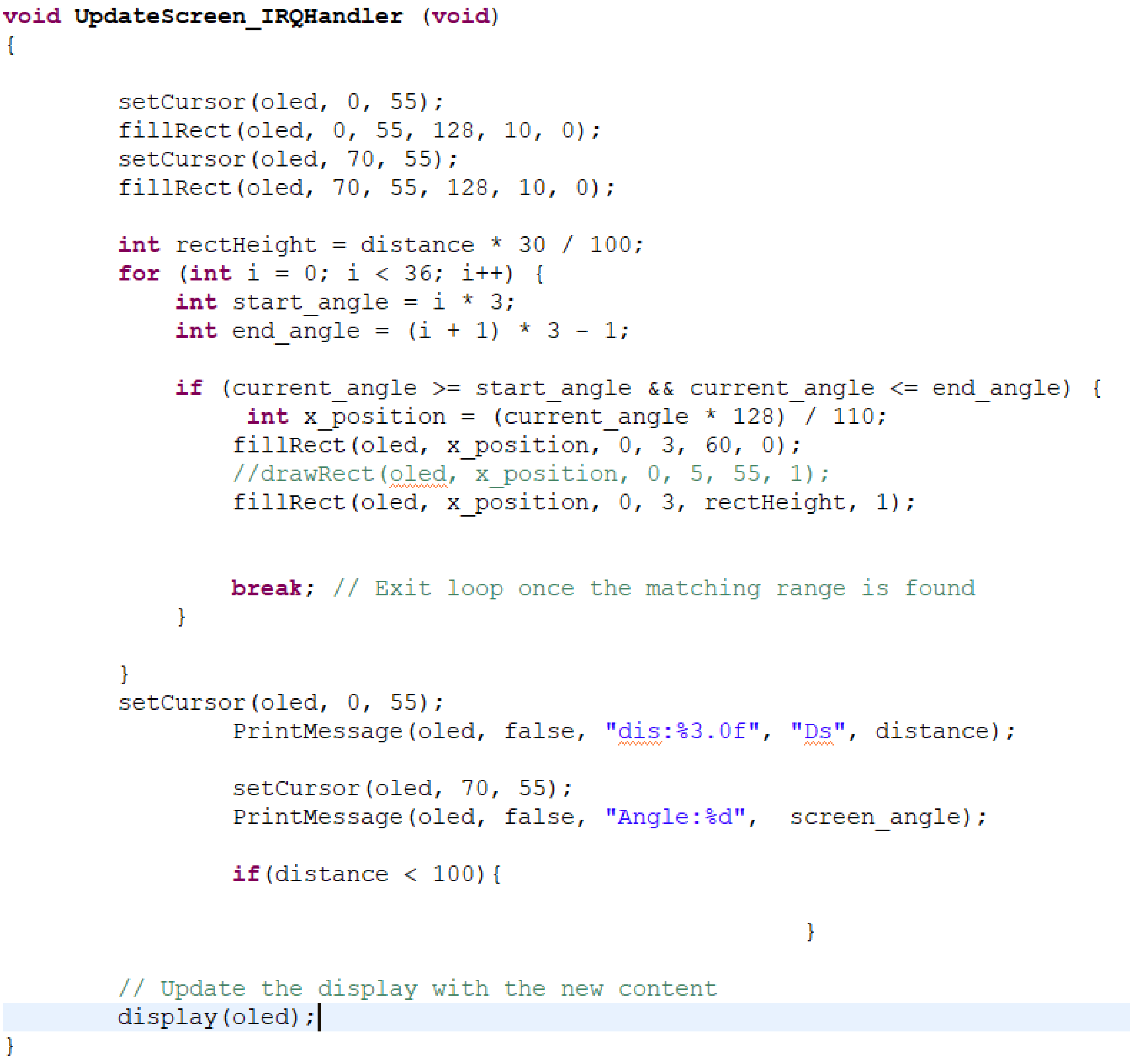
and adding this private values is these for the lcd and the ultrasound .



