```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.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 */

/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim2;

/* USER CODE BEGIN PV */
char Q=0;
unsigned int cnt=0;
char button;
unsigned int T=400;
char flag=0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */
```

```
 60
 61 /* Private user code ---------------------------------------------------------*/
 62 /* USER CODE BEGIN 0 */
 63 /* Timer2 Interrupt Service Routine*/
 64 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim2)
 65 {
 66 switch (Q)
 67 {
 68 case 0:
 69  button = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13); //read user button
 70  if (button== GPIO_PIN_RESET) Q = 1;
 71 break;
 72
 73 case 1:
 74  button = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13); //read user button
 75  if (button== GPIO_PIN_SET) Q = 2;
 76 break;
 77 case 2:
 78     if (flag == 0) T=400;
 79     else T=800;
 80     flag = (flag+1)%2;
 81     Q = 0;
 82 break;
 83 }
 84 if (cnt < T/4)
 85 {
 86      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, 0); // all OFF
 87      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, 0);
 88      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 0);
 89 }
 90
 91 if (cnt >= T/4 && cnt < T/2)
 92 {
 93      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, 1); // RED
 94      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, 0);
 95      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 0);
 96 }
 97
 98 if (cnt >= T/2 && cnt <3*T/4)
 99 {
100      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, 0);
101      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, 1);// GREEN
102      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 0);
103 }
104
105 if (cnt >= 3*T/4 && cnt < T)
106 {
107      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, 0);
108      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, 0);
109      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 1); // BLUE
110 }
111 cnt++;
112 if(cnt == T) cnt=0;
113
114
115 }
116
117
118
```

```
119
120
121 /* USER CODE END 0 */
122
123 /**
124   * @brief  The application entry point.
125   * @retval int
126   */
127 int main(void)
128 {
129
130   /* USER CODE BEGIN 1 */
131
132   /* USER CODE END 1 */
133
134   /* MCU Configuration--------------------------------------------------------*/
135
136   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
137   HAL_Init();
138
139   /* USER CODE BEGIN Init */
140
141   /* USER CODE END Init */
142
143   /* Configure the system clock */
144   SystemClock_Config();
145
146   /* USER CODE BEGIN SysInit */
147
148   /* USER CODE END SysInit */
149
150   /* Initialize all configured peripherals */
151   MX_GPIO_Init();
152   MX_TIM2_Init();
153   /* USER CODE BEGIN 2 */
154   HAL_TIM_Base_Start_IT(&htim2);  //enable Timer 2 interrupt
155   /* USER CODE END 2 */
156
157   /* Infinite loop */
158   /* USER CODE BEGIN WHILE */
159   while (1)
160   {
161     /* USER CODE END WHILE */
162
163     /* USER CODE BEGIN 3 */
164   }
165   /* USER CODE END 3 */
166 }
167
168 /**
169   * @brief System Clock Configuration
170   * @retval None
171   */
172 void SystemClock_Config(void)
173 {
174   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
175   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
176
177   /** Initializes the RCC Oscillators according to the specified parameters
```

```
178   * in the RCC_OscInitTypeDef structure.
