EE242-MICROPROCESSOR SYSTEMS

Project Report
Implemention of Alarm Clock by Using Stm32F103

by

Mustafa AKSOY

20160701106

YEDİTEPE ÜNİVERSİTESİ

Faculty of Engineering

Department of Electrical and Electronics Engineering

2020 Fall

1 .PURPOSE OF THE PROJECT

The aim of this project is to make an alarm clock with a stm32f407vg microprocessor. The stm32f407vg software language is C. An LCD screen is used to display the time. The buttons are used to adjust the time. Buzzer is used to make a sound when the time comes.

2. METHODOLOGY

2.1 Harware System

In this project, stm32f407vg microprocessor, 16x2 LCD Display, buzzer and button are used.

2.1.1 STM32F103 Microprocessor

The STM32F103xx medium-density performance line family incorporates the high-performance ARM®Cortex®-M3 32-bit RISC core operating at a 72 MHz frequency, high-speed embedded memories (Flash memory up to 128 Kbytes and SRAM up to 20 Kbytes), and an extensive range of enhanced I/Os and peripherals connected to two APB buses. All devices offer two 12-bit ADCs, three general purpose 16-bit timers plus one PWM timer, as well as standard and advanced communication interfaces: up to two I^{2}Cs and SPIs, three USARTs, an USB and a CAN.

Figure 2. 1: STM32F103

2.1.2 16x2 LCD Screen

It has white writing color on a blue background. It works with 5V. It does not need any other power source. Compatible with development boards. There are 16 pins on it. The number of pins can be reduced by using a converter. This project is also used to show the time and the alarm**.**

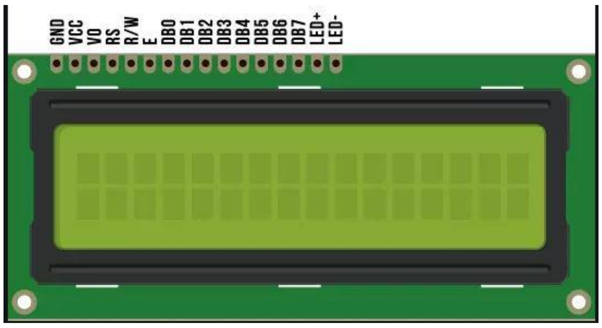


Figure 2. 2: 16x2 LCD Screen

2.1.3 Buzzer

Buzzer; It is a type of auditory warning device that works based on mechanical, electromechanical or piezoelectric principles. Buzzers, which have a lot of usage areas, generally work with the piezoelectric principle. Buzzers can be used in functions such as alarm, timer, confirmation response alert, depending on their area of use. As a matter of fact, as mentioned in the definition, buzzers are types of auditory warning devices. They have types such as illuminated buzzer, non-light buzzer, passive buzzer and active buzzer. Buzzer devices have two legs in total. One of the legs is (+) and the other is (-) leg. Therefore, attention should be paid to the legs of the buzzer when connecting the buzzer. Buzzers usually have (+) or (-) indicators on the legs to help you distinguish their legs. In this project, buzzer was used to wake up.

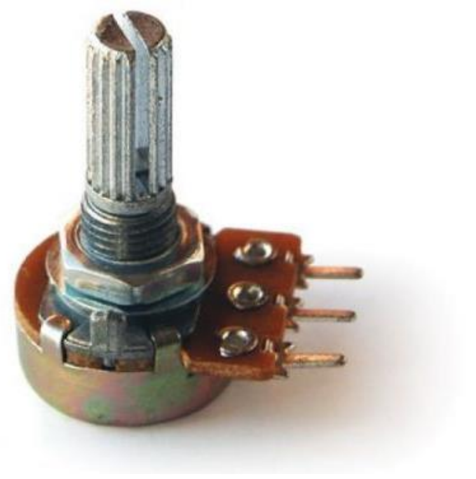
Figure 2.3 : Buzzer

2.1.4 Button

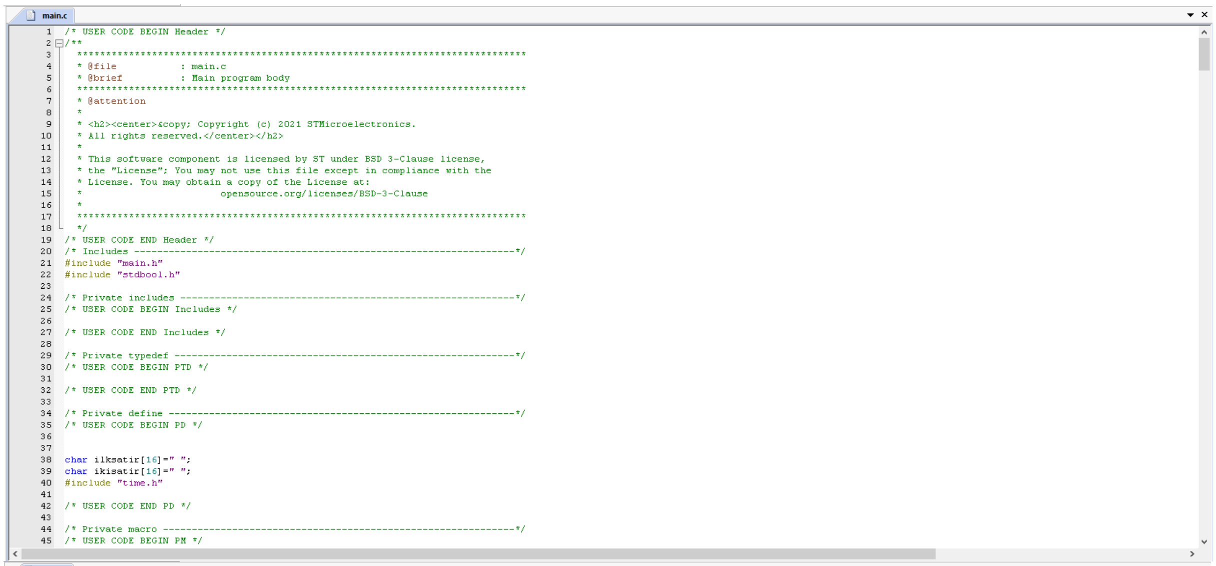
It is available with 4 legs or 2 legs. It is powered by 5V. It allows us to keep it constant while push up and pull down logic is at 0. In this project, the hour, minute, second changes as the button is pressed. That's why 3 were used.

