```
/* 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
            HAL_Delay(0) ; //1ms wachten
            htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(1) ; //2ms wachten
            htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0) ; //1ms wachten
        }
        else
        {
            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) ; //3ms wachten
            htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(2) ; //1ms wachten
        }
    }
}
void stuurDataIrAll (uint8_t data_ch1, uint8_t data_ch2, uint8_t data_ch3)
{
    uint8_t bit_ch1, bit_ch2, bit_ch3;
    for (uint8_t i = 0; i < 8; i++)
    {
        bit_ch1 = data_ch1 & 128;
        bit_ch2 = data_ch2 & 128;
        bit_ch3 = data_ch3 & 128;
        data_ch1 = data_ch1 <<  1;
        data_ch2 = data_ch2 <<  1;
        data_ch3 = data_ch3 <<  1;
        if (bit_ch1 == 128 && bit_ch2 == 128 && bit_ch3 == 128)
        {
            htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(2) ; //3ms wachten
            htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
            htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0) ; //1ms wachten
        }
        else if (bit_ch1 == 0 && bit_ch2 == 128 && bit_ch3 == 128)
        {
            htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0) ; //3ms wachten
            htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(1) ; //1ms wachten
            htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
            htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0) ; //1ms wachten
        }
        else if (bit_ch1 == 128 && bit_ch2 == 0 && bit_ch3 == 128)
        {
            htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0) ; //3ms wachten
            htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(1) ; //1ms wachten
            htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
            htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0) ; //1ms wachten
        }
        else if (bit_ch1 == 128 && bit_ch2 == 128 && bit_ch3 == 0)
        {
            htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0); //3ms wachten
            htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(1); //1ms wachten
            htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
            htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0); //1ms wachten
        }
        else if (bit_ch1 == 0 && bit_ch2 == 0 && bit_ch3 == 128)
        {
            htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0); //3ms wachten
            htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(1); //1ms wachten
            htim2.Instance->CCR1 = 0; //Duty Cycle op 0% => led uit
            HAL_Delay(0); //1ms wachten
        }
        else if (bit_ch1 == 128 && bit_ch2 == 0 && bit_ch3 == 0)
        {
            htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0); //3ms wachten
            htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(1); //1ms wachten
            htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
            HAL_Delay(0); //1ms wachten
        }

        else if (bit_ch1 == 0 && bit_ch2 == 128 && bit_ch3 == 0)
        {
            htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0); //3ms wachten
            htim2.Instance->CCR3 = 0; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(1); //1ms wachten
            htim2.Instance->CCR4 = 0; //Duty Cycle op 0% => led uit
            HAL_Delay(0); //1ms wachten
        }
        else
        {
            htim2.Instance->CCR3 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR4 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = DUTY_VAL; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(0); //3ms wachten
            htim2.Instance->CCR3 = 0; //Duty Cycle op 0% => led uit
            htim2.Instance->CCR4 = 0; //Duty Cycle op 50% => led 38kHz
            htim2.Instance->CCR1 = 0; //Duty Cycle op 50% => led 38kHz
            HAL_Delay(2); //1ms wachten
        }
    }
}
/* USER CODE END 0 */

/**
```

```c
  * @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_USART1_UART_Init();
  MX_TIM2_Init();
  /* USER CODE BEGIN 2 */
  HAL_GPIO_WritePin(STATUS_GPIO_Port, STATUS_Pin, 1);
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
      stuurDataIrSolo(0xa1);
      stuurDataIrDuo(0xa4, 0xa8);
      //stuurDataIrAll(0x0, 0x0, 0x0);
  }
  /* 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};

  /** Configure the main internal regulator output voltage
  */
  __HAL_PWR_VOLTAGESCALING_CONFIG(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_BYPASS;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  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_HSE;
  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_1) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART1;
  PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

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

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 23;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 25;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_LOW;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */

  /* USER CODE END TIM2_Init 2 */
  HAL_TIM_MspPostInit(&htim2);

}

/**
  * @brief USART1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART1_UART_Init(void)
{

  /* USER CODE BEGIN USART1_Init 0 */

  /* USER CODE END USART1_Init 0 */

  /* USER CODE BEGIN USART1_Init 1 */

  /* USER CODE END USART1_Init 1 */
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART1_Init 2 */

  /* USER CODE END USART1_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(STATUS_GPIO_Port, STATUS_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : STATUS_Pin */
  GPIO_InitStruct.Pin = STATUS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(STATUS_GPIO_Port, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* 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 */
  __disable_irq();
  while (1)
  {
  }
  /* 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,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

```