```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 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"
#include "stdlib.h"
#include "string.h"
#include "stdbool.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

// Estado do Cursor
#define CURSOR_OFF 0x0C   // Apagado
#define CURSOR_ON 0x0E    // Ligado
#define CURSOR_BLINK 0x0F // Piscante

// Estado dos pinos de Controle...
#define RS_0 GPIOA -> BRR =  1<<9 //PA9
#define RS_1 GPIOA -> BSRR = 1<<9 //PA9
#define EN_0 GPIOC -> BRR =  1<<7 //PC7
#define EN_1 GPIOC -> BSRR = 1<<7 //PC7

// Estado dos pinos do Barramento do LCD...
#define D7_0 GPIOA -> BRR  = 1<<8 //PA8
#define D7_1 GPIOA -> BSRR = 1<<8 //PA8

#define D6_0 GPIOB -> BRR  = 1<<10 //PB10
#define D6_1 GPIOB -> BSRR = 1<<10 //PB10

#define D5_0 GPIOB -> BRR  = 1<<4 //PB4
#define D5_1 GPIOB -> BSRR = 1<<4 //PB4

#define D4_0 GPIOB -> BRR  = 1<<5 //PB5
#define D4_1 GPIOB -> BSRR = 1<<5 //PB5

// Para usarmos o terminal
#define NO_LCD 1
#define NA_SERIAL 2

// Para usarmos o DHT22
#define DHT22_PORT GPIOA
#define DHT22_PIN GPIO_PIN_1
/* USER CODE END PD */
```

```
70
71  /* Private macro -------------------------------------------------------------*/
72  /* USER CODE BEGIN PM */
73
74  /* USER CODE END PM */
75
76  /* Private variables ---------------------------------------------------------*/
77  RTC_HandleTypeDef hrtc;
78
79  TIM_HandleTypeDef htim1;
80
81  UART_HandleTypeDef huart2;
82
83  /* USER CODE BEGIN PV */
84
85  /* USER CODE END PV */
86
87  /* Private function prototypes -----------------------------------------------*/
88  void SystemClock_Config(void);
89  static void MX_GPIO_Init(void);
90  static void MX_USART2_UART_Init(void);
91  static void MX_RTC_Init(void);
92  static void MX_TIM1_Init(void);
93  /* USER CODE BEGIN PFP */
94  void udelay(void);
95  void delayus(int tempo);
96  void lcd_wrcom4 (uint8_t com4);
97  void lcd_wrcom(uint8_t com);
98  void lcd_wrchar(char ch);
99  void lcd_init(uint8_t cursor);
100 void lcd_wrstr(char *str);
101 void lcd_wr2dig(uint8_t valor);
102 void lcd_senddata(uint8_t data);
103 void lcd_clear(void);
104 void lcd_progchar(uint8_t n);
105 void lcd_goto(uint8_t x, uint8_t y);
106 int __io_putschar(int ch);
107 int fputc(int ch, FILE * f);
108 //--------------------------------DHT----------------------------
109 //void udelay2(void);
110 //void delay(int tempo);
111 void delay(uint16_t us);
112 void Set_Pin_Output(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin);
113 void Set_Pin_Input(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin);
114 void DHT22_init(void);
115 void DHT22_Start(void);
116 uint8_t DHT22_Check_Response(void);
117 uint8_t DHT22_Read(void);
118 int DHT22(float *Temperature, float *Humidity);
119 /* USER CODE END PFP */
120
121 /* Private user code ---------------------------------------------------------*/
122 /* USER CODE BEGIN 0 */
123 HAL_StatusTypeDef o;
124 char AONDE=NO_LCD;
125 // ----------- Variaveis globais --------------------
126
127 int __io_putchar(int ch){
128     if (AONDE == NO_LCD){
129         if (ch != '\n') lcd_wrchar(ch);
130     }
131     if (AONDE == NA_SERIAL){
132         HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 100);
133     }
134     return ch;
135 }
136 //--------------------------------DHT----------------------------
137 void delay(uint16_t us){
138     uint16_t start = __HAL_TIM_GET_COUNTER(&htim1); // Lê o valor atual do contador
```

```
139         while ((__HAL_TIM_GET_COUNTER(&htim1) - start) < us); // Aguarda até que a diferença
            atinja 'us'
140     }
141
142     void Set_Pin_Output(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin){
143         GPIO_InitTypeDef GPIO_InitStruct = {0};
144         GPIO_InitStruct.Pin = GPIO_Pin;
145         GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
146         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
147         GPIO_InitStruct.Pull = GPIO_NOPULL;
148         HAL_GPIO_Init(GPIOx, &GPIO_InitStruct);
149     }
150     void Set_Pin_Input(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin){
151         GPIO_InitTypeDef GPIO_InitStruct = {0};
152         GPIO_InitStruct.Pin = GPIO_Pin;
153         GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
154         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
155         GPIO_InitStruct.Pull = GPIO_PULLUP;
156         HAL_GPIO_Init(GPIOx, &GPIO_InitStruct);
157     }
158
159     void DHT22_init(void){