Figure 2. 4 : Button

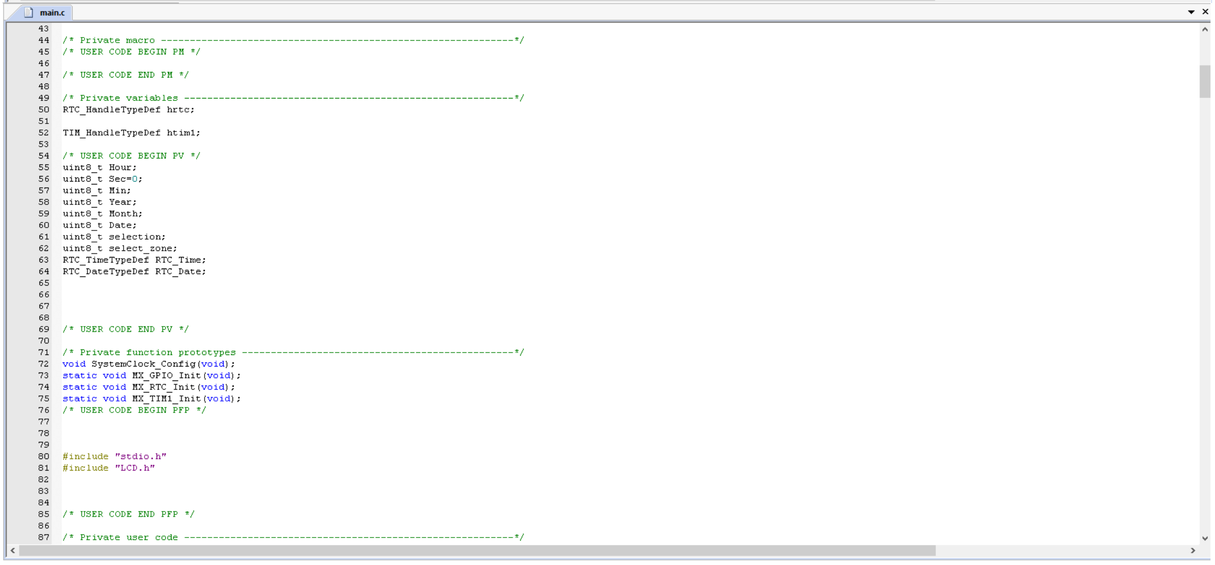
2.1.5 Potansiyometre



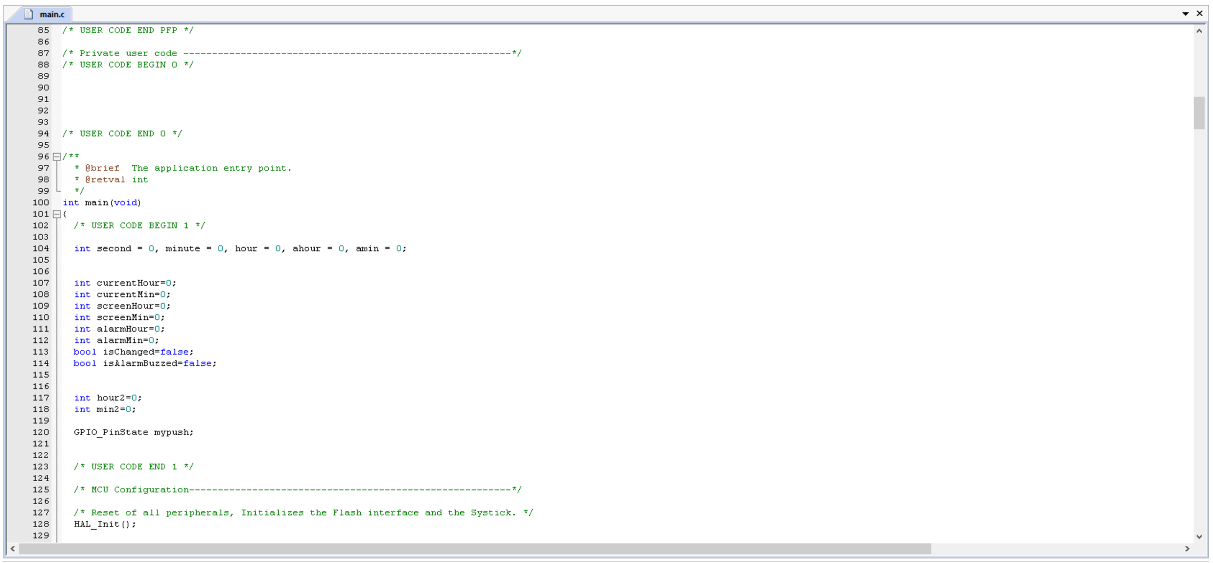
2.2 Code



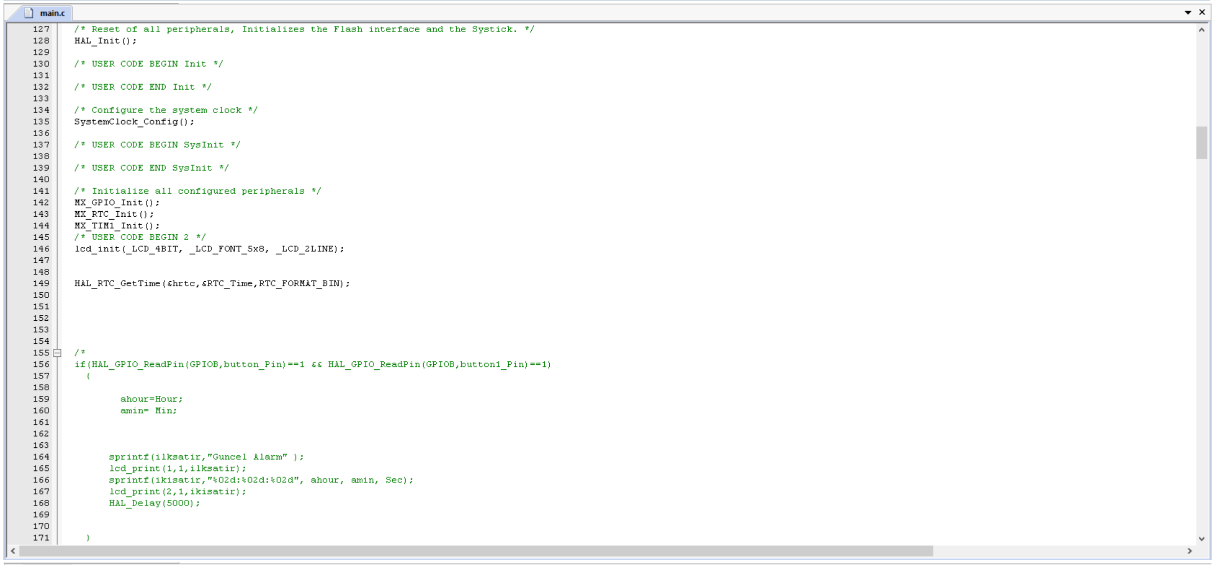
```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stdbool.h"

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

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */


char ilksatir[16]=" ";
char ikisatir[16]=" ";
#include "time.h"

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
```



