```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2025 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 */
#include <stdio.h>
/* 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 */
#define DUTY_VAL 14
/* USER CODE END PM */

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

UART_HandleTypeDef huart1;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_TIM2_Init(void);
/* USER CODE BEGIN PFP */
int _write(int, char*, int);
void stuurDataIrSolo(uint8_t);
void stuurDataIrDuo(uint8_t, uint8_t);
void stuurDataIrAll(uint8_t, uint8_t, uint8_t);
/* USER CODE END PFP */

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

```c
int _write(int file, char *ptr, int len) {
    for int i = 0; i < len; i++){
        if(ptr[i]=='\n'){
            HAL_UART_Transmit(&huart1, (uint8_t*)"\r", 1, HAL_MAX_DELAY);
        }
        HAL_UART_Transmit(&huart1,  uint8_t*)&ptr[i], 1, HAL_MAX_DELAY);
    }
    return len;
}
void stuurDataIrSolo  uint8_t data  //MSB eerst
{
    uint8_t bit;
    for  uint8_t i = 0; i < 8; i++)
    {
        bit = data & 128;
        data = data <<  1;
        if (bit == 128)
        {
            htim2.Instance->CCR2 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay 2); //3ms wachten
            htim2.Instance->CCR2 = 0; //Duty Cycle op 0% => led uit
            HAL_Delay 0); //1ms wachten
        }
        else
        {
            htim2.Instance->CCR2 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay 0); //3ms wachten
            htim2.Instance->CCR2 = 0; //Duty Cycle op 0% => led uit
            HAL_Delay 2); //1ms wachten
        }
    }
}
void stuurDataIrDuo  uint8_t data_ch1, uint8_t data_ch2  //MSB eerst
{
    uint8_t bit_ch1, bit_ch2;
    for  uint8_t i = 0; i < 8; i++)
    {
        bit_ch1 = data_ch1 & 128;
        bit_ch2 = data_ch2 & 128;
        data_ch1 = data_ch1 <<  1;
        data_ch2 = data_ch2 <<  1;
        if (bit_ch1 == 128 && bit_ch2 == 128)
        {
            htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay 2); //3ms wachten
            htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay 0); //1ms wachten
        }
        else if (bit_ch1 == 128 && bit_ch2 == 0)
        {
            htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay 0); //1ms wachten
            htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay 1); //2ms wachten
            htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay 0); //1ms wachten
        }
        else if (bit_ch1 == 0 && bit_ch2 == 128)
        {
            htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
```

```
129             HAL_Delay 0); //1ms wachten
130             htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
131             HAL_Delay 1); //2ms wachten
132             htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
133             HAL_Delay 0); //1ms wachten
134 
135         else
136 
137             htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
138             htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
139             HAL_Delay 0); //3ms wachten
140             htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
141             htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
142             HAL_Delay 2); //1ms wachten
143 
144     }
145 }
146 void stuurDataIrAll  uint8_t data_ch1, uint8_t data_ch2, uint8_t data_ch3)
147 {
148     uint8_t bit_ch1, bit_ch2, bit_ch3;
149     for  uint8_t i = 0; i < 8; i++)
150     {
151         bit_ch1 = data_ch1 & 128;
152         bit_ch2 = data_ch2 & 128;
153         bit_ch3 = data_ch3 & 128;
154         data_ch1 = data_ch1 <<  1;
155         data_ch2 = data_ch2 <<  1;
156         data_ch3 = data_ch3 <<  1;
157         if  bit_ch1 == 128 && bit_ch2 == 128 && bit_ch3 == 128)
158 
159             htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
160             htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
161             htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
162             HAL_Delay 2); //3ms wachten
163             htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
164             htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
165             htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
166             HAL_Delay 0); //1ms wachten
167 
168         else if  bit_ch1 == 0 && bit_ch2 == 128 && bit_ch3 == 128)
169 
170             htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
171             htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
172             htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
173             HAL_Delay 0); //3ms wachten
174             htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
175             HAL_Delay 1); //1ms wachten
176             htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
177             htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
178             HAL_Delay 0); //1ms wachten
179 
180         else if  bit_ch1 == 128 && bit_ch2 == 0 && bit_ch3 == 128)
181 
182             htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
183             htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
184             htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
185             HAL_Delay 0); //3ms wachten
186             htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
187             HAL_Delay 1); //1ms wachten
188             htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
189             htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
190             HAL_Delay 0); //1ms wachten
191 
192         else if  bit_ch1 == 128 && bit_ch2 == 128 && bit_ch3 == 0)
```