```
/* USER CODE BEGIN 0 */
int _write(int32_t file, uint8_t *ptr, int32_t len)
{
    /* Implement your write code here, this is used by puts and printf for example */
    int i=0;
    for(i=0 ; i<len ; i++)
        ITM_SendChar((*ptr++));
    return len;
}
// Prints on console and LED display
void PrintMessage(OLED* oled, bool bSWVPrint, const char* msgFmt, ...)
{
    va_list varg;
    char message[100];
    va_start(varg, msgFmt);
    vsniprintf(message, 100, msgFmt, varg);
    va_end(varg);
    if(bSWVPrint)
        printf("%s", message);
    printText(oled, message);
}
int current_angle=0;
int screen_angle =0;
```

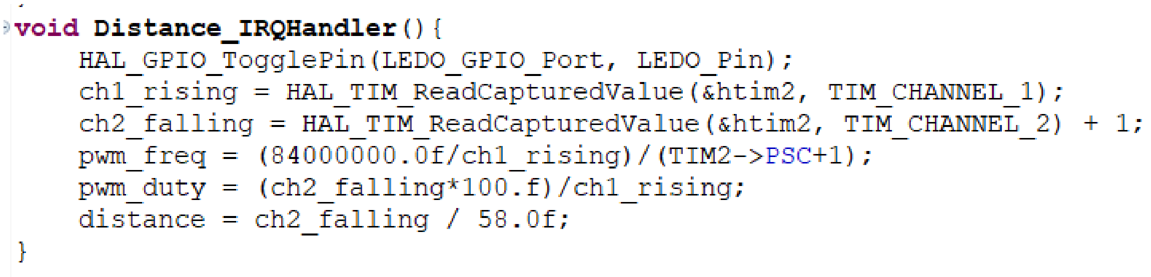
In the USER CODE BEGIN add these functions one for the lcd to write and the other to print message .

Still in the USER BEGING CODE add these functions.



```
void UpdateScreen_IRQHandler (void)
{

        setCursor(oled, 0, 55);
        fillRect(oled, 0, 55, 128, 10, 0);
        setCursor(oled, 70, 55);
        fillRect(oled, 70, 55, 128, 10, 0);

        int rectHeight = distance * 30 / 100;
        for (int i = 0; i < 36; i++) {
            int start_angle = i * 3;
            int end_angle = (i + 1) * 3 - 1;

            if (current_angle >= start_angle && current_angle <= end_angle) {
                 int x_position = (current_angle * 128) / 110;
