```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 */

// 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
//RELES
#define BUZZER_ON   HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, 1)
#define BUZZER_OFF  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, 0)
// ------ Controle ----------
volatile int corrente = 2;

/* USER CODE END PTD */

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

```
70
71  /* USER CODE END PD */
72
73  /* Private macro -------------------------------------------------------------*/
74  /* USER CODE BEGIN PM */
75
76  /* USER CODE END PM */
77
78  /* Private variables ---------------------------------------------------------*/
79  RTC_HandleTypeDef hrtc;
80
81  TIM_HandleTypeDef htim1;
82
83  UART_HandleTypeDef huart2;
84
85  /* USER CODE BEGIN PV */
86
87  /* USER CODE END PV */
88
89  /* Private function prototypes -----------------------------------------------*/
90  void SystemClock_Config(void);
91  static void MX_GPIO_Init(void);
92  static void MX_USART2_UART_Init(void);
93  static void MX_RTC_Init(void);
94  static void MX_TIM1_Init(void);
95  /* USER CODE BEGIN PFP */
96  void udelay(void);
97  void delayus(int tempo);
98  void lcd_wrcom4 (uint8_t com4);
99  void lcd_wrcom(uint8_t com);
100 void lcd_wrchar(char ch);
101 void lcd_init(uint8_t cursor);
102 void lcd_wrstr(char *str);
103 void lcd_wr2dig(uint8_t valor);
104 void lcd_senddata(uint8_t data);
105 void lcd_clear(void);
106 void lcd_progchar(uint8_t n);
107 void lcd_goto(uint8_t x, uint8_t y);
108 int __io_putschar(int ch);
109 int fputc(int ch, FILE * f);
110 //-------------Ajuste e funcionamento do alarme-------------//
111 void ajuste_hora(void);
112 void ajuste_data(void);
113 void mostrahoras(int x);
114 //-------------Ajuste e funcionamento do alarme-------------//
115 void desliga(void);
116 void horas_alarme(void);
117 void alarme(void);
118 void acinona_se(void);
119 void ajuste_hora_inicial(void);
120 void ajuste_hora_final(void);
121 void delay(uint16_t us);
122 //---------------------------
123 void save_pw_alarme(void);
124 void save_pw_in(void);
125 //---------------------------
126 /* USER CODE END PFP */
127
128 /* Private user code ---------------------------------------------------------*/
129 /* USER CODE BEGIN 0 */
130 // ----------- Variaveis globais --------------------
131 uint8_t  hora, min, seg;
132 RTC_TimeTypeDef relogio;
133 RTC_DateTypeDef calendario;
134 TIM_OC_InitTypeDef sConfig = {0};
135 char senha_alarme[5]= "3872", senha_user[5] = {0};
136 volatile HAL_StatusTypeDef ret;
137 int ret_error=0;
138 int erro = 0;
```

```c
int ok_alarme=0;// Para conferir se esta na faixa de acionamento do alarme
int abertura=0;// Verifica se abriu dentro da faixa de acionamento do alarme
int INVASAO=0;// Caso ok_alarme e abertura == 1... Aciona o alarme
uint8_t h_init, h_end, m_init, m_end;

char AONDE=NO_LCD;
// ----------- Variaveis globais --------------------

int __io_putchar(int ch){
    if (AONDE == NO_LCD){
        if (ch != '\n') lcd_wrchar(ch);
    }
    if (AONDE == NA_SERIAL){
            HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 100);
        }
    return ch;
}

// --------------------------------Interrupção de GPIO-------------
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
//  if((GPIO_Pin == GPIO_PIN_1)){
//      delay(100);
//      if((GPIOC->IDR & (1<<1))==0){corrente = 0;}// Entra cartao
//      else{corrente = 1;}// Sai cartao
//  }

    if((GPIO_Pin == GPIO_PIN_2))corrente =4;
}
/* USER CODE END 0 */

/**
  * @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_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();
```

