

```

1  /* USER CODE BEGIN Header */
2  /**
3   *
4   * @file      : main.c
5   * @brief     : Main program body
6   *
7   * @attention
8   *
9   * Copyright (c) 2024 STMicroelectronics.
10  * All rights reserved.
11  *
12  * This software is licensed under terms that can be found in the LICENSE file
13  * in the root directory of this software component.
14  * If no LICENSE file comes with this software, it is provided AS-IS.
15  *
16  */
17  /* USER CODE END Header */
18  /* Includes -----*/
19  #include "main.h"
20  #include "adc.h"
21  #include "dac.h"
22  #include "dma.h"
23  #include "tim.h"
24  #include "usart.h"
25  #include "gpio.h"
26
27  /* Private includes -----*/
28  /* USER CODE BEGIN Includes */
29  #include "lcd.h"
30  #include "led.h"
31  #include "interrupt.h"
32  #include "string.h"
33  #include "stdio.h"
34  #include "i2c_hal.h"
35
36  /* USER CODE END Includes */
37
38  /* Private typedef -----*/
39  /* USER CODE BEGIN PTD */
40  extern struct keys key[4];
41  extern char rx_array[50];
42  extern char rx_data;
43  extern char rx_pointer;
44  extern uint PA15_rise, PA15_fall; //TIM2
45  extern uint PA15_freq, PA15_duty;
46  extern uint PB4_rise, PB4_fall; //TIM3
47  extern uint PB4_freq, PB4_duty;
48  /* USER CODE END PTD */
49
50  /* Private define -----*/
51  /* USER CODE BEGIN PD */
52
53  /* USER CODE END PD */
54
55  /* Private macro -----*/
56  /* USER CODE BEGIN PM */
57
58  /* USER CODE END PM */
59
60  /* Private variables -----*/
61  /* USER CODE BEGIN PV */
62  __IO uint32_t key_uwTick;
63  __IO uint32_t adc_uwTick;
64  __IO uint32_t led_uwTick;
65  uint16_t adc1_array[2];
66  uint16_t adc2_array[1];
67  char lcd_array[50];
68  char lcd_view;
69
70
71

```

```

72 float R37_vol;
73 float distance;
74 uint direction;//0直行, 1左, 2右
75 __IO uint32_t dir_uwTick;
76 uint left_turn_flag;
77 uint right_turn_flag;
78 char led_num;
79
80
81
82 /* USER CODE END PV */
83
84 /* Private function prototypes -----*/
85 void SystemClock_Config(void);
86 /* USER CODE BEGIN PFP */
87
88 /* USER CODE END PFP */
89
90 /* Private user code -----*/
91 /* USER CODE BEGIN 0 */
92 void key_proc();
93 void rx_proc();
94 void adc_proc();
95 void lcd_proc();
96 void led_proc();
97
98
99 /* USER CODE END 0 */
100
101 /**
102  * @brief The application entry point.
103  * @retval int
104  */
105 int main(void)
106 {
107     /* USER CODE BEGIN 1 */
108
109     /* USER CODE END 1 */
110
111     /* MCU Configuration-----*/
112
113     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
114     HAL_Init();
115
116     /* USER CODE BEGIN Init */
117
118     /* USER CODE END Init */
119
120     /* Configure the system clock */
121     SystemClock_Config();
122
123     /* USER CODE BEGIN SysInit */
124
125     /* USER CODE END SysInit */
126
127     /* Initialize all configured peripherals */
128     MX_GPIO_Init();
129     MX_DMA_Init();
130     MX_TIM6_Init();
131     MX_USART1_UART_Init();
132     MX_ADC2_Init();
133     MX_ADC1_Init();
134     MX_DAC1_Init();
135     MX_TIM2_Init();
136     MX_TIM3_Init();
137     MX_TIM8_Init();
138     /* USER CODE BEGIN 2 */
139
140     HAL_TIM_Base_Start_IT(&htim6);
141
142     LCD_Init();