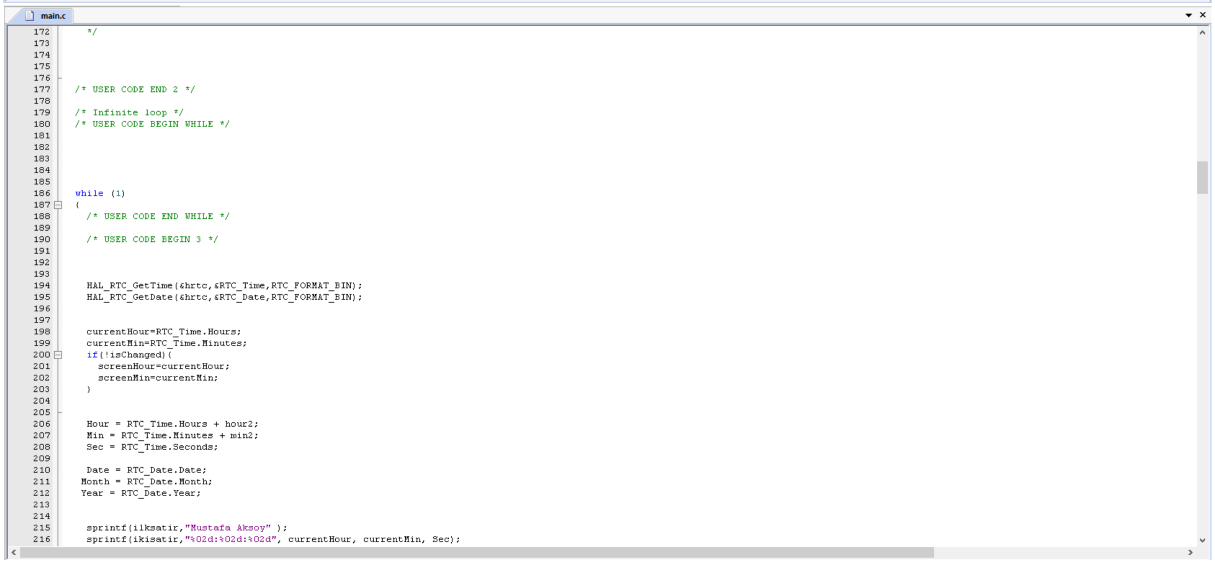
```c

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
RTC_HandleTypeDef hrtc;

TIM_HandleTypeDef htim1;

/* USER CODE BEGIN PV */
uint8_t Hour;
uint8_t Sec=0;
uint8_t Min;
uint8_t Year;
uint8_t Month;
uint8_t Date;
uint8_t selection;
uint8_t select_zone;
RTC_TimeTypeDef RTC_Time;
RTC_DateTypeDef RTC_Date;




/* USER CODE END PV */

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



#include "stdio.h"
#include "LCD.h"



/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
```



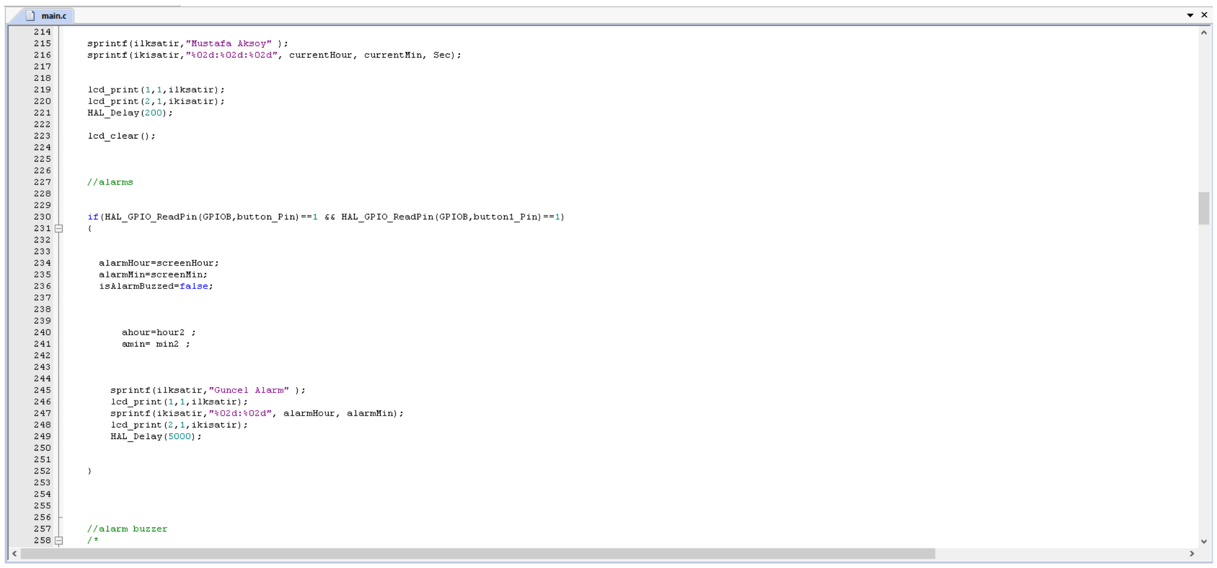
```c
/* USER CODE END PFP */

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





/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  int second = 0, minute = 0, hour = 0, ahour = 0, amin = 0;


  int currentHour=0;
  int currentMin=0;
  int screenHour=0;
  int screenMin=0;
  int alarmHour=0;
  int alarmMin=0;
  bool isChanged=false;
  bool isAlarmBuzzed=false;


  int hour2=0;
  int min2=0;

  GPIO_PinState mypush;


  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

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

```



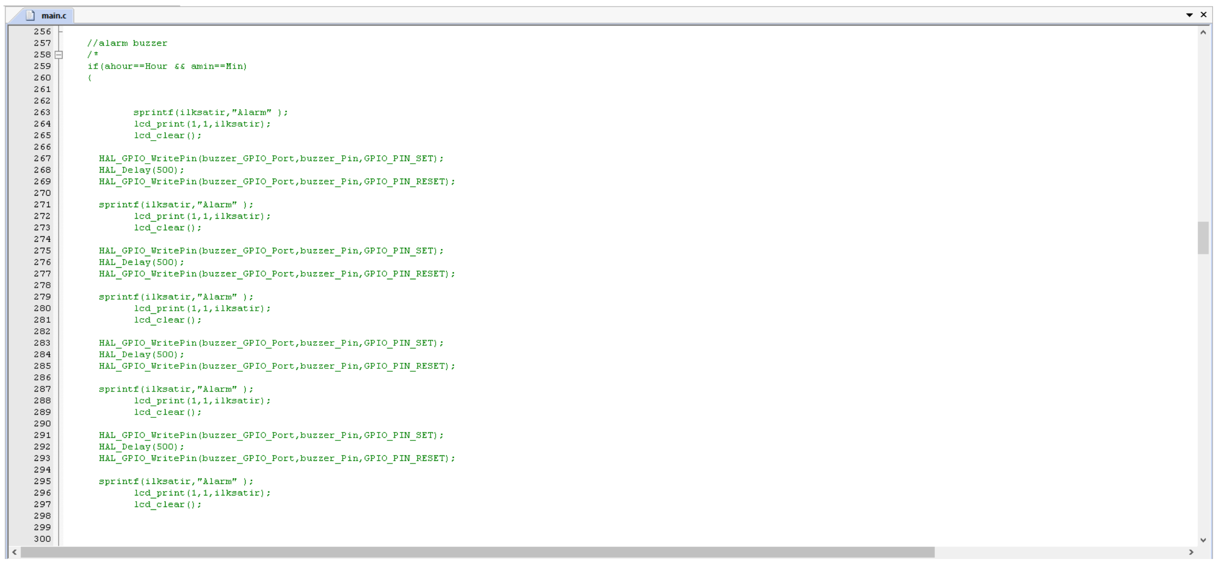
```c
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* 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_RTC_Init();
  MX_TIM1_Init();
  /* USER CODE BEGIN 2 */
  lcd_init(_LCD_4BIT, _LCD_FONT_5x8, _LCD_2LINE);


  HAL_RTC_GetTime(&hrtc,&RTC_Time,RTC_FORMAT_BIN);






  /*
  if(HAL_GPIO_ReadPin(GPIOB,button_Pin)==1 && HAL_GPIO_ReadPin(GPIOB,button1_Pin)==1)
    {

          ahour=Hour;
          amin= Min;



        sprintf(ilksatir,"Guncel Alarm" );
        lcd_print(1,1,ilksatir);
        sprintf(ikisatir,"%02d:%02d:%02d", ahour, amin, Sec);
        lcd_print(2,1,ikisatir);
        HAL_Delay(5000);


    }
```