```
208       HAL_Delay(1);
209     //   ----------------------Msg inicial--------------------------------
210        AONDE=NA_SERIAL;
211        printf("\rBUZZER\r\n");
212
213     //  ----------------------Msg inicial--------------------------------
214
215     //  ----------------------Ajusta hora do rtc e do alarme----------------------
216    //      ajuste_hora();
217    //      HAL_Delay(1500);
218    //      mostrahoras(NA_SERIAL);
219    //      ajuste_hora_inicial();
220    //      ajuste_hora_final();
221    //
222    //      AONDE= NO_LCD;
223    //      lcd_clear();
224    //      lcd_goto(0,0);
225    //      printf("Init %02d:%02d\n", m_init, h_init);
226    //      lcd_goto(1,0);
227    //      printf("End %02d:%02d\n", m_end, h_end);
228    //      HAL_Delay(2000);
229    //  lcd_clear();
230     //  ----------------------Ajusta hora do alarme----------------------
231        AONDE= NA_SERIAL;
232        printf("\rDigite uma senha de 4 digitos para desligar o alarme\r\n");
233        save_pw_alarme();
234        HAL_Delay(100);
235        printf("\rDigite uma senha de 4 digitos para entrar\r\n");
236        save_pw_in();
237
238    //  ----------------------Ajusta hora do alarme----------------------
239      /* USER CODE END 2 */
240
241      /* Infinite loop */
242      /* USER CODE BEGIN WHILE */
243      while (1)
244      {
245          if(corrente == 2){
246             // printf("\rC\n\r");
247            acinona_se();// Muda a variavel pra 1 se estiver dentro do tempo de acionamento
248             // printf("\r2\n\r");
249            alarme();// Liga o rele se tiver  alarme_ok e abertura ==1
250             // printf("\r3\n\r");
251            mostrahoras(NO_LCD);
252            AONDE=NO_LCD;
253            lcd_goto(0,0);
254            if((GPIOC->IDR & (1<<2))== 0){
255              printf("CLOSED(%d)| OK(%d)\n", abertura, ok_alarme);
256            }
257            else{
258              printf("OPENED(%d)| OK(%d)\n", abertura, ok_alarme);
259            }
260            lcd_goto(1,0);
261            printf("INV(%d)\n", INVASAO);
262          }
263          if(corrente == 3){//DESACIONA O ALARME
264              AONDE=NO_LCD;
265              lcd_clear();
266              lcd_goto(0,0);
267              //printf("OPEN(%d)\n", abertura);
268              lcd_goto(1,0);
269              //printf("INV(%d)\n", INVASAO);
270              desliga();// Espera a senha certa pra desacionar e se desaciona volta pro
                 estado 2
271          }
272          if(corrente == 4){// A interrupcao deixa ele aqui Aqui verifica se abriu a porta
             na faixa da hora
273            HAL_Delay(50);
274            //printf("\rx\n\r");
```

```
275           if(ok_alarme == 1){// Porta aberta e dentro da hora do alarme
276               if((GPIOC->IDR & (1<<2))== 0) abertura = 1;
277               corrente = 2;
278           }
279           else {
280               corrente = 2;
281           }
282           if(INVASAO==1) corrente = 3;// Caso esta aqui por engano
283           AONDE = NO_LCD;
284           lcd_goto(1, 9);
285           printf("4\n");
286           lcd_clear();
287         }
288           HAL_Delay(1000);
289       /* USER CODE END WHILE */
290 
291       /* USER CODE BEGIN 3 */
292     }
293     /* USER CODE END 3 */
294   }
295 
296   /**
297     * @brief System Clock Configuration
298     * @retval None
299     */
300   void SystemClock_Config(void)
301   {
302     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
303     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
304     RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
305 
306     /** Initializes the RCC Oscillators according to the specified parameters
307     * in the RCC_OscInitTypeDef structure.
308     */
309     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI48|RCC_OSCILLATORTYPE_LSI;
310     RCC_OscInitStruct.HSI48State = RCC_HSI48_ON;
311     RCC_OscInitStruct.LSIState = RCC_LSI_ON;
312     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
313     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
314     {
315       Error_Handler();
316     }
317 
318     /** Initializes the CPU, AHB and APB buses clocks
319     */
320     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
321                                   |RCC_CLOCKTYPE_PCLK1;
322     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI48;
323     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
324     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
325 
326     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
327     {
328       Error_Handler();
329     }
330     PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART2|RCC_PERIPHCLK_RTC;
331     PeriphClkInit.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
332     PeriphClkInit.RTCClockSelection = RCC_RTCCLKSOURCE_LSI;
333     if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
334     {
335       Error_Handler();
336     }
337   }
338 
339   /**
340     * @brief RTC Initialization Function
341     * @param None
342     * @retval None
343     */
```