160         GPIO_InitTypeDef GPIO_InitStruct = {0};
161         HAL_GPIO_DeInit(DHT22_PORT, DHT22_PIN);
162         GPIO_InitStruct.Pin = DHT22_PIN;
163         GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
164         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
165         GPIO_InitStruct.Pull = GPIO_NOPULL;
166         HAL_GPIO_Init(DHT22_PORT, &GPIO_InitStruct);
167         HAL_Delay(1000);
168         HAL_GPIO_WritePin(DHT22_PORT, DHT22_PIN, GPIO_PIN_SET);// pull the pin high
169     }
170     void DHT22_Start(void){
171         Set_Pin_Output(DHT22_PORT, DHT22_PIN);// set the pin as output
172         HAL_GPIO_WritePin(DHT22_PORT, DHT22_PIN, 0);// pull the pin low
173         delay(1500);// wait for > 1ms
174         HAL_GPIO_WritePin(DHT22_PORT, DHT22_PIN, 1);// pull the pin high
175         delay(30);   // wait for 30us
176         Set_Pin_Input(DHT22_PORT, DHT22_PIN);// set as input
177     }
178
179     uint8_t DHT22_Check_Response(void){
180         uint8_t Response = 0;
181     //  AONDE=NO_LCD;
182     //  lcd_goto(0,0);
183     //  printf("4\n");
184         delay(40);  // wait for 40us
185     //  printf("5\n");
186         if (!(HAL_GPIO_ReadPin (DHT22_PORT, DHT22_PIN))){// if the pin is low
187     //      printf("6\n");
188             delay(80);// wait for 80us
189     //      printf("7\n");
190             if ((HAL_GPIO_ReadPin (DHT22_PORT, DHT22_PIN))) Response = 1;// if the pin is
                high, response is ok
191             else Response = -1;
192     //          printf("8\n");
193         }
194
195     //  pMillis = HAL_GetTick();
196     //  cMillis = HAL_GetTick();
197     //  printf("9\n");
198         while((HAL_GPIO_ReadPin(DHT22_PORT, DHT22_PIN)));// wait for the pin to go low
199     //      cMillis = HAL_GetTick();
200     //  printf("10\n");
201         return Response;
202     }
203     uint8_t DHT22_Read(void){
204         uint8_t i,j;
205     //  AONDE=NO_LCD;
```

```
206   //  lcd_goto(0,0);
207       for (j=0;j<8;j++){
208           //pMillis = HAL_GetTick();
209           //cMillis = HAL_GetTick();
210   //      printf("3\n");
211           while(!(HAL_GPIO_ReadPin(DHT22_PORT, DHT22_PIN)));// wait for the pin to go high
212   //      printf("2\n");  //cMillis = HAL_GetTick();
213           delay(30);// wait for 40 us
214   //      printf("3\n");
215           if (!(HAL_GPIO_ReadPin(DHT22_PORT, DHT22_PIN))) i&= ~(1<<(7-j));// if the pin is
          lOW(write 0)
216           else i|= (1<<(7-j));// if the pin is high, write 1
217   //      printf("4\n");
218           while((HAL_GPIO_ReadPin(DHT22_PORT, DHT22_PIN)));// wait for the pin to go low
219               //cMillis = HAL_GetTick();
220   //      printf("5\n");
221       }
222   //  printf("6\n");
223       return i;
224   }
225
226   //int DHT22(float *Temperature, float *Humidity, uint8_t *t1, uint8_t *t2, uint8_t *r1,
      uint8_t *r2, uint16_t *t, uint16_t *r){
227   int DHT22(float *Temperature, float *Humidity){
228       uint8_t Rh_byte1, Rh_byte2, Temp_byte1, Temp_byte2;
229       uint16_t SUM, CHECK, TEMP, RH;
230       //float Temperature, Humidity;// Debug
231       ////////////////////////////////////////////////////////////////////////
232       HAL_GPIO_WritePin(DHT22_PORT, DHT22_PIN, GPIO_PIN_RESET);
233       HAL_GPIO_WritePin(DHT22_PORT, DHT22_PIN, GPIO_PIN_SET);
234       ////////////////////////////////////////////////////////////////////////
235       AONDE = NO_LCD;
236       lcd_goto(0,0);
237       //printf("1\n");
238       DHT22_Start();
239       //printf("2\n");
240       if(DHT22_Check_Response()){
241           //printf("3\n");
242           Rh_byte1 = DHT22_Read();
243           //printf("4\n");
244           Rh_byte2 = DHT22_Read();
245           //printf("5\n");
246           Temp_byte1 = DHT22_Read();
247           //printf("6\n");
248           Temp_byte2 = DHT22_Read();
249           //printf("7\n");
250           SUM = DHT22_Read();
251           //printf("8\n");
252           CHECK = Rh_byte1 + Rh_byte2 +  Temp_byte1 + Temp_byte2;
253           if (CHECK == SUM){
254               //printf("9\n");
255               TEMP=((Temp_byte1 << 7 ) | Temp_byte2);
256               RH =((Rh_byte1 << 8 ) | Rh_byte2);
257               *Temperature=(float)TEMP/(10.0);
258               *Humidity=(float)RH/(10.0);
259               return 1;
260           }
261       }
262       return 0;
263   }
264   /* USER CODE END 0 */
265
266   /**
267     * @brief  The application entry point.