                fillRect(oled, x_position, 0, 3, 60, 0);
                //drawRect(oled, x_position, 0, 5, 55, 1);
                fillRect(oled, x_position, 0, 3, rectHeight, 1);


                break; // Exit loop once the matching range is found
            }

        }
        setCursor(oled, 0, 55);
                PrintMessage(oled, false, "dis:%3.0f", "Ds", distance);

                setCursor(oled, 70, 55);
                PrintMessage(oled, false, "Angle:%d",  screen_angle);

                if(distance < 100){

                                                        }

        // Update the display with the new content
        display(oled);
}
```

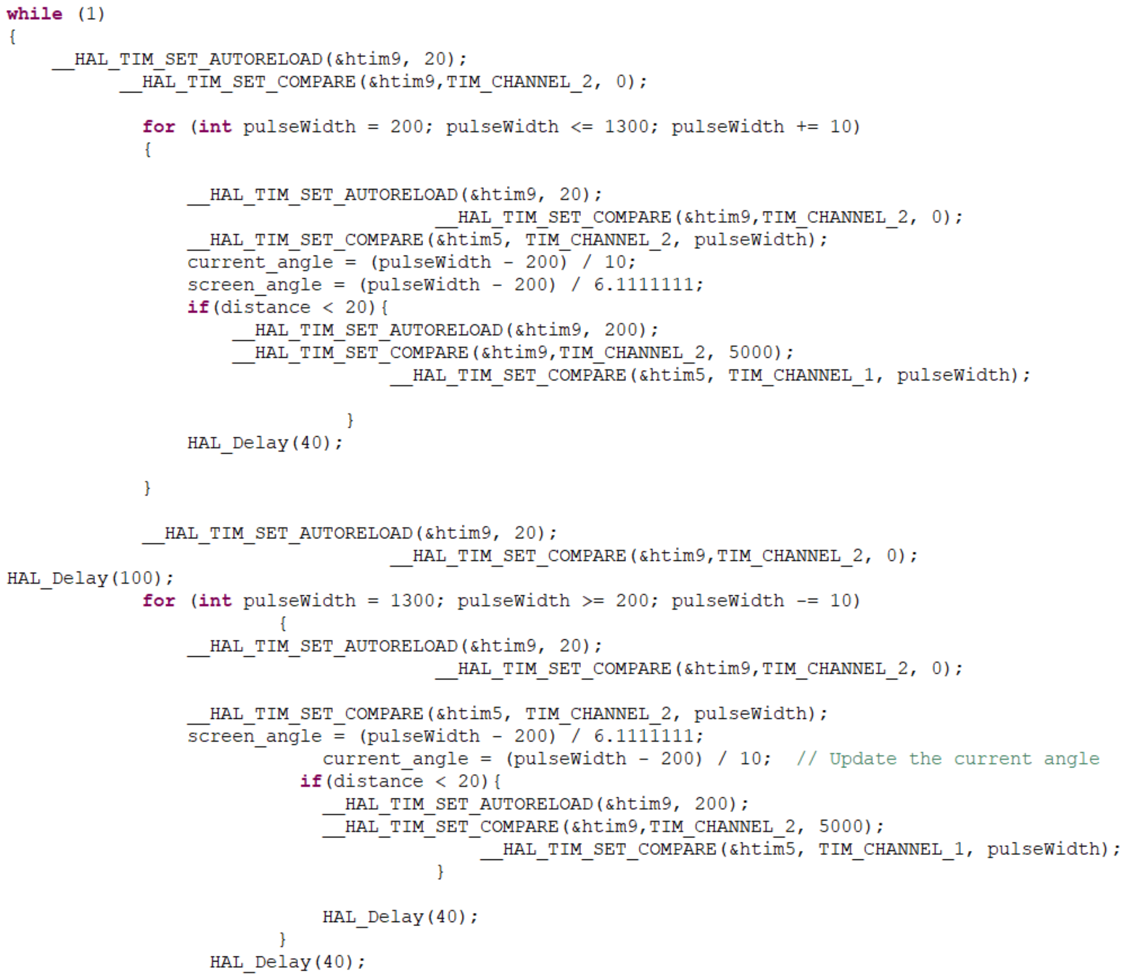
this one is for updating the screen and it will be connected to timer7 .



```
void Distance_IRQHandler(){
    HAL_GPIO_TogglePin(LEDO_GPIO_Port, LEDO_Pin);
    ch1_rising = HAL_TIM_ReadCapturedValue(&htim2, TIM_CHANNEL_1);
    ch2_falling = HAL_TIM_ReadCapturedValue(&htim2, TIM_CHANNEL_2) + 1;
    pwm_freq = (84000000.0f/ch1_rising)/(TIM2->PSC+1);
    pwm_duty = (ch2_falling*100.f)/ch1_rising;
    distance = ch2_falling / 58.0f;
}
```

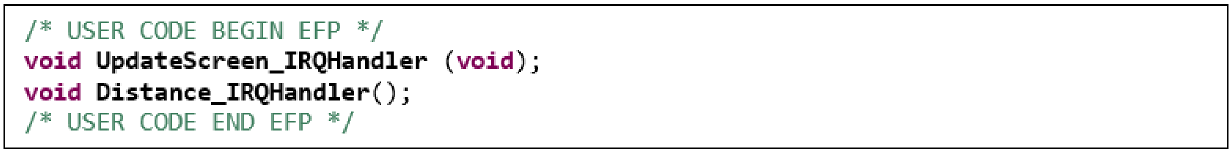
In the main USER BEGINNING CODE you should add these code for the ultrasound and the screen to be reading and to name the timers .