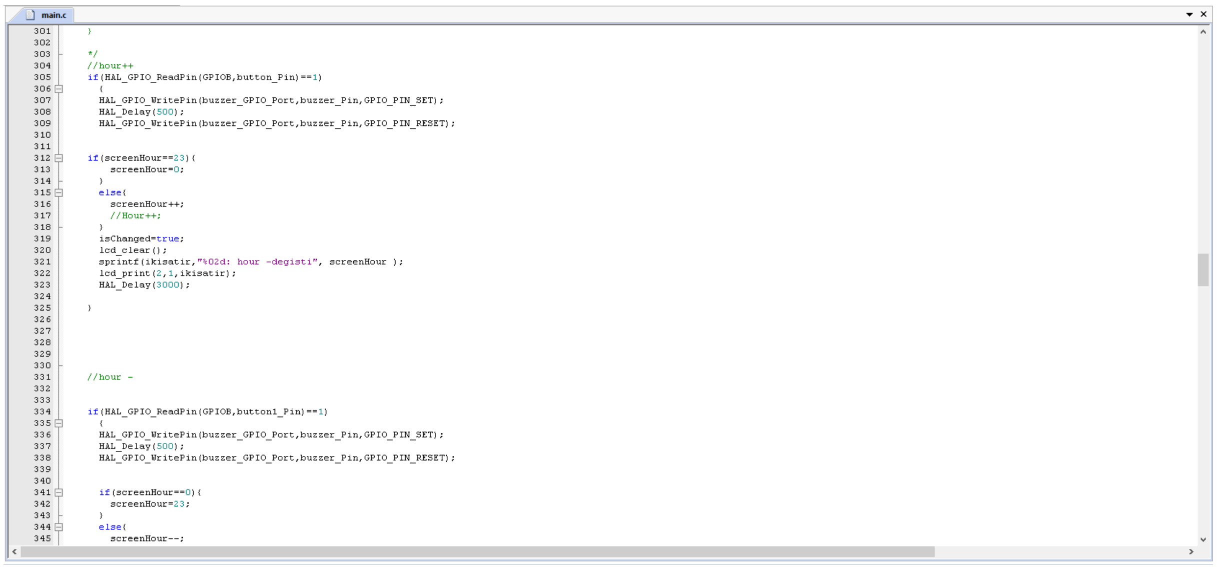
```c
    */




  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */





  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */



    HAL_RTC_GetTime(&hrtc,&RTC_Time,RTC_FORMAT_BIN);
    HAL_RTC_GetDate(&hrtc,&RTC_Date,RTC_FORMAT_BIN);


    currentHour=RTC_Time.Hours;
    currentMin=RTC_Time.Minutes;
    if(!isChanged){
      screenHour=currentHour;
      screenMin=currentMin;
    }


    Hour = RTC_Time.Hours + hour2;
    Min = RTC_Time.Minutes + min2;
    Sec = RTC_Time.Seconds;

    Date = RTC_Date.Date;
   Month = RTC_Date.Month;
   Year = RTC_Date.Year;


    sprintf(ilksatir,"Mustafa Aksoy" );
    sprintf(ikisatir,"%02d:%02d:%02d", currentHour, currentMin, Sec);
```



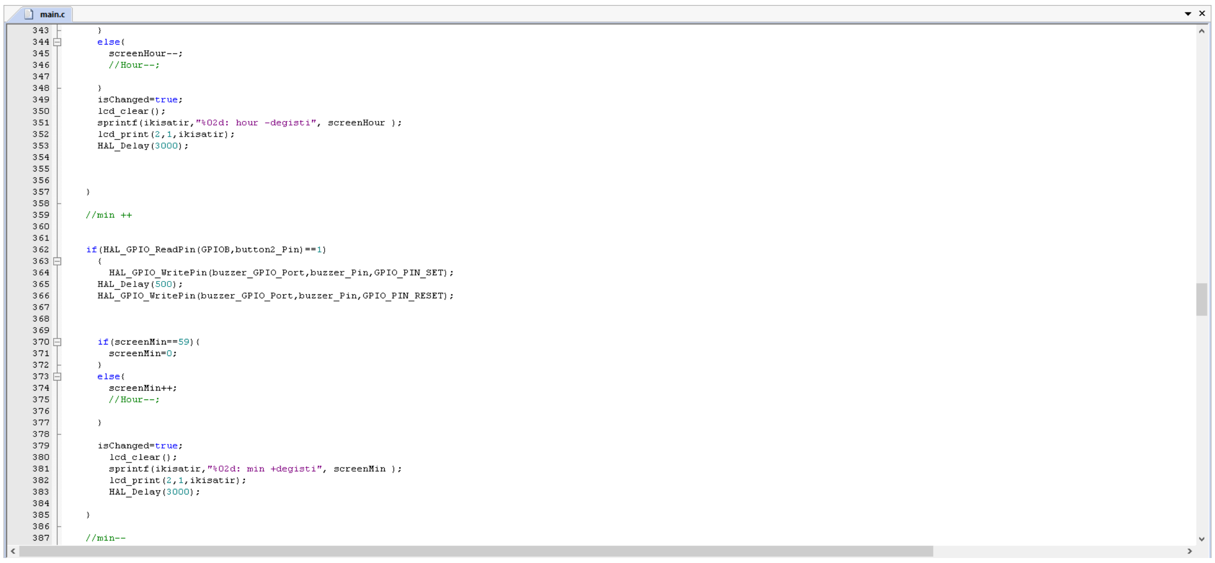
```c

sprintf(ilksatir,"Mustafa Aksoy" );
sprintf(ikisatir,"%02d:%02d:%02d", currentHour, currentMin, Sec);


lcd_print(1,1,ilksatir);
lcd_print(2,1,ikisatir);
HAL_Delay(200);

lcd_clear();



//alarms


if(HAL_GPIO_ReadPin(GPIOB,button_Pin)==1 && HAL_GPIO_ReadPin(GPIOB,button1_Pin)==1)
{


 alarmHour=screenHour;
 alarmMin=screenMin;
 isAlarmBuzzed=false;



   ahour=hour2 ;
   amin= min2 ;



  sprintf(ilksatir,"Guncel Alarm" );
  lcd_print(1,1,ilksatir);
  sprintf(ikisatir,"%02d:%02d", alarmHour, alarmMin);
  lcd_print(2,1,ikisatir);
  HAL_Delay(5000);


}




//alarm buzzer
/*
```



```c

//alarm buzzer
/*
if(ahour==Hour && amin==Min)
{


    sprintf(ilksatir,"Alarm" );
    lcd_print(1,1,ilksatir);
    lcd_clear();

 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_SET);
 HAL_Delay(500);
 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_RESET);

 sprintf(ilksatir,"Alarm" );
    lcd_print(1,1,ilksatir);
    lcd_clear();

 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_SET);
 HAL_Delay(500);
 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_RESET);

 sprintf(ilksatir,"Alarm" );
    lcd_print(1,1,ilksatir);
    lcd_clear();

 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_SET);
 HAL_Delay(500);
 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_RESET);

 sprintf(ilksatir,"Alarm" );
    lcd_print(1,1,ilksatir);
    lcd_clear();

 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_SET);
 HAL_Delay(500);
 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_RESET);

 sprintf(ilksatir,"Alarm" );
    lcd_print(1,1,ilksatir);
    lcd_clear();