```c
static void MX_RTC_Init(void)
{

  /* USER CODE BEGIN RTC_Init 0 */

  /* USER CODE END RTC_Init 0 */

  RTC_TimeTypeDef sTime = {0};
  RTC_DateTypeDef sDate = {0};

  /* USER CODE BEGIN RTC_Init 1 */

  /* USER CODE END RTC_Init 1 */

  /** Initialize RTC Only
  */
  hrtc.Instance = RTC;
  hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
  hrtc.Init.AsynchPrediv = 127;
  hrtc.Init.SynchPrediv = 255;
  hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
  hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
  hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
  if (HAL_RTC_Init(&hrtc) != HAL_OK)
  {
    Error_Handler();
  }

  /* USER CODE BEGIN Check_RTC_BKUP */

  /* USER CODE END Check_RTC_BKUP */

  /** Initialize RTC and set the Time and Date
  */
  sTime.Hours = 0;
  sTime.Minutes = 0;
  sTime.Seconds = 0;
  sTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;
  sTime.StoreOperation = RTC_STOREOPERATION_RESET;
  if (HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BIN) != HAL_OK)
  {
    Error_Handler();
  }
  sDate.WeekDay = RTC_WEEKDAY_MONDAY;
  sDate.Month = RTC_MONTH_JANUARY;
  sDate.Date = 1;
  sDate.Year = 0;

  if (HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BIN) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN RTC_Init 2 */

  /* USER CODE END RTC_Init 2 */

}

/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */
```

```
413
414     TIM_ClockConfigTypeDef sClockSourceConfig = {0};
415     TIM_MasterConfigTypeDef sMasterConfig = {0};
416
417     /* USER CODE BEGIN TIM1_Init 1 */
418
419     /* USER CODE END TIM1_Init 1 */
420     htim1.Instance = TIM1;
421     htim1.Init.Prescaler = 47;
422     htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
423     htim1.Init.Period = 65535;
424     htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
425     htim1.Init.RepetitionCounter = 0;
426     htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
427     if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
428     {
429       Error_Handler();
430     }
431     sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
432     if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
433     {
434       Error_Handler();
435     }
436     sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
437     sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
438     if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
439     {
440       Error_Handler();
441     }
442     /* USER CODE BEGIN TIM1_Init 2 */
443
444     /* USER CODE END TIM1_Init 2 */
445
446   }
447
448   /**
449     * @brief USART2 Initialization Function
450     * @param None
451     * @retval None
452     */
453   static void MX_USART2_UART_Init(void)
454   {
455
456     /* USER CODE BEGIN USART2_Init 0 */
457
458     /* USER CODE END USART2_Init 0 */
459
460     /* USER CODE BEGIN USART2_Init 1 */
461
462     /* USER CODE END USART2_Init 1 */
463     huart2.Instance = USART2;
464     huart2.Init.BaudRate = 9600;
465     huart2.Init.WordLength = UART_WORDLENGTH_8B;
466     huart2.Init.StopBits = UART_STOPBITS_1;
467     huart2.Init.Parity = UART_PARITY_NONE;
468     huart2.Init.Mode = UART_MODE_TX_RX;
469     huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
470     huart2.Init.OverSampling = UART_OVERSAMPLING_16;
471     huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
472     huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
473     if (HAL_UART_Init(&huart2) != HAL_OK)
474     {
475       Error_Handler();
476     }
477     /* USER CODE BEGIN USART2_Init 2 */
478
479     /* USER CODE END USART2_Init 2 */
480
481   }
```