179   */
180   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
181   RCC_OscInitStruct.HSEState = RCC_HSE_ON;
182   RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
183   RCC_OscInitStruct.HSIState = RCC_HSI_ON;
184   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
185   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
186   RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
187   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
188   {
189     Error_Handler();
190   }
191
192   /** Initializes the CPU, AHB and APB buses clocks
193   */
194   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
195                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
196   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
197   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
198   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
199   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
200
201   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
202   {
203     Error_Handler();
204   }
205 }
206
207 /**
208   * @brief TIM2 Initialization Function
209   * @param None
210   * @retval None
211   */
212 static void MX_TIM2_Init(void)
213 {
214
215   /* USER CODE BEGIN TIM2_Init 0 */
216
217   /* USER CODE END TIM2_Init 0 */
218
219   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
220   TIM_MasterConfigTypeDef sMasterConfig = {0};
221   TIM_OC_InitTypeDef sConfigOC = {0};
222
223   /* USER CODE BEGIN TIM2_Init 1 */
224
225   /* USER CODE END TIM2_Init 1 */
226   htim2.Instance = TIM2;
227   htim2.Init.Prescaler = 7199;
228   htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
229   htim2.Init.Period = 100;
230   htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
231   htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
232   if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
233   {
234     Error_Handler();
235   }
236   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
```

```
237   if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
238   {
239     Error_Handler();
240   }
241   if (HAL_TIM_OC_Init(&htim2) != HAL_OK)
242   {
243     Error_Handler();
244   }
245   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
246   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
247   if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
248   {
249     Error_Handler();
250   }
251   sConfigOC.OCMode = TIM_OCMODE_TIMING;
252   sConfigOC.Pulse = 0;
253   sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
254   sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
255   if (HAL_TIM_OC_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
256   {
257     Error_Handler();
258   }
259   /* USER CODE BEGIN TIM2_Init 2 */
260
261   /* USER CODE END TIM2_Init 2 */
262   HAL_TIM_MspPostInit(&htim2);
263
264 }
265
266 /**
267   * @brief GPIO Initialization Function
268   * @param None
269   * @retval None
270   */
271 static void MX_GPIO_Init(void)
272 {
273   GPIO_InitTypeDef GPIO_InitStruct = {0};
274 /* USER CODE BEGIN MX_GPIO_Init_1 */
275 /* USER CODE END MX_GPIO_Init_1 */
276
277   /* GPIO Ports Clock Enable */
278   __HAL_RCC_GPIOC_CLK_ENABLE();
279   __HAL_RCC_GPIOD_CLK_ENABLE();
280   __HAL_RCC_GPIOA_CLK_ENABLE();
281   __HAL_RCC_GPIOB_CLK_ENABLE();
282
283   /*Configure GPIO pin Output Level */
284   HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6
285                           |GPIO_PIN_8, GPIO_PIN_RESET);
286
287   /*Configure GPIO pin Output Level */
288   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_12|GPIO_PIN_13
289                           |GPIO_PIN_14|GPIO_PIN_15, GPIO_PIN_RESET);
290
291   /*Configure GPIO pin : PC13 */
292   GPIO_InitStruct.Pin = GPIO_PIN_13;
293   GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
294   GPIO_InitStruct.Pull = GPIO_PULLUP;
295   HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
```

```
296
297   /*Configure GPIO pins : PA3 PA4 PA5 PA6
298                            PA8 */
299   GPIO_InitStruct.Pin = GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6
300                         |GPIO_PIN_8;
301   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
302   GPIO_InitStruct.Pull = GPIO_NOPULL;
303   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
304   HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
305
306   /*Configure GPIO pins : PB0 PB1 PB12 PB13
307                            PB14 PB15 */
308   GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_12|GPIO_PIN_13
309                         |GPIO_PIN_14|GPIO_PIN_15;
310   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
311   GPIO_InitStruct.Pull = GPIO_NOPULL;
312   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
313   HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
314
315 /* USER CODE BEGIN MX_GPIO_Init_2 */
316 /* USER CODE END MX_GPIO_Init_2 */
317 }
318
319 /* USER CODE BEGIN 4 */
320
321 /* USER CODE END 4 */
322
323 /**
324   * @brief  This function is executed in case of error occurrence.
325   * @retval None
326   */
327 void Error_Handler(void)
328 {
329   /* USER CODE BEGIN Error_Handler_Debug */
330   /* User can add his own implementation to report the HAL error return state */
331   __disable_irq();
332   while (1)
333   {
334   }
335   /* USER CODE END Error_Handler_Debug */
336 }
337
338 #ifdef  USE_FULL_ASSERT
339 /**
340   * @brief  Reports the name of the source file and the source line number
341   *         where the assert_param error has occurred.
342   * @param  file: pointer to the source file name
343   * @param  line: assert_param error line source number
344   * @retval None
345   */
346 void assert_failed(uint8_t *file, uint32_t line)
347 {
348   /* USER CODE BEGIN 6 */
349   /* User can add his own implementation to report the file name and line number,
350      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
351   /* USER CODE END 6 */
352 }
353 #endif /* USE_FULL_ASSERT */
354
```