```

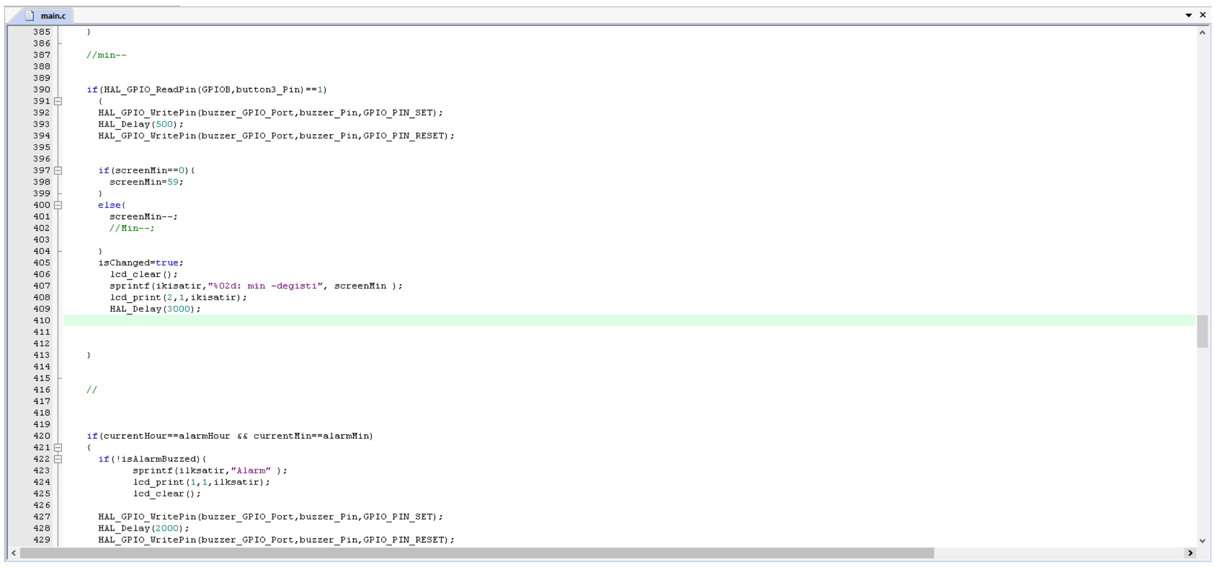


```c
}

*/
//hour++
if(HAL_GPIO_ReadPin(GPIOB,button_Pin)==1)
 {
 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_SET);
 HAL_Delay(500);
 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_RESET);


if(screenHour==23){
  screenHour=0;
 }
 else{
  screenHour++;
  //Hour++;
 }
 isChanged=true;
 lcd_clear();
 sprintf(ikisatir,"%02d: hour -degisti", screenHour );
 lcd_print(2,1,ikisatir);
 HAL_Delay(3000);

}





//hour -


if(HAL_GPIO_ReadPin(GPIOB,button1_Pin)==1)
 {
 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_SET);
 HAL_Delay(500);
 HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_RESET);


 if(screenHour==0){
  screenHour=23;
 }
 else{
  screenHour--;
```



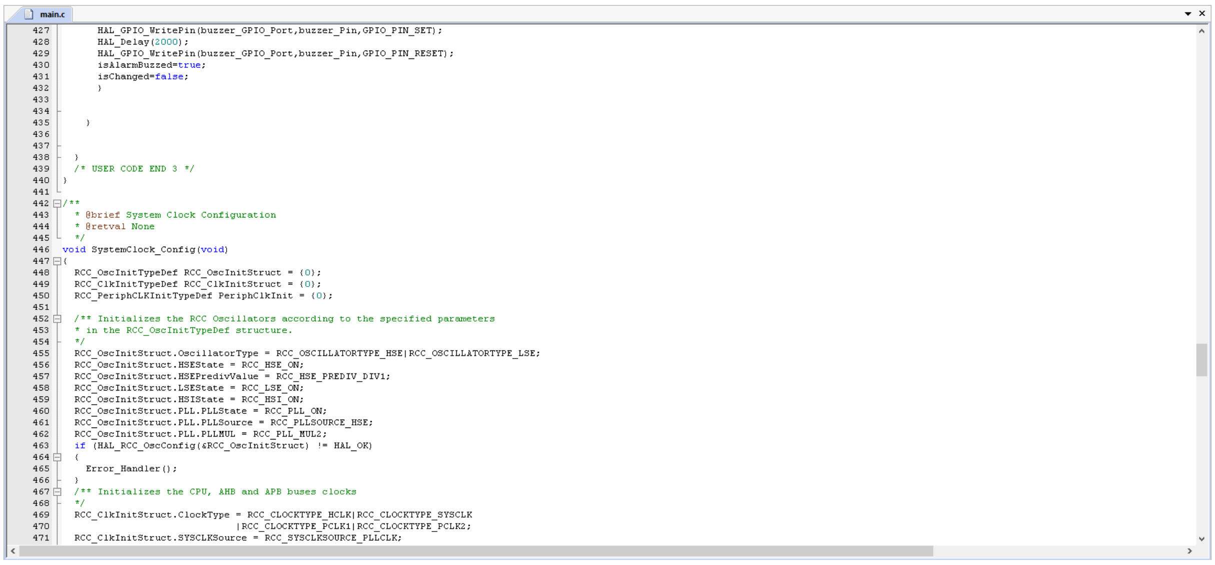
main.c

```c
      }
      else{
        screenHour--;
        //Hour--;

      }
      isChanged=true;
      lcd_clear();
      sprintf(ikisatir,"%02d: hour -degisti", screenHour );
      lcd_print(2,1,ikisatir);
      HAL_Delay(3000);



    }

    //min ++


    if(HAL_GPIO_ReadPin(GPIOB,button2_Pin)==1)
      {
        HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_SET);
      HAL_Delay(500);
      HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_RESET);



      if(screenMin==59){
        screenMin=0;
      }
      else{
        screenMin++;
        //Hour--;

      }

      isChanged=true;
        lcd_clear();
        sprintf(ikisatir,"%02d: min +degisti", screenMin );
        lcd_print(2,1,ikisatir);
        HAL_Delay(3000);

    }

    //min--
```