```
193         {
194             htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
195             htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
196             htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
197             HAL_Delay(0); //3ms wachten
198             htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
199             HAL_Delay(1); //1ms wachten
200             htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
201             htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
202             HAL_Delay(0); //1ms wachten
203         }
204         else if(bit_ch1 == 0 && bit_ch2 == 0 && bit_ch3 == 128)
205         {
206             htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
207             htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
208             htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
209             HAL_Delay(0); //3ms wachten
210             htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
211             htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
212             HAL_Delay(1); //1ms wachten
213             htim2.Instance->CCR1 = 0; //Duty Cycle op 0% => led uit
214             HAL_Delay(0); //1ms wachten
215         }
216         else if(bit_ch1 == 128 && bit_ch2 == 0 && bit_ch3 == 0)
217         {
218             htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
219             htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
220             htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
221             HAL_Delay(0); //3ms wachten
222             htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
223             htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
224             HAL_Delay(1); //1ms wachten
225             htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
226             HAL_Delay(0); //1ms wachten
227         }
228
229         else if(bit_ch1 == 0 && bit_ch2 == 128 && bit_ch3 == 0)
230         {
231             htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
232             htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
233             htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
234             HAL_Delay(0); //3ms wachten
235             htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
236             htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
237             HAL_Delay(1); //1ms wachten
238             htim2.Instance->CCR4 = 0; //Duty Cycle op 0% => led uit
239             HAL_Delay(0); //1ms wachten
240         }
241         else
242         {
243             htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
244             htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
245             htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
246             HAL_Delay(0); //3ms wachten
247             htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
248             htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
249             htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
250             HAL_Delay(2); //1ms wachten
251         }
252     }
253 }
254 /* USER CODE END 0 */
255
256 /**
```

```
257   * @brief  The application entry point.
258   * @retval int
259   */
260 int main void
261 {
262
263   /* USER CODE BEGIN 1 */
264
265   /* USER CODE END 1 */
266
267   /* MCU Configuration--------------------------------------------------------*/
268
269   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
270   HAL_Init();
271
272   /* USER CODE BEGIN Init */
273
274   /* USER CODE END Init */
275
276   /* Configure the system clock */
277   SystemClock_Config();
278
279   /* USER CODE BEGIN SysInit */
280
281   /* USER CODE END SysInit */
282
283   /* Initialize all configured peripherals */
284   MX_GPIO_Init();
285   MX_USART1_UART_Init();
286   MX_TIM2_Init();
287   /* USER CODE BEGIN 2 */
288   HAL_GPIO_WritePin(STATUS_GPIO_Port, STATUS_Pin, 1);
289   HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
290   HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);
291   HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3);
292   /* USER CODE END 2 */
293
294   /* Infinite loop */
295   /* USER CODE BEGIN WHILE */
296   while (1)
297   {
298     /* USER CODE END WHILE */
299
300     /* USER CODE BEGIN 3 */
301       stuurDataIrSolo(0xa1);
302       stuurDataIrDuo(0xa4, 0xa8);
303       //stuurDataIrAll(0x0, 0x0, 0x0);
304   }
305   /* USER CODE END 3 */
306 }
307
308 /**
309   * @brief System Clock Configuration
310   * @retval None
311   */
312 void SystemClock_Config void
313 {
314   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
315   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
316   RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
317
318   /** Configure the main internal regulator output voltage
319   */
320   __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
```

```
321
322   /** Initializes the RCC Oscillators according to the specified parameters
323   * in the RCC_OscInitTypeDef structure.
324   */
325   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
326   RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
327   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
328   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
329   {
330     Error_Handler();
331   }
332
333   /** Initializes the CPU, AHB and APB buses clocks
334   */
335   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
336                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
337   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSE;
338   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
339   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
340   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
341
342   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
343   {
344     Error_Handler();
345   }
346   PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART1;
347   PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
348   if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
349   {
350     Error_Handler();
351   }
352 }
353
354 /**
355   * @brief TIM2 Initialization Function
356   * @param None
357   * @retval None
358   */
359 static void MX_TIM2_Init(void)
360 {
361
362   /* USER CODE BEGIN TIM2_Init 0 */
363
364   /* USER CODE END TIM2_Init 0 */
365
366   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
367   TIM_MasterConfigTypeDef sMasterConfig = {0};
368   TIM_OC_InitTypeDef sConfigOC = {0};
369
370   /* USER CODE BEGIN TIM2_Init 1 */
371
372   /* USER CODE END TIM2_Init 1 */
373   htim2.Instance = TIM2;
374   htim2.Init.Prescaler = 23;
375   htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
376   htim2.Init.Period = 25;
377   htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
378   htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
379   if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
380   {
381     Error_Handler();
382   }
383   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
384   if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
```