268     * @retval int
269     */
270   int main(void)
271   {
272     /* USER CODE BEGIN 1 */
```

```
  float Temp = 0, Humi = 0;
  int x;
/* 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_USART2_UART_Init();
MX_RTC_Init();
MX_TIM1_Init();
/* USER CODE BEGIN 2 */
HAL_UART_Init(&huart2);
  HAL_RTC_Init(&hrtc);
  HAL_RTC_WaitForSynchro(&hrtc);
  HAL_TIM_Base_Init(&htim1);
  HAL_TIM_Base_Start(&htim1);
  lcd_init(CURSOR_OFF);
  lcd_clear();
  DHT22_init();
  HAL_Delay(1000);

  AONDE = NA_SERIAL;
  printf("\r Estou imprimindo na serial \n\r");
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
      x = DHT22(&Temp, &Humi);
      AONDE = NO_LCD;
      lcd_goto(0,0);
      printf("%.1f%cC\n", Temp, 223);
      lcd_goto(1,0);
      printf("RH %.2f\n", Humi);
      if(x ==0)DHT22_init();
//  if(DHT22(&Temp, &Humi)){
//      AONDE = NO_LCD;
//      lcd_goto(0,0);
//      printf("%.1f%cC\n", Temp, 223);
//      lcd_goto(1,0);
//      printf("RH %.2f\n", Humi);
//  }
//  else {
//      DHT22_init();

      //printf("\rInicializando novamente\n\r");
//  }
    HAL_Delay(2000);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

```
342   }
343
344   /**
345     * @brief System Clock Configuration
346     * @retval None
347     */
348   void SystemClock_Config(void)
349   {
350     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
351     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
352     RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
353
354     /** Initializes the RCC Oscillators according to the specified parameters
355     * in the RCC_OscInitTypeDef structure.
356     */
357     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI48|RCC_OSCILLATORTYPE_LSI;
358     RCC_OscInitStruct.HSI48State = RCC_HSI48_ON;
359     RCC_OscInitStruct.LSIState = RCC_LSI_ON;
360     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
361     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
362     {
363       Error_Handler();
364     }
365
366     /** Initializes the CPU, AHB and APB buses clocks
367     */
368     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
369                                   |RCC_CLOCKTYPE_PCLK1;
370     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI48;
371     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
372     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
373
374     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
375     {
376       Error_Handler();
377     }
378     PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART2|RCC_PERIPHCLK_RTC;
379     PeriphClkInit.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
380     PeriphClkInit.RTCClockSelection = RCC_RTCCLKSOURCE_LSI;
381     if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
382     {
383       Error_Handler();
384     }
385   }
386
387   /**
388     * @brief RTC Initialization Function
389     * @param None
390     * @retval None
391     */
392   static void MX_RTC_Init(void)
393   {
394
395     /* USER CODE BEGIN RTC_Init 0 */
396
397     /* USER CODE END RTC_Init 0 */
398
399     /* USER CODE BEGIN RTC_Init 1 */
400
401     /* USER CODE END RTC_Init 1 */
402
403     /** Initialize RTC Only
404     */
405     hrtc.Instance = RTC;
406     hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
407     hrtc.Init.AsynchPrediv = 127;
408     hrtc.Init.SynchPrediv = 255;
409     hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
410     hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
```

```
411     hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
412     if (HAL_RTC_Init(&hrtc) != HAL_OK)
413     {
414       Error_Handler();
415     }
416     /* USER CODE BEGIN RTC_Init 2 */
417
418     /* USER CODE END RTC_Init 2 */
419
420   }
421
422   /**
423     * @brief TIM1 Initialization Function
424     * @param None
425     * @retval None
426     */
427   static void MX_TIM1_Init(void)
428   {
429
430     /* USER CODE BEGIN TIM1_Init 0 */
431
432     /* USER CODE END TIM1_Init 0 */
433