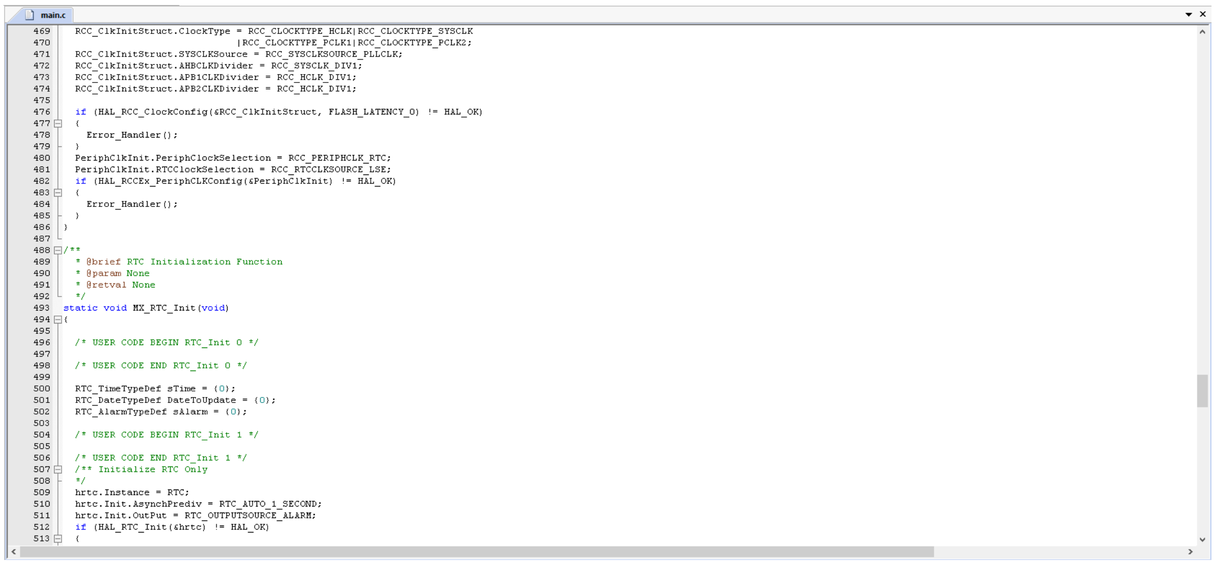
main.c

```c
    }

    //min--


    if(HAL_GPIO_ReadPin(GPIOB,button3_Pin)==1)
      {
      HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_SET);
      HAL_Delay(500);
      HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_RESET);


      if(screenMin==0){
        screenMin=59;
      }
      else{
        screenMin--;
        //Min--;

      }
      isChanged=true;
        lcd_clear();
        sprintf(ikisatir,"%02d: min -degisti", screenMin );
        lcd_print(2,1,ikisatir);
        HAL_Delay(3000);




    }


    //




    if(currentHour==alarmHour && currentMin==alarmMin)
    {
      if(!isAlarmBuzzed){
            sprintf(ilksatir,"Alarm" );
            lcd_print(1,1,ilksatir);
            lcd_clear();

      HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_SET);
      HAL_Delay(2000);
      HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_RESET);
```

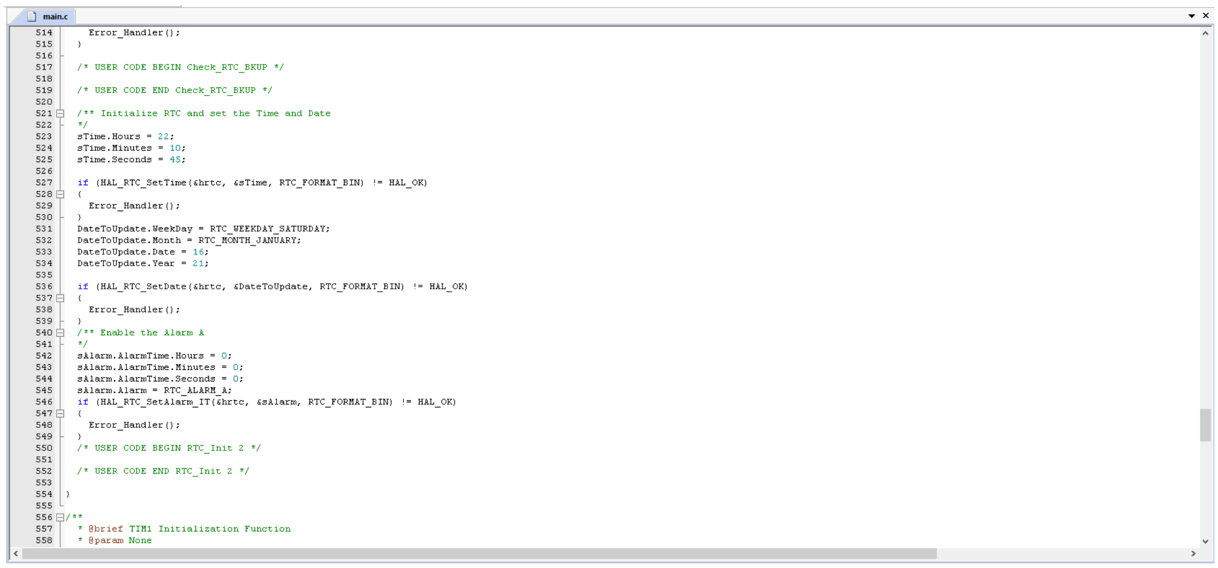


main.c

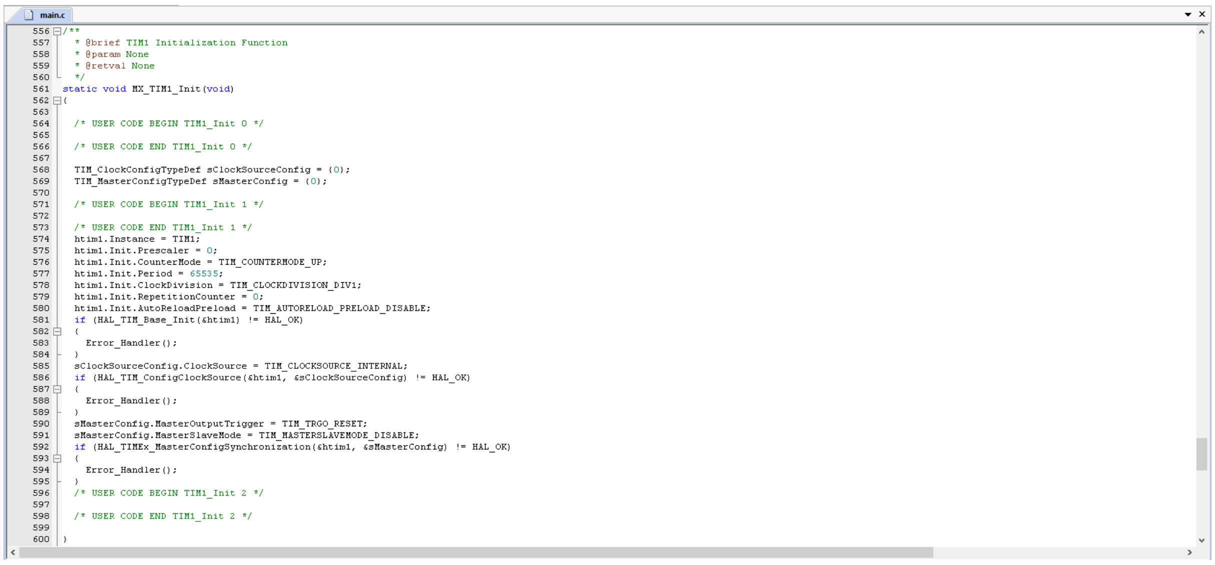
```c
      HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_SET);
      HAL_Delay(2000);
      HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,GPIO_PIN_RESET);
      isAlarmBuzzed=true;
      isChanged=false;
      }


    }


  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE|RCC_OSCILLATORTYPE_LSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.LSEState = RCC_LSE_ON;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
```



