```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 "adc.h"
#include "dma.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "lcd.h"
#include "stdio.h"
#include "string.h"
#include "interrupt.h"
#include "led.h"
#include "i2c_hal.h"

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
extern char rx_arry[50];
extern char rx_data;
extern char rx_pointer;
extern uint PA15_freq,PA15_duty,PA15_rise,PA15_fall;
extern uint PB4_freq,PB4_duty,PB4_rise,PB4_fall;
extern struct keys key[4];

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define PA1_FREQ 1000000

/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
u16 adc1_arry[1],adc2_arry[1];
float adc2_vol;
uint PA1_autoreload,PA1_compare;
uint PA1_freq = 2000;
uint PA1_duty = 10;
char lcd_arry[50];
char lcd_view;
char ctrl_mode = 0;//初始自动控制
char GEAR_num = 0;
char change_flag;
__IO uint32_t key_uwTick;
__IO uint32_t rx_uwTick;
__IO uint32_t led_uwTick;
float temp_value;
char led_num;
char rx_right_flag;
char key_sleep_flag;
char rx_sleep_flag;
char sleep_flag;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void rx_proc();
void pwm_proc();
void lcd_proc();
void key_proc();
void adc_proc();
void led_proc();
void sleep_proc();
/* USER CODE END 0 */

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

  /* USER CODE END 1 */

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

  /* 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_DMA_Init();
  MX_TIM2_Init();
  MX_TIM3_Init();
  MX_TIM6_Init();
  MX_TIM15_Init();
  MX_ADC2_Init();
  MX_USART1_UART_Init();
  MX_ADC1_Init();
  MX_TIM8_Init();
  /* USER CODE BEGIN 2 */

    LCD_Init();
  /* USER CODE END 2 */

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

    LCD_Clear(Black);
    LCD_SetBackColor(Black);
    LCD_SetTextColor(White);

    HAL_TIM_Base_Start_IT(&htim6);//定时器中断

    HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_2);//PWM

    HAL_UART_Receive_IT(&huart1, (uint8_t *)&rx_data, 1);//串口接收中断

    HAL_TIM_IC_Start_IT(&htim3,TIM_CHANNEL_1);//捕获中断
    HAL_TIM_IC_Start_IT(&htim3,TIM_CHANNEL_2);
    HAL_TIM_IC_Start_IT(&htim8,TIM_CHANNEL_1);

    HAL_ADC_Start_DMA(&hadc1,(uint32_t *)adc1_arry,1);//ADC_DMA
    HAL_ADC_Start_DMA(&hadc2,(uint32_t *)adc2_arry,1);

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

    /* USER CODE BEGIN 3 */
     rx_proc();
     pwm_proc();
     lcd_proc();
     key_proc();
     led_proc();
     adc_proc();
     sleep_proc();
    }
  /* USER CODE END 3 */
}

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

  /** Configure the main internal regulator output voltage
  */
  HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV3;
  RCC_OscInitStruct.PLL.PLLN = 20;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  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
```

```c
                              |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_2) != HAL_OK)
  {
    Error_Handler();
  }
}

/* USER CODE BEGIN 4 */
void sleep_proc()
{
    if(uwTick-key_uwTick>5000)//5s内按键无动作
        key_sleep_flag = 1;
    else key_sleep_flag = 0;

    if(uwTick-rx_uwTick>5000)//5s内串口未下发命令
        rx_sleep_flag = 1;
    else rx_sleep_flag = 0;

    if(key_sleep_flag&&rx_sleep_flag)
        lcd_view = 1;


}
void led_proc()
{
    if(ctrl_mode == 0)
        led_num |=0x80;//亮
    else
        led_num &=0x7f;//灭

    if(GEAR_num == 1)
        led_num |=0x01;//亮
    else led_num &=0xfe;//灭

    if(GEAR_num == 2)
        led_num |=0x02;//亮
    else led_num &=0xfd;//灭

    if(GEAR_num == 3)
        led_num |=0x04;//亮
    else led_num &=0xfb;//灭

    if(rx_right_flag == 1)
    {
        led_num |=0x08;//亮
        if(uwTick-led_uwTick>3000)
        {
            rx_right_flag = 0;
            led_num &=0xf7;//灭
        }
    }

    led_disp(led_num);


}
void adc_proc()
{
    adc2_vol = adc2_arry[0]*3.3/4096;
    if(ctrl_mode == 0)//自动控制模式
    {
        if(adc2_vol<1.5)
            GEAR_num = 1;
        else if(adc2_vol<2)
