```c
/* USER CODE BEGIN Header */
  * @file           : main.c
/* 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 */
#define t 1000
/* USER CODE END PD */

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

/* USER CODE END PM */

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

TIM_HandleTypeDef htim2;

UART_HandleTypeDef huart1;

/* USER CODE BEGIN PV */
uint8_t stand[8][8] = {
        {02, 03, 04, 05, 06, 04, 03, 02},
        {01, 01, 01, 01, 01, 01, 01, 01},
        {00, 00, 00, 00, 00, 00, 00, 00},
        {00, 00, 00, 00, 00, 00, 00, 00},
        {00, 00, 00, 00, 00, 00, 00, 00},
        {00, 00, 00, 00, 00, 00, 00, 00},
        {10, 10, 10, 10, 10, 10, 10, 10},
        {20, 30, 40, 50, 60, 40, 30, 20}
};
uint8_t Bstand[8][8] = {
        {02, 03, 04, 05, 06, 04, 03, 02},
        {01, 01, 01, 01, 01, 01, 01, 01},
        {00, 00, 00, 00, 00, 00, 00, 00},
        {00, 00, 00, 00, 00, 00, 00, 00},
        {00, 00, 00, 00, 00, 00, 00, 00},
        {00, 00, 00, 00, 00, 00, 00, 00},
        {10, 10, 10, 10, 10, 10, 10, 10},
        {20, 30, 40, 50, 60, 40, 30, 20}
};
uint8_t buf[8][8] = {
        {00, 00, 00, 00, 00, 00, 00, 01},
        {01, 01, 01, 01, 01, 01, 01, 01},
        {00, 00, 00, 00, 00, 00, 00, 00},
```

```
 74         {00, 00, 00, 00, 00, 00, 00, 00},
 75         {00, 00, 00, 00, 00, 00, 00, 00},
 76         {00, 00, 00, 00, 00, 00, 00, 00},
 77         {01, 01, 01, 01, 01, 01, 01, 01},
 78         {01, 01, 01, 01, 01, 01, 01, 01}
 79 };
 80 uint8_t bufV[8][8] = {
 81         {00, 00, 00, 00, 00, 00, 00, 01},
 82         {01, 01, 01, 01, 01, 01, 01, 01},
 83         {00, 00, 00, 00, 00, 00, 00, 00},
 84         {00, 00, 00, 00, 00, 00, 00, 00},
 85         {00, 00, 00, 00, 00, 00, 00, 00},
 86         {00, 00, 00, 00, 00, 00, 00, 00},
 87         {01, 01, 01, 01, 01, 01, 01, 01},
 88         {01, 01, 01, 01, 01, 01, 01, 01}
 89 };
 90 uint8_t coor[2];
 91 uint8_t i = 0;
 92 uint8_t j = 0;
 93 uint8_t weg = 0;
 94 /* USER CODE END PV */
 95
 96 /* Private function prototypes -----------------------------------------------*/
 97 void SystemClock_Config(void);
 98 static void MX_GPIO_Init(void);
 99 static void MX_TIM2_Init(void);
100 static void MX_USART1_UART_Init(void);
101 /* USER CODE BEGIN PFP */
102
103 /* USER CODE END PFP */
104
105 /* Private user code ---------------------------------------------------------*/
106 /* USER CODE BEGIN 0 */
107 #include <errno.h>
108 #include <sys/stat.h>
109 #include <sys/times.h>
110 #include <sys/unistd.h>
111 int _write(int file, char *ptr, int len) {
112     HAL_StatusTypeDef xStatus;
113     switch (file) {
114     case STDOUT_FILENO: /*stdout*/
115         xStatus = HAL_UART_Transmit(&huart1, (uint8_t*)ptr, len, HAL_MAX_DELAY);
116         if (xStatus != HAL_OK) {
117             errno = EIO;
118             return -1;
119         }
120         break;
121     case STDERR_FILENO: /* stderr */
122         xStatus = HAL_UART_Transmit(&huart1, (uint8_t*)ptr, len, HAL_MAX_DELAY);
123         if (xStatus != HAL_OK) {
124             errno = EIO;
125             return -1;
126         }
127         break;
128     default:
129         errno = EBADF;
130         return -1;
```

```
131     }
132     return len;
133 }
134 
135 void SysTickDelayCount2(unsigned long ulCount){
136     CoreDebug->DEMCR |= CoreDebug_DEMCR_TRCENA_Msk;
137     DWT->LAR = 0xC5ACCE55;
138     DWT->CYCCNT = 0;
139     DWT->CTRL |= DWT_CTRL_CYCCNTENA_Msk;
140 
141     while(DWT->CYCCNT < ulCount);
142 }
143 
144 void Meten(){