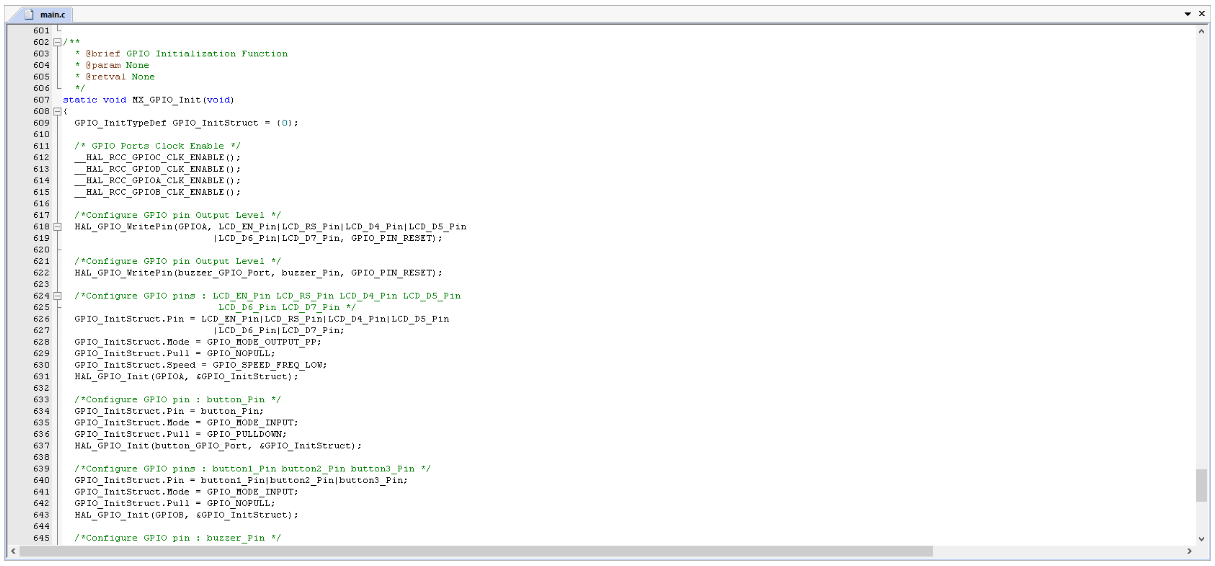
```c
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_RTC;
  PeriphClkInit.RTCClockSelection = RCC_RTCCLKSOURCE_LSE;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief RTC Initialization Function
  * @param None
  * @retval None
  */
static void MX_RTC_Init(void)
{

  /* USER CODE BEGIN RTC_Init 0 */

  /* USER CODE END RTC_Init 0 */

  RTC_TimeTypeDef sTime = {0};
  RTC_DateTypeDef DateToUpdate = {0};
  RTC_AlarmTypeDef sAlarm = {0};

  /* USER CODE BEGIN RTC_Init 1 */

  /* USER CODE END RTC_Init 1 */
  /** Initialize RTC Only
  */
  hrtc.Instance = RTC;
  hrtc.Init.AsynchPrediv = RTC_AUTO_1_SECOND;
  hrtc.Init.OutPut = RTC_OUTPUTSOURCE_ALARM;
  if (HAL_RTC_Init(&hrtc) != HAL_OK)
  {
```



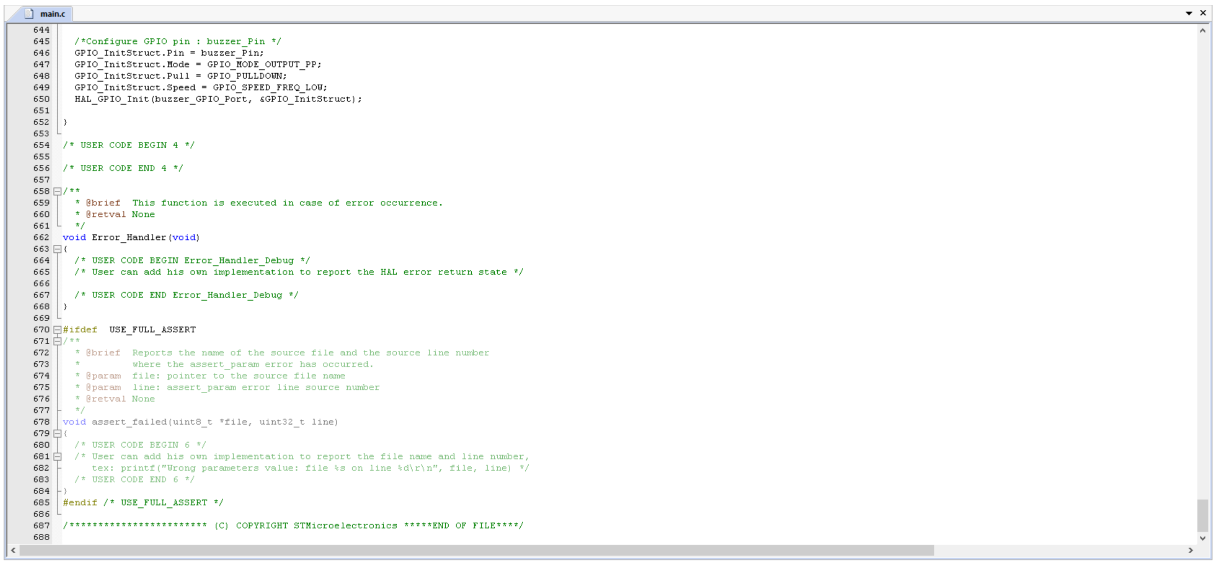
```c
    Error_Handler();
  }

  /* USER CODE BEGIN Check_RTC_BKUP */

  /* USER CODE END Check_RTC_BKUP */

  /** Initialize RTC and set the Time and Date
  */
  sTime.Hours = 22;
  sTime.Minutes = 10;
  sTime.Seconds = 45;

  if (HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BIN) != HAL_OK)
  {
    Error_Handler();
  }
  DateToUpdate.WeekDay = RTC_WEEKDAY_SATURDAY;
  DateToUpdate.Month = RTC_MONTH_JANUARY;
  DateToUpdate.Date = 16;
  DateToUpdate.Year = 21;

  if (HAL_RTC_SetDate(&hrtc, &DateToUpdate, RTC_FORMAT_BIN) != HAL_OK)
  {
    Error_Handler();
  }
  /** Enable the Alarm A
  */
  sAlarm.AlarmTime.Hours = 0;
  sAlarm.AlarmTime.Minutes = 0;
  sAlarm.AlarmTime.Seconds = 0;
  sAlarm.Alarm = RTC_ALARM_A;
  if (HAL_RTC_SetAlarm_IT(&hrtc, &sAlarm, RTC_FORMAT_BIN) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN RTC_Init 2 */

  /* USER CODE END RTC_Init 2 */

}

/**
  * @brief TIM1 Initialization Function
  * @param None
```