            GEAR_num = 2;
        else GEAR_num = 3;
    }

    temp_value = 10*adc2_vol + 10;
    if(temp_value>40)
        temp_value = 40;

}
void key_proc()
{
    for(int i=0;i<4;i++)
        if(key[i].short_flag == 1)
        {
            LCD_Clear(Black);
            key_uwTick = uwTick;
        }

    if(key[0].short_flag == 1)
    {
        key[0].short_flag = 0;
        if(lcd_view == 0)
        {
            ctrl_mode = !ctrl_mode;
        }
        else change_flag = 1;
    }

    if(key[1].short_flag == 1)
    {
        key[1].short_flag = 0;
        if(lcd_view == 0)//数据界面
        {
        if(ctrl_mode == 1)//手动控制
            {
                GEAR_num++;
                if(GEAR_num>3)
                    GEAR_num = 3;
            }
        }
        else
            change_flag = 1;
    }

    if(key[2].short_flag == 1)
    {
        key[2].short_flag = 0;
        if(lcd_view == 0)//数据界面
        {
            if(ctrl_mode == 1)//手动控制
            {
                GEAR_num--;
                if(GEAR_num<1)
                    GEAR_num = 1;
            }
        }
        else
            change_flag = 1;
    }

    if(key[3].short_flag == 1)
        key[3].short_flag = 0;

    if(change_flag == 1)
    {
        lcd_view = 0;//切换到数据界面
        change_flag = 0;//标志位清零
    }

}
void lcd_proc()
{
    if(lcd_view == 0)//
    {
    sprintf(lcd_arry,"        DATA");
    LCD_DisplayStringLine(Line1,(u8 *)lcd_arry);
    sprintf(lcd_arry,"     TEMP:%-4.1f",temp_value);
    LCD_DisplayStringLine(Line3,(u8 *)lcd_arry);
    if(ctrl_mode == 0)
        sprintf(lcd_arry,"     MODE:Auto");
    else sprintf(lcd_arry,"     MODE:Manu");
    LCD_DisplayStringLine(Line4,(u8 *)lcd_arry);
    sprintf(lcd_arry,"     GEAR:%d",GEAR_num);
    LCD_DisplayStringLine(Line5,(u8 *)lcd_arry);

    }
    else
    {
    sprintf(lcd_arry,"                    ");
    LCD_DisplayStringLine(Line1,(u8 *)lcd_arry);
    sprintf(lcd_arry,"                    ");
    LCD_DisplayStringLine(Line3,(u8 *)lcd_arry);
    sprintf(lcd_arry,"     SLEEPING   ");
    LCD_DisplayStringLine(Line4,(u8 *)lcd_arry);
    sprintf(lcd_arry,"     TEMP:%-4.1f",temp_value);
    LCD_DisplayStringLine(Line5,(u8 *)lcd_arry);
    }

}
void pwm_proc()
{
    if(GEAR_num == 1)
        PA1_duty = 10;

    if(GEAR_num == 2)
        PA1_duty = 40;

    if(GEAR_num == 3)
        PA1_duty = 80;

    PA1_autoreload = PA1_FREQ/PA1_freq;
    PA1_compare = PA1_autoreload*PA1_duty/100;

    __HAL_TIM_SET_AUTORELOAD(&htim2,PA1_autoreload);
    __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2,PA1_compare);

}
void rx_proc()
{
    if(strcmp(rx_arry,"B1") == 0 || strcmp(rx_arry,"B2") == 0 || strcmp(rx_arry,"B3") == 0)
    {
                      = 1;
        led_uwTick = uwTick;
        rx_uwTick = uwTick;
    }

    if(rx_pointer!=0)
    {
        int temp = rx_pointer;
        HAL_Delay(1);
        if(temp==rx_pointer)
        {
            printf("arry is :%s",rx_arry);
            if(rx_pointer == 2)
            {
                if(strcmp(rx_arry,"B1") == 0)
                {
                    if(lcd_view == 0)
                        key[0].short_flag = 1;
                    else change_flag = 1;
                }

                else if(strcmp(rx_arry,"B2") == 0)
                    if(lcd_view == 0)
```

```
                            key[1].short_flag = 1;
                        else change_flag = 1;
                    else if(strcmp(rx_arry,"B3") == 0)
                        if(lcd_view == 0)
                            key[2].short_flag = 1;
                        else change_flag = 1;
                    else printf("NULL");

                }
                else printf("NULL");
                rx_pointer = 0;memset(rx_arry,0,50);
        }
    }


}
int fputc(int ch, FILE *f)
{
  HAL_UART_Transmit(&huart1, (const uint8_t *)&ch, 1,20);
  return ch;
}


/* 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 */

```