Inside the while loop addd the servo code loop .

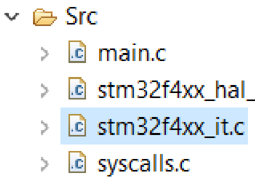


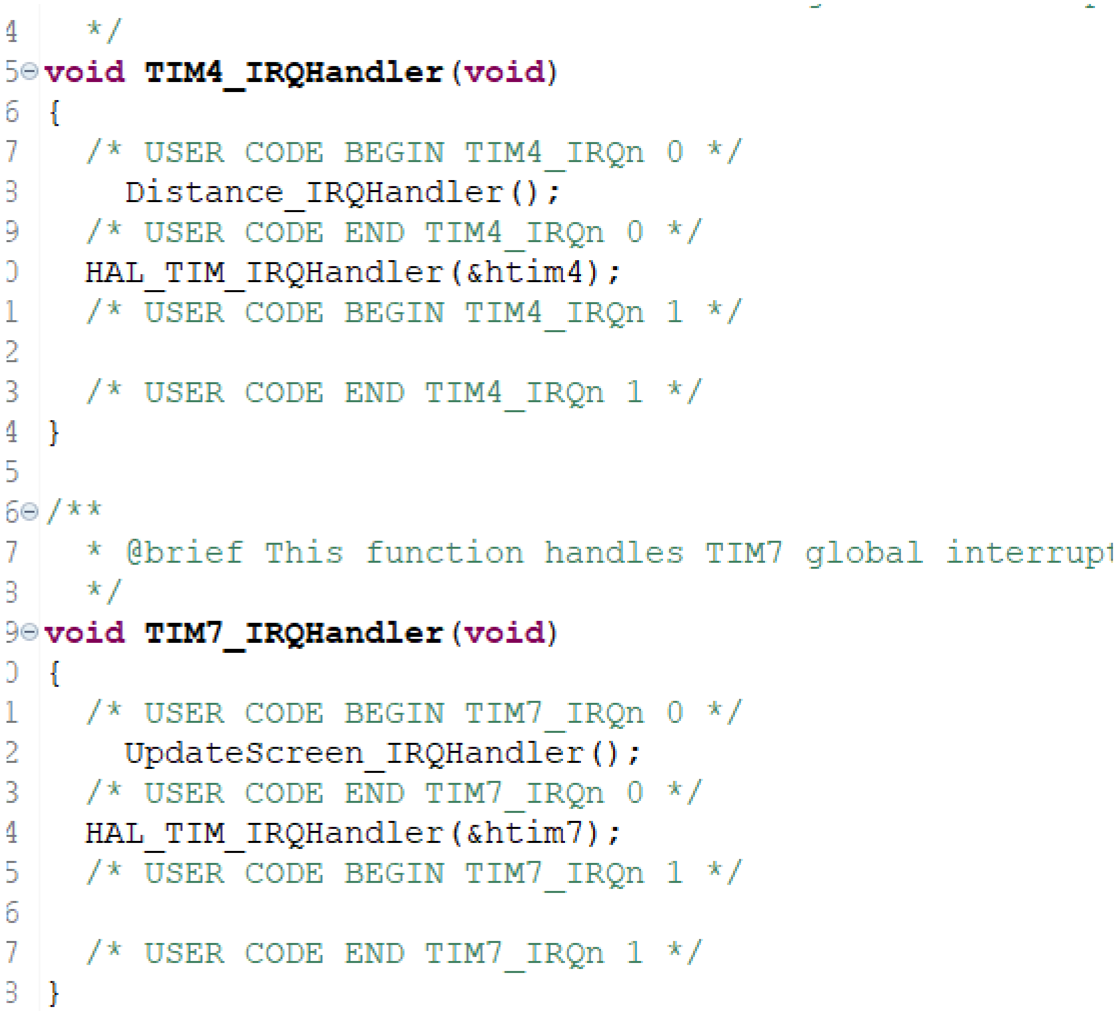
```
while (1)
{
    __HAL_TIM_SET_AUTORELOAD(&htim9, 20);
          __HAL_TIM_SET_COMPARE(&htim9,TIM_CHANNEL_2, 0);

            for (int pulseWidth = 200; pulseWidth <= 1300; pulseWidth += 10)
            {

                __HAL_TIM_SET_AUTORELOAD(&htim9, 20);
                                      __HAL_TIM_SET_COMPARE(&htim9,TIM_CHANNEL_2, 0);
                __HAL_TIM_SET_COMPARE(&htim5, TIM_CHANNEL_2, pulseWidth);
                current_angle = (pulseWidth - 200) / 10;
                screen_angle = (pulseWidth - 200) / 6.1111111;
                if(distance < 20){
                    __HAL_TIM_SET_AUTORELOAD(&htim9, 200);
                    __HAL_TIM_SET_COMPARE(&htim9,TIM_CHANNEL_2, 5000);
                                  __HAL_TIM_SET_COMPARE(&htim5, TIM_CHANNEL_1, pulseWidth);

                              }
                HAL_Delay(40);

            }

            __HAL_TIM_SET_AUTORELOAD(&htim9, 20);
                                  __HAL_TIM_SET_COMPARE(&htim9,TIM_CHANNEL_2, 0);
HAL_Delay(100);
            for (int pulseWidth = 1300; pulseWidth >= 200; pulseWidth -= 10)
                        {
                __HAL_TIM_SET_AUTORELOAD(&htim9, 20);
                                      __HAL_TIM_SET_COMPARE(&htim9,TIM_CHANNEL_2, 0);