434     TIM_ClockConfigTypeDef sClockSourceConfig = {0};
435     TIM_MasterConfigTypeDef sMasterConfig = {0};
436
437     /* USER CODE BEGIN TIM1_Init 1 */
438
439     /* USER CODE END TIM1_Init 1 */
440     htim1.Instance = TIM1;
441     htim1.Init.Prescaler = 47;
442     htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
443     htim1.Init.Period = 65535;
444     htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
445     htim1.Init.RepetitionCounter = 0;
446     htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
447     if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
448     {
449       Error_Handler();
450     }
451     sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
452     if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
453     {
454       Error_Handler();
455     }
456     sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
457     sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
458     if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
459     {
460       Error_Handler();
461     }
462     /* USER CODE BEGIN TIM1_Init 2 */
463
464     /* USER CODE END TIM1_Init 2 */
465
466   }
467
468   /**
469     * @brief USART2 Initialization Function
470     * @param None
471     * @retval None
472     */
473   static void MX_USART2_UART_Init(void)
474   {
475
476     /* USER CODE BEGIN USART2_Init 0 */
477
478     /* USER CODE END USART2_Init 0 */
479
```

```
480     /* USER CODE BEGIN USART2_Init 1 */
481
482     /* USER CODE END USART2_Init 1 */
483     huart2.Instance = USART2;
484     huart2.Init.BaudRate = 9600;
485     huart2.Init.WordLength = UART_WORDLENGTH_8B;
486     huart2.Init.StopBits = UART_STOPBITS_1;
487     huart2.Init.Parity = UART_PARITY_NONE;
488     huart2.Init.Mode = UART_MODE_TX_RX;
489     huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
490     huart2.Init.OverSampling = UART_OVERSAMPLING_16;
491     huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
492     huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
493     if (HAL_UART_Init(&huart2) != HAL_OK)
494     {
495       Error_Handler();
496     }
497     /* USER CODE BEGIN USART2_Init 2 */
498
499     /* USER CODE END USART2_Init 2 */
500
501   }
502
503   /**
504     * @brief GPIO Initialization Function
505     * @param None
506     * @retval None
507     */
508   static void MX_GPIO_Init(void)
509   {
510     GPIO_InitTypeDef GPIO_InitStruct = {0};
511   /* USER CODE BEGIN MX_GPIO_Init_1 */
512   /* USER CODE END MX_GPIO_Init_1 */
513
514     /* GPIO Ports Clock Enable */
515     __HAL_RCC_GPIOC_CLK_ENABLE();
516     __HAL_RCC_GPIOF_CLK_ENABLE();
517     __HAL_RCC_GPIOA_CLK_ENABLE();
518     __HAL_RCC_GPIOB_CLK_ENABLE();
519
520     /*Configure GPIO pin Output Level */
521     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1|LD2_Pin|GPIO_PIN_8|GPIO_PIN_9, GPIO_PIN_RESET);
522
523     /*Configure GPIO pin Output Level */
524     HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10|GPIO_PIN_4|GPIO_PIN_5, GPIO_PIN_RESET);
525
526     /*Configure GPIO pin Output Level */
527     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET);
528
529     /*Configure GPIO pin : B1_Pin */
530     GPIO_InitStruct.Pin = B1_Pin;
531     GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
532     GPIO_InitStruct.Pull = GPIO_NOPULL;
533     HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
534
535     /*Configure GPIO pins : PA1 LD2_Pin PA8 PA9 */
536     GPIO_InitStruct.Pin = GPIO_PIN_1|LD2_Pin|GPIO_PIN_8|GPIO_PIN_9;
537     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
538     GPIO_InitStruct.Pull = GPIO_NOPULL;
539     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
540     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
541
542     /*Configure GPIO pins : PB10 PB4 PB5 */
543     GPIO_InitStruct.Pin = GPIO_PIN_10|GPIO_PIN_4|GPIO_PIN_5;
544     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
545     GPIO_InitStruct.Pull = GPIO_NOPULL;
546     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