```

```

143 LCD_Clear(Black);
144 LCD_SetBackColor(Black);
145 LCD_SetTextColor(White);
146
147 HAL_UART_Receive_IT(&huart1, (uint8_t *)&rx_data, 1);
148 HAL_ADC_Start_DMA(&hadc1, (uint32_t *)adc1_array, 2);
149 HAL_ADC_Start_DMA(&hadc2, (uint32_t *)adc2_array, 1);
150 HAL_DAC_Start(&hdac1, DAC_CHANNEL_1);
151
152 HAL_TIM_IC_Start_IT(&htim8, TIM_CHANNEL_1);
153 HAL_TIM_IC_Start_IT(&htim8, TIM_CHANNEL_2);
154 HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1);
155 HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_2);
156
157 HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);
158
159 led_disp(0x00);
160
161 /* USER CODE END 2 */
162
163 /* Infinite loop */
164 /* USER CODE BEGIN WHILE */
165 while (1)
166 {
167     /* USER CODE END WHILE */
168
169     /* USER CODE BEGIN 3 */
170     key_proc();
171     lcd_proc();
172     rx_proc();
173     adc_proc();
174     led_proc();
175 }
176 /* USER CODE END 3 */
177 }
178
179 /**
180  * @brief System Clock Configuration
181  * @retval None
182  */
183 void SystemClock_Config(void)
184 {
185     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
186     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
187
188     /** Configure the main internal regulator output voltage
189     */
190     HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);
191
192     /** Initializes the RCC Oscillators according to the specified parameters
193     * in the RCC_OscInitTypeDef structure.
194     */
195     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
196     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
197     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
198     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
199     RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV3;
200     RCC_OscInitStruct.PLL.PLLN = 20;
201     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
202     RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
203     RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
204     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
205     {
206         Error_Handler();
207     }
208
209     /** Initializes the CPU, AHB and APB buses clocks
210     */
211     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
212                                     | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
213     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;

```

```

214 RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
215 RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
216 RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
217
218 if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
219 {
220     Error_Handler();
221 }
222
223
224 /* USER CODE BEGIN 4 */
225 void led_proc()
226 {
227     if(uwTick - led_uwTick < 100)
228         return;
229     led_uwTick = uwTick;
230
231     if(lcd_view == 0)
232     {
233         if(direction == 1)
234             led_num = led_num ^ 0x01;
235         else
236             led_num = led_num & 0xfe;
237         if(direction == 2)
238             led_num = led_num ^ 0x02;
239         else
240             led_num = led_num & 0xfd;
241     }
242     else
243     {
244         led_num = led_num & 0xfc;
245     }
246
247
248
249     if(lcd_view == 1)
250         led_num = led_num | 0x80;
251     else led_num = led_num & 0x7f;
252
253     led_disp(led_num);
254 }
255
256 void lcd_proc()
257 {
258     if(lcd_view == 0)
259     {
260         sprintf(lcd_array, "          DATA");
261         LCD_DisplayStringLine(Line1, (u8 *)lcd_array);
262
263         if(direction == 0)
264             sprintf(lcd_array, "          N:S");
265         else if(direction == 1)
266             sprintf(lcd_array, "          N:L");
267         else if(direction == 2)
268             sprintf(lcd_array, "          N:R");
269         LCD_DisplayStringLine(Line3, (u8 *)lcd_array);
270
271         sprintf(lcd_array, "          D:%-6.1f", distance);
272         LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
273     }
274     if(lcd_view == 1)
275     {
276         sprintf(lcd_array, "          WARN");
277         LCD_DisplayStringLine(Line4, (u8 *)lcd_array);
278     }
279 }
280
281 void adc_proc()
282 {
283     R37_vol = adc2_array[0] * 3.3 / 4096;
284     if(R37_vol > 3)