                __HAL_TIM_SET_COMPARE(&htim5, TIM_CHANNEL_2, pulseWidth);
                screen_angle = (pulseWidth - 200) / 6.1111111;
                            current_angle = (pulseWidth - 200) / 10;  // Update the current angle
                          if(distance < 20){
                            __HAL_TIM_SET_AUTORELOAD(&htim9, 200);
                            __HAL_TIM_SET_COMPARE(&htim9,TIM_CHANNEL_2, 5000);
                                          __HAL_TIM_SET_COMPARE(&htim5, TIM_CHANNEL_1, pulseWidth);
                                      }

                            HAL_Delay(40);
                        }
                  HAL_Delay(40);
```

Main.h



```
/* USER CODE BEGIN EFP */
void UpdateScreen_IRQHandler (void);
void Distance_IRQHandler();
/* USER CODE END EFP */
```





```
  */
void TIM4_IRQHandler(void)
{
  /* USER CODE BEGIN TIM4_IRQn 0 */
    Distance_IRQHandler();
  /* USER CODE END TIM4_IRQn 0 */
  HAL_TIM_IRQHandler(&htim4);
  /* USER CODE BEGIN TIM4_IRQn 1 */

  /* USER CODE END TIM4_IRQn 1 */
}

/**
  * @brief This function handles TIM7 global interrupt
  */
void TIM7_IRQHandler(void)
{
  /* USER CODE BEGIN TIM7_IRQn 0 */
    UpdateScreen_IRQHandler();
  /* USER CODE END TIM7_IRQn 0 */
  HAL_TIM_IRQHandler(&htim7);
  /* USER CODE BEGIN TIM7_IRQn 1 */

  /* USER CODE END TIM7_IRQn 1 */
}
```