145     HAL_GPIO_WritePin(SH_Port,SH_Pin,GPIO_PIN_RESET);
146     SysTickDelayCount2(t);
147     HAL_GPIO_WritePin(SH_Port,SH_Pin,GPIO_PIN_SET);
148     for (j = 0; j < 1; j++){
149         for (i = 0; i < 8; i++){
150             SysTickDelayCount2((t/2));
151             buf[j][i] = (HAL_GPIO_ReadPin(SPI_MISO_GPIO_Port,SPI_MISO_Pin) ^ 1); //Inverteer
    de lezing
152             SysTickDelayCount2((t/2));
153             HAL_GPIO_WritePin(SPI_CLK_GPIO_Port,SPI_CLK_Pin,GPIO_PIN_SET);
154             SysTickDelayCount2(t);
155             HAL_GPIO_WritePin(SPI_CLK_GPIO_Port,SPI_CLK_Pin,GPIO_PIN_RESET);
156         }
157     }
158 }
159 
160 void Controle(){
161     for (j = 0; j < 8; j++){
162         for (i = 0; i < 8; i++){
163             if (bufV[j][i] != buf[j][i]){
164                 coor[0] = j;
165                 coor[1] = i;
166                 Bstand[j][i] = 0;
167                 weg = 1;
168                 printf("Verandering op positie: %d %d \n\r", coor[0], coor[1]);
169             } else {
170                 Bstand[j][i] = stand[j][i];
171             }
172         }
173     }
174 }
175 /* USER CODE END 0 */
176 
177 /**
178   * @brief  The application entry point.
179   * @retval int
180   */
181 int main(void)
182 {
183   /* USER CODE BEGIN 1 */
184 
185   /* USER CODE END 1 */
186 
```

```
187  /* MCU Configuration--------------------------------------------------------*/
188
189  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
190  HAL_Init();
191
192  /* USER CODE BEGIN Init */
193
194  /* USER CODE END Init */
195
196  /* Configure the system clock */
197  SystemClock_Config();
198
199  /* USER CODE BEGIN SysInit */
200
201  /* USER CODE END SysInit */
202
203  /* Initialize all configured peripherals */
204  MX_GPIO_Init();
205  MX_TIM2_Init();
206  MX_USART1_UART_Init();
207  /* USER CODE BEGIN 2 */
208  /* USER CODE END 2 */
209
210  /* Infinite loop */
211  /* USER CODE BEGIN WHILE */
212  while (1)
213  {
214    /* USER CODE END WHILE */
215
216    /* USER CODE BEGIN 3 */
217        Meten();
218        for (j = 0; j < 8; j++){
219            printf("Matrix %d: ", j);
220            for (i = 0; i < 8; i++){
221                printf("%02d ", buf[j][i]);
222            }
223            printf("\n\r");
224        }
225        printf("\n\r");
226        for (j = 0; j < 8; j++){
227            printf("Matrix %d: ", j);
228            for (i = 0; i < 8; i++){
229                printf("%02d ", Bstand[j][i]);
230            }
231            printf("\n\r");
232        }
233        printf("\n\r");
234
235        Controle();
236
237        for (j = 0; j < 8; j++){
238            for (i = 0; i < 8; i++){
239                bufV[j][i] = buf[j][i];
240            }
241        }
242
243        HAL_Delay(1000);
```

```
244
245   }
246   /* USER CODE END 3 */
247 }
248
249 /**
250   * @brief System Clock Configuration
251   * @retval None
252   */
253 void SystemClock_Config(void)
254 {
255   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
256   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
257   RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
258
259   /** Configure LSE Drive Capability
260   */
261   HAL_PWR_EnableBkUpAccess();
262   /** Configure the main internal regulator output voltage
263   */
264   __HAL_RCC_PWR_CLK_ENABLE();
265   __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
266   /** Initializes the RCC Oscillators according to the specified parameters
267   * in the RCC_OscInitTypeDef structure.