547     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
548
```

```
549     /*Configure GPIO pin : PC7 */
550     GPIO_InitStruct.Pin = GPIO_PIN_7;
551     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
552     GPIO_InitStruct.Pull = GPIO_NOPULL;
553     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
554     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
555
556   /* USER CODE BEGIN MX_GPIO_Init_2 */
557   /* USER CODE END MX_GPIO_Init_2 */
558   }
559
560   /* USER CODE BEGIN 4 */
561
562   //---------------------------------LCD----------------------------
563   void lcd_backlight (uint8_t light){
564       if(light == 0){
565           GPIOB -> BRR = 1<<3;
566       } else {
567           GPIOB -> BSRR= 1<<3;
568       }
569   }
570   void lcd_init(uint8_t cursor){
571       lcd_wrcom4(3);
572       lcd_wrcom4(3);
573       lcd_wrcom4(3);
574       lcd_wrcom4(2);
575       lcd_wrcom(0x28);
576       lcd_wrcom(cursor);
577       lcd_wrcom(0x06);
578       lcd_wrcom(0x01);
579   }
580   void lcd_wrcom4(uint8_t com4){
581       lcd_senddata(com4); //D4...d0
582       RS_0;
583       EN_1;
584       delayus(5);
585       EN_0;
586       HAL_Delay(5);
587   }
588   void lcd_wrcom(uint8_t com){
589       lcd_senddata(com>>4); //0000D7...D4
590       RS_0;
591       EN_1;
592       delayus(5);
593       EN_0;
594       delayus(5);
595
596       lcd_senddata(com & 0x0F); //0000D3...d0
597       EN_1;
598       delayus(5);
599       EN_0;
600       HAL_Delay(5);
601   }
602   void lcd_clear(void){
603       lcd_wrcom(0x01);
604   }
605   //goto para 16x2
606   void lcd_goto(uint8_t x, uint8_t y){
607       uint8_t com = 0x80;
608       if (x==0 && y<16) com = 0x80 + y;
609       else if (x==1 && y<16) com = 0xC0 + y;
610       else com = 0x80;
611       lcd_wrcom(com);
612   }
613   void lcd_wrchar(char ch){
614       lcd_senddata(ch>>4); //D7...D4
615       RS_1;
616       EN_1;
617       delayus(5);
```

```
618       EN_0;
619       delayus(5);
620
621       lcd_senddata(ch & 0x0F); //D3...D0
622       RS_1;
623       EN_1;
624       delayus(5);
625       EN_0;
626       HAL_Delay(5);
627   }
628
629   void lcd_wrstr(char *str){
630       while(*str) lcd_wrchar(*str++);
631   }
632
633
634   void udelay(void){
635       int tempo = 7;
636       while(tempo--);
637   }
638
639   void delayus(int tempo){
640       while(tempo--) udelay();
641   }
642
643   void lcd_wr2dig(uint8_t valor){
644       lcd_wrchar(valor/10 + '0'); // ou +48 -> dezena
645       lcd_wrchar(valor%10 + '0'); // ou +48 -> unidade
646   }
647
648   void lcd_senddata(uint8_t data){
649       if((data & (1<<3))==0) D7_0; else D7_1;
650       if((data & (1<<2))==0) D6_0; else D6_1;
651       if((data & (1<<1))==0) D5_0; else D5_1;
652       if((data & (1<<0))==0) D4_0; else D4_1;
653   }
654   //-----------------------------------------------------------------
655
656   /* USER CODE END 4 */
657
658   /**
659     * @brief  This function is executed in case of error occurrence.
660     * @retval None
661     */
662   void Error_Handler(void)
663   {
664     /* USER CODE BEGIN Error_Handler_Debug */
665     /* User can add his own implementation to report the HAL error return state */
666     __disable_irq();
667     while (1)
668     {
669     }
670     /* USER CODE END Error_Handler_Debug */
671   }
672
673   #ifdef  USE_FULL_ASSERT
674   /**
675     * @brief  Reports the name of the source file and the source line number
676     *         where the assert_param error has occurred.
677     * @param  file: pointer to the source file name
678     * @param  line: assert_param error line source number
679     * @retval None
680     */
681   void assert_failed(uint8_t *file, uint32_t line)
682   {
683     /* USER CODE BEGIN 6 */
684     /* User can add his own implementation to report the file name and line number,
685        ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
686     /* USER CODE END 6 */
```

```
687   }
688   #endif /* USE_FULL_ASSERT */
689
```