```
385   {
386     Error_Handler();
387   }
388   if (HAL_TIM_PWM_Init(&htim2) != HAL_OK)
389   {
390     Error_Handler();
391   }
392   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
393   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
394   if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
395   {
396     Error_Handler();
397   }
398   sConfigOC.OCMode = TIM_OCMODE_PWM1;
399   sConfigOC.Pulse = 0;
400   sConfigOC.OCPolarity = TIM_OCPOLARITY_LOW;
401   sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
402   if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
403   {
404     Error_Handler();
405   }
406   if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
407   {
408     Error_Handler();
409   }
410   if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
411   {
412     Error_Handler();
413   }
414   /* USER CODE BEGIN TIM2_Init 2 */
415
416   /* USER CODE END TIM2_Init 2 */
417   HAL_TIM_MspPostInit(&htim2);
418
419 }
420
421 /**
422   * @brief USART1 Initialization Function
423   * @param None
424   * @retval None
425   */
426 static void MX_USART1_UART_Init(void)
427 {
428
429   /* USER CODE BEGIN USART1_Init 0 */
430
431   /* USER CODE END USART1_Init 0 */
432
433   /* USER CODE BEGIN USART1_Init 1 */
434
435   /* USER CODE END USART1_Init 1 */
436   huart1.Instance = USART1;
437   huart1.Init.BaudRate = 115200;
438   huart1.Init.WordLength = UART_WORDLENGTH_8B;
439   huart1.Init.StopBits = UART_STOPBITS_1;
440   huart1.Init.Parity = UART_PARITY_NONE;
441   huart1.Init.Mode = UART_MODE_TX_RX;
442   huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
443   huart1.Init.OverSampling = UART_OVERSAMPLING_16;
444   huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
445   huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
446   if (HAL_UART_Init(&huart1) != HAL_OK)
447   {
448     Error_Handler();
```

```
449   }
450   /* USER CODE BEGIN USART1_Init 2 */
451
452   /* USER CODE END USART1_Init 2 */
453
454 }
455
456 /**
457   * @brief GPIO Initialization Function
458   * @param None
459   * @retval None
460   */
461 static void MX_GPIO_Init(void)
462 {
463   GPIO_InitTypeDef GPIO_InitStruct = {0};
464 /* USER CODE BEGIN MX_GPIO_Init_1 */
465 /* USER CODE END MX_GPIO_Init_1 */
466
467   /* GPIO Ports Clock Enable */
468   __HAL_RCC_GPIOH_CLK_ENABLE();
469   __HAL_RCC_GPIOB_CLK_ENABLE();
470   __HAL_RCC_GPIOA_CLK_ENABLE();
471
472   /*Configure GPIO pin Output Level */
473   HAL_GPIO_WritePin(STATUS_GPIO_Port, STATUS_Pin, GPIO_PIN_RESET);
474
475   /*Configure GPIO pin : STATUS_Pin */
476   GPIO_InitStruct.Pin = STATUS_Pin;
477   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
478   GPIO_InitStruct.Pull = GPIO_NOPULL;
479   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
480   HAL_GPIO_Init(STATUS_GPIO_Port, &GPIO_InitStruct);
481
482 /* USER CODE BEGIN MX_GPIO_Init_2 */
483 /* USER CODE END MX_GPIO_Init_2 */
484 }
485
486 /* USER CODE BEGIN 4 */
487
488 /* USER CODE END 4 */
489
490 /**
491   * @brief  This function is executed in case of error occurrence.
492   * @retval None
493   */
494 void Error_Handler(void)
495 {
496   /* USER CODE BEGIN Error_Handler_Debug */
497   /* User can add his own implementation to report the HAL error return state */
498   __disable_irq();
499   while (1)
500   {
501   }
502   /* USER CODE END Error_Handler_Debug */
503 }
504
505 #ifdef  USE_FULL_ASSERT
506 /**
507   * @brief  Reports the name of the source file and the source line number
508   *         where the assert_param error has occurred.
509   * @param  file: pointer to the source file name
510   * @param  line: assert_param error line source number
511   * @retval None
512   */
```

```
513 void assert_failed(uint8_t *file, uint32_t line)
514 {
515   /* USER CODE BEGIN 6 */
516   /* User can add his own implementation to report the file name and line number,
517      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
518   /* USER CODE END 6 */
519 }
520 #endif /* USE_FULL_ASSERT */
521
```