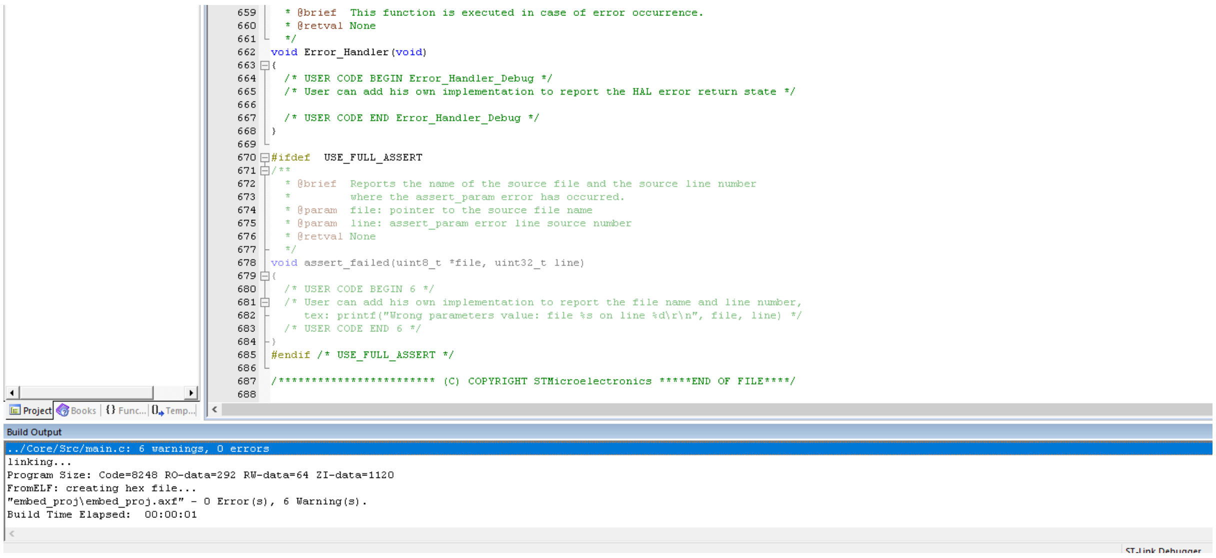
```c
/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 0;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 65535;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_Init 2 */

}
```



```c

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, LCD_EN_Pin|LCD_RS_Pin|LCD_D4_Pin|LCD_D5_Pin
                          |LCD_D6_Pin|LCD_D7_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(buzzer_GPIO_Port, buzzer_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : LCD_EN_Pin LCD_RS_Pin LCD_D4_Pin LCD_D5_Pin
                           LCD_D6_Pin LCD_D7_Pin */
  GPIO_InitStruct.Pin = LCD_EN_Pin|LCD_RS_Pin|LCD_D4_Pin|LCD_D5_Pin
                          |LCD_D6_Pin|LCD_D7_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : button_Pin */
  GPIO_InitStruct.Pin = button_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(button_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : button1_Pin button2_Pin button3_Pin */
  GPIO_InitStruct.Pin = button1_Pin|button2_Pin|button3_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : buzzer_Pin */
```



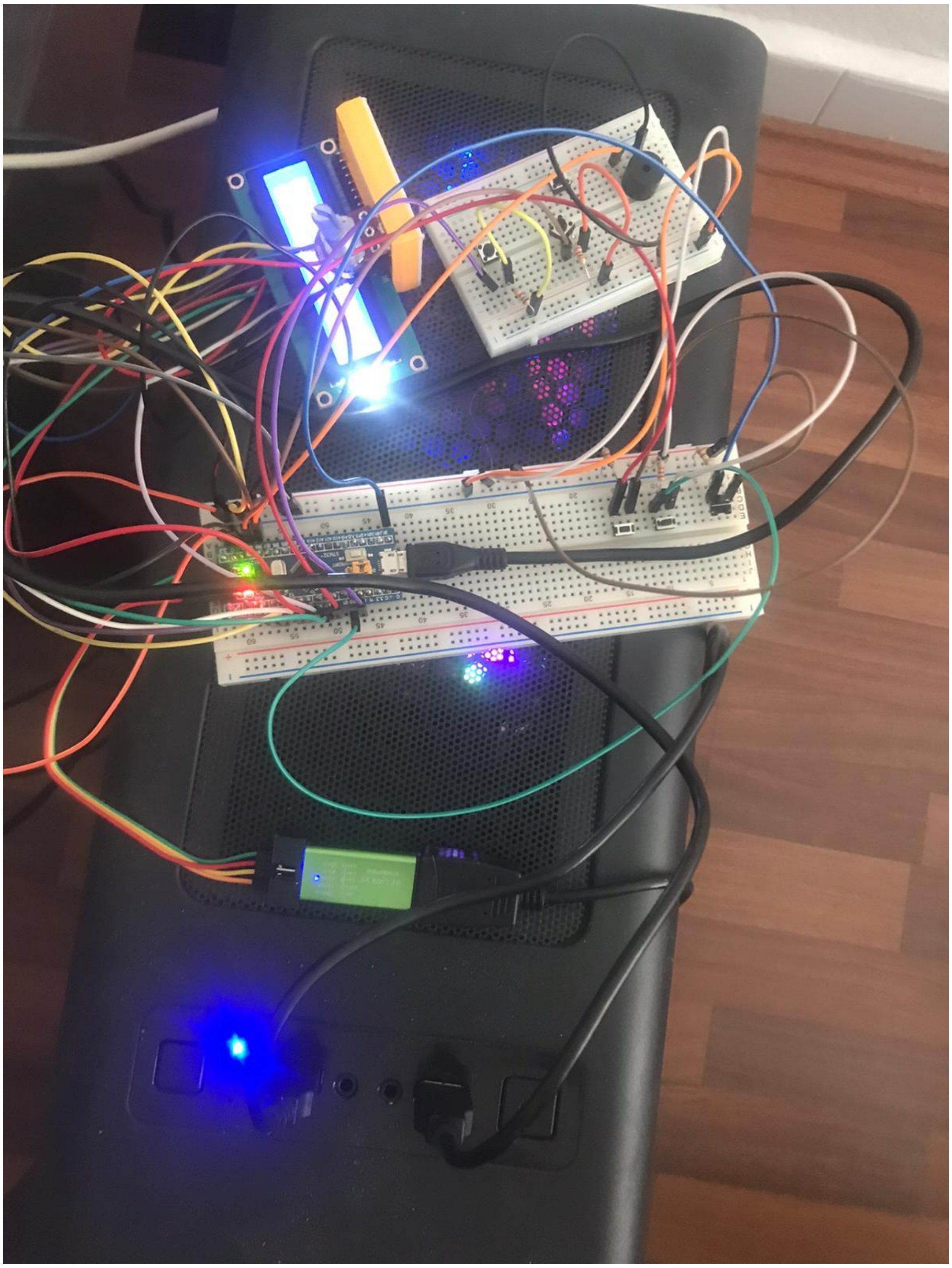
```c

  /*Configure GPIO pin : buzzer_Pin */
  GPIO_InitStruct.Pin = buzzer_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(buzzer_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/

```



```c
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/

```

```
../Core/Src/main.c: 6 warnings, 0 errors
linking...
Program Size: Code=8248 RO-data=292 RW-data=64 ZI-data=1120
FromELF: creating hex file...
"embed_proj\embed_proj.axf" - 0 Error(s), 6 Warning(s).
Build Time Elapsed:  00:00:01
```

2.3 Total System



Project link : https://www.youtube.com/watch?v=rWvaX9QZEO4&list=LL&index=1