```

```

285     distance = 300;
286     else distance = 100*R37_vol;
287 }
288 void rx_proc()
289 {
290     if(rx_pointer!=0)
291     {
292         int temp = rx_pointer;
293         HAL_Delay(1);
294         if(temp == rx_pointer)//接收完毕
295         {
296             if(lcd_view == 0)
297             {
298                 if((strcmp(rx_array,"L") == 0) && ((uwTick - dir_uwTick) > 5000 || dir_uwTick == 0))
299                     //5s内不能再次改变方向
300                 {
301                     direction = 1;//导航方向切换为左
302                     left_turn_flag = 1;//
303                     dir_uwTick = uwTick;//开始计时
304                 }
305             }
306             else if((strcmp(rx_array,"R") == 0) && ((uwTick - dir_uwTick) > 5000 || dir_uwTick
307 == 0))
308             {
309                 direction = 2;//导航方向切换为右
310                 right_turn_flag = 1;
311                 dir_uwTick = uwTick;//开始计时
312             }
313             else printf("fei fa zi fu:ERROR\r\n");
314         }
315         else printf("WAIT\r\n");//偏离导航界面下
316
317         rx_pointer = 0;memset(rx_array,0,50);
318     }
319 }
320 }
321 }
322 void key_proc()
323 {
324
325     for(int i=0;i<4;i++)
326         if(key[i].short_flag == 1)
327             LCD_Clear(Black);
328
329     if(key[0].short_flag == 1)
330     {
331         key[0].short_flag = 0;
332         if(lcd_view == 1)//偏离导航界面下
333         {
334             lcd_view = 0;
335             direction = 0;
336             printf("diao_tou:Success\r\n");//车辆掉头成功
337         }
338     }
339
340     if(key[2].short_flag == 1)
341     {
342         key[2].short_flag = 0;
343         if(lcd_view == 0)
344         {
345             if(direction == 0)
346             {
347                 lcd_view = 1;
348                 printf("no_rx_but_turn:Warn\r\n");
349             }
350             else if(direction == 1)
351             {
352                 if(uwTick - dir_uwTick<5000)
353                 {
354                     left_turn_flag = 0;//转弯标志结束

```

```

355         printf("rx_and_turn:Success\r\n"); //车辆转向成功
356         direction = 0; //导航方向切换为直行
357     }
358 }
359 }
360 }
361
362 if(key[3].short_flag == 1)
363 {
364     key[3].short_flag = 0;
365     if(lcd_view == 0)
366     {
367         if(direction == 0)
368         {
369             lcd_view = 1;
370             printf("no_rx_but_turn:Warn\r\n");
371         }
372         else if(direction == 2)
373         {
374             if(uwTick - dir_uwTick < 5000)
375             {
376                 right_turn_flag = 0; //转弯标志结束
377                 printf("rx_and_turn:Success\r\n"); //车辆转向成功
378                 direction = 0; //导航方向切换为直行
379             }
380         }
381     }
382 }
383
384 if((uwTick - dir_uwTick > 5000) && ((right_turn_flag == 1) || (left_turn_flag == 1)))
385 {
386     right_turn_flag = 0;
387     left_turn_flag = 0;
388     lcd_view = 1;
389     printf("rx_but_not_turn:Warn\r\n");
390     LCD_Clear(Black);
391 }
392 }
393
394 int fputc(int ch, FILE *f)
395 {
396     HAL_UART_Transmit(&huart1, (const uint8_t *)&ch, 1, 20);
397     return ch;
398 }
399
400
401 /* USER CODE END 4 */
402
403 /**
404  * @brief This function is executed in case of error occurrence.
405  * @retval None
406  */
407 void Error_Handler(void)
408 {
409     /* USER CODE BEGIN Error_Handler_Debug */
410     /* User can add his own implementation to report the HAL error return state */
411     __disable_irq();
412     while (1)
413     {
414     }
415     /* USER CODE END Error_Handler_Debug */
416 }
417
418 #ifndef USE_FULL_ASSERT
419 /**
420  * @brief Reports the name of the source file and the source line number
421  * where the assert_param error has occurred.
422  * @param file: pointer to the source file name
423  * @param line: assert_param error line source number
424  * @retval None
425  */

```

```
426 void assert_failed(uint8_t *file, uint32_t line)
427
428 /* USER CODE BEGIN 6 */
429 /* User can add his own implementation to report the file name and line number,
430    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
431 /* USER CODE END 6 */
432
433 #endif /* USE_FULL_ASSERT */
434
```