```c

/**
  * @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_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3|GPIO_PIN_7, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, LD2_Pin|GPIO_PIN_8|GPIO_PIN_9, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10|GPIO_PIN_4|GPIO_PIN_5, GPIO_PIN_RESET);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : PC2 */
  GPIO_InitStruct.Pin = GPIO_PIN_2;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : PC3 PC7 */
  GPIO_InitStruct.Pin = GPIO_PIN_3|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : LD2_Pin PA8 PA9 */
  GPIO_InitStruct.Pin = LD2_Pin|GPIO_PIN_8|GPIO_PIN_9;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PB10 PB4 PB5 */
  GPIO_InitStruct.Pin = GPIO_PIN_10|GPIO_PIN_4|GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI2_3_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI2_3_IRQn);

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

/* USER CODE BEGIN 4 */
```

```c
void save_pw_alarme(void){
    char senha[5];
    int k=0;
    for (k=0; k<4; k++){//Digita a senha
        do{
            erro = HAL_UART_Receive(&huart2, (uint8_t*)&senha[k], 1, 10);
        }while(erro != HAL_UART_ERROR_NONE);
        HAL_UART_Transmit(&huart2, (uint8_t*)&senha[k], 1, 2);
        if((k+1)==4)senha[4]='\0';
    }
    HAL_Delay(3);
    strcpy(senha_alarme, senha);
    printf("\rSenha alarme definida: %s\r\n", senha_alarme);
}
void save_pw_in(void){
    char senha[5];
    int k=0;
    for (k=0; k<4; k++){//Digita a senha
        do{
            erro = HAL_UART_Receive(&huart2, (uint8_t*)&senha[k], 1, 10);
        }while(erro != HAL_UART_ERROR_NONE);
        HAL_UART_Transmit(&huart2, (uint8_t*)&senha[k], 1, 2);
        if((k+1)==4)senha[4]='\0';
    }
    HAL_Delay(3);
    strcpy(senha_user, senha);
    printf("\rSenha user definida: %s\r\n", senha_user);
}
// --------------------------------Delay--------------------------
void delay(uint16_t us){
    uint16_t start = __HAL_TIM_GET_COUNTER(&htim1); // Lê o valor atual do contador
    while ((__HAL_TIM_GET_COUNTER(&htim1) - start) < us); // Aguarda até que a diferença
    atinja 'us'
}
// ------------------------------------- HORAS AJUSTES-------------------------------
void ajuste_hora(void){// CONFIGURAÇÃO DA HORA E MINUTOS
    char ch=0, u=0, d=0, h=0, m=0;
    AONDE = NO_LCD;
    lcd_goto(0,0);
    printf("CONFIG. HORAS:\n");
    HAL_Delay(1500);
    lcd_clear();
    printf("Digite a hora:\n");
    AONDE = NA_SERIAL;
    printf("\n");
    printf("-----------------------------------------------\n");
    printf("\rConfiguração das horas do RTC:\r\n");
    do{// AJUSTE DO HORA
        ch=0;u=0; d=0; h=0;
        AONDE = NA_SERIAL;
        printf("\rDigite a hora [00-23]:\r\n");
        do{
            erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
        } while (erro != HAL_UART_ERROR_NONE);
        HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2); // eco na serial
        d = ch-'0'; // Converte ASCII em DECIMAL
        do{
            erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
        } while (erro != HAL_UART_ERROR_NONE);
        HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2);
        u = ch-'0';
        h = 10 * d + u;
        if(h>23)printf("\rHora inválida, digite novamente!\r\n");
        HAL_Delay(3);
        printf("\r\n");
    }while(h>23);
    AONDE = NO_LCD;
    lcd_clear();
    lcd_goto(0,0);
```

```
619         printf("Horas config.: %d\n", h);
620         HAL_Delay(3);
621         lcd_clear();
622         lcd_goto(0,0);
623         printf("Digite a min:\n");
624         do{// AJUSTE DO MINUTO
625             ch=0;u=0; d=0; m=0;
626             AONDE = NA_SERIAL;
627             printf("\rDigite a minutos [00-59]: \r\n");
628             do{
629                 erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
630             } while (erro != HAL_UART_ERROR_NONE);
631             HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2);
632             d = ch-'0'; // Converte ASCII em DECIMAL
633             do{
634                 erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
635             } while (erro != HAL_UART_ERROR_NONE);
636             HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2);
637             u = ch-'0';
638             m = 10 * d + u;
639             if(m>59)printf("\rMinutos inválido, digite novamente!\r\n");
640             HAL_Delay(3);
641             printf("\r\n");
642         }while(m>59);
643         hora = (uint8_t)h;
644         min = (uint8_t)m;
645         seg = 0;
646         HAL_Delay(1500);
647         relogio.Hours = hora;
648         relogio.Minutes = min;
649         relogio.Seconds = seg;
650         HAL_RTC_SetTime(&hrtc, &relogio, RTC_FORMAT_BIN);
651         calendario.Year = 23;
652         calendario.Month = 12;
653         calendario.Date = 12;
654         HAL_RTC_SetDate(&hrtc, &calendario, RTC_FORMAT_BIN);
655         HAL_RTC_WaitForSynchro(&hrtc);
656         HAL_Delay(1500);
657 
658     }
659     void ajuste_hora_inicial(void){// CONFIGURAÇÃO DA HORA E MINUTOS
660         char ch=0, u=0, d=0;
661         char h=0, m=0;
662         AONDE = NO_LCD;
663         lcd_goto(0,0);
664         lcd_clear();
665         printf("Digite a hora:\n");
666         AONDE = NA_SERIAL;
667         printf("\r-----------------------------------------\r\n");
668         printf("\rConfiguração do horário inicial do alarme:\r\n");
669         do{// AJUSTE DO HORA
670             ch=0;u=0; d=0; h=0;
671             AONDE = NA_SERIAL;
672             printf("\rDigite a hora inicial [00-23]:\r\n");
673             do{
674                 erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
675             } while (erro != HAL_UART_ERROR_NONE);
676             HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2); // eco na serial
677             d = ch-'0'; // Converte ASCII em DECIMAL
678             do{
679                 erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
680             } while (erro != HAL_UART_ERROR_NONE);
681             HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2);
682             u = ch-'0';
683             h = 10 * d + u;
684             if(h>23)printf("\rHora inválida, digite novamente!\r\n");
685             HAL_Delay(30);
686             printf("\r\n");
687         }while(h>23);
```

```
688         h_init = h;
689         AONDE = NO_LCD;
690         lcd_clear();
691         lcd_goto(0,0);
692         printf("Horas config: %d\n", h);
693         HAL_Delay(1000);
694         lcd_clear();
695         lcd_goto(0,0);
696         printf("Digite a min:\n");
697         do{// AJUSTE DO MINUTO
698             ch=0;u=0; d=0; m=0;
699             AONDE = NA_SERIAL;
700             printf("\rDigite a minutos [00-59]:\r\n");
701             do{
702                 erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
703             } while (erro != HAL_UART_ERROR_NONE);
704             HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2);
705             d = ch-'0'; // Converte ASCII em DECIMAL
706             do{
707                 erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
708             } while (erro != HAL_UART_ERROR_NONE);
709             HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2);
710             u = ch-'0';
711             m = 10 * d + u;
712             if(m>59)printf("\rMinutos inválido, digite novamente!\r\n");
713             HAL_Delay(30);
714             printf("\r\n");
715         }while(m>59);
716         m_init = m;
717         AONDE = NO_LCD;
718         lcd_clear();
719         lcd_goto(0,0);
720         printf("Min init conf:%d\n", m_init);
721         HAL_Delay(30);
722         lcd_clear();
723         AONDE = NA_SERIAL;
724         printf("\n");
725         printf("\rDef init %02d:%02d\r\n", h_init, m_init);
726         printf("\r-----------------------------------------\r\n");
727     }
728     void ajuste_hora_final(void){// CONFIGURAÇÃO DA HORA E MINUTOS
729         char ch=0, u=0, d=0, h=0, m=0;
730         AONDE = NO_LCD;
731         lcd_goto(0,0);
732         lcd_clear();
733         printf("Digite a hora:\n");
734         AONDE = NA_SERIAL;
735         printf("\n");
736         printf("\r-----------------------------------------\r\n");
737         printf("\rConfiguração das horas finais do alarme:\r\n");
738         do{// AJUSTE DO HORA
739             ch=0;u=0; d=0; h=0;
740             AONDE = NA_SERIAL;
741             printf("\rDigite a hora final[00-23]:\r\n");
742             do{
743                 erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
744             } while (erro != HAL_UART_ERROR_NONE);
745             HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2); // eco na serial
746             d