268   */
269   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
270   RCC_OscInitStruct.HSIState = RCC_HSI_ON;
271   RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
272   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
273   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
274   RCC_OscInitStruct.PLL.PLLM = 8;
275   RCC_OscInitStruct.PLL.PLLN = 200;
276   RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
277   RCC_OscInitStruct.PLL.PLLQ = 2;
278   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
279   {
280     Error_Handler();
281   }
282   /** Activate the Over-Drive mode
283   */
284   if (HAL_PWREx_EnableOverDrive() != HAL_OK)
285   {
286     Error_Handler();
287   }
288   /** Initializes the CPU, AHB and APB buses clocks
289   */
290   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
291                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
292   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
293   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
294   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
295   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
296
297   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_6) != HAL_OK)
298   {
299     Error_Handler();
300   }
```

```
301   PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_USART1;
302   PeriphClkInitStruct.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
303   if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
304   {
305     Error_Handler();
306   }
307 }
308 
309 /**
310   * @brief TIM2 Initialization Function
311   * @param None
312   * @retval None
313   */
314 static void MX_TIM2_Init(void)
315 {
316 
317   /* USER CODE BEGIN TIM2_Init 0 */
318 
319   /* USER CODE END TIM2_Init 0 */
320 
321   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
322   TIM_MasterConfigTypeDef sMasterConfig = {0};
323 
324   /* USER CODE BEGIN TIM2_Init 1 */
325 
326   /* USER CODE END TIM2_Init 1 */
327   htim2.Instance = TIM2;
328   htim2.Init.Prescaler = 0;
329   htim2.Init.CounterMode = TIM_COUNTERMODE_DOWN;
330   htim2.Init.Period = 650;
331   htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
332   htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
333   if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
334   {
335     Error_Handler();
336   }
337   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
338   if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
339   {
340     Error_Handler();
341   }
342   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
343   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
344   if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
345   {
346     Error_Handler();
347   }
348   /* USER CODE BEGIN TIM2_Init 2 */
349 
350   /* USER CODE END TIM2_Init 2 */
351 
352 }
353 
354 /**
355   * @brief USART1 Initialization Function
356   * @param None
357   * @retval None
```

```c
  */
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};

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

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(SPI_CLK_PIN_GPIO_Port, SPI_CLK_PIN_Pin, GPIO_PIN_SET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(SH_GPIO_Port, SH_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : SPI_CLK_PIN_Pin */
  GPIO_InitStruct.Pin = SPI_CLK_PIN_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
```

```
415   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
416   HAL_GPIO_Init(SPI_CLK_PIN_GPIO_Port, &GPIO_InitStruct);
417
418   /*Configure GPIO pin : SPI_MISO_PIN_Pin */
419   GPIO_InitStruct.Pin = SPI_MISO_PIN_Pin;
420   GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
421   GPIO_InitStruct.Pull = GPIO_NOPULL;
422   HAL_GPIO_Init(SPI_MISO_PIN_GPIO_Port, &GPIO_InitStruct);
423
424   /*Configure GPIO pin : SH_Pin */
425   GPIO_InitStruct.Pin = SH_Pin;
426   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
427   GPIO_InitStruct.Pull = GPIO_PULLUP;
428   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
429   HAL_GPIO_Init(SH_GPIO_Port, &GPIO_InitStruct);
430
431 }
432
433 /* USER CODE BEGIN 4 */
434
435 /* USER CODE END 4 */
436
437 /**
438   * @brief  This function is executed in case of error occurrence.
439   * @retval None
440   */
441 void Error_Handler(void)
442 {
443   /* USER CODE BEGIN Error_Handler_Debug */
444   /* User can add his own implementation to report the HAL error return state */
445   __disable_irq();
446   while (1)
447   {
448   }
449   /* USER CODE END Error_Handler_Debug */
450 }
451
452 #ifdef  USE_FULL_ASSERT
453 /**
454   * @brief  Reports the name of the source file and the source line number
455   *         where the assert_param error has occurred.
456   * @param  file: pointer to the source file name
457   * @param  line: assert_param error line source number
458   * @retval None
459   */
460 void assert_failed(uint8_t *file, uint32_t line)
461 {
462   /* USER CODE BEGIN 6 */
463   /* User can add his own implementation to report the file name and line number,
464      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
465   /* USER CODE END 6 */
466 }
467 #endif /* USE_FULL_ASSERT */
468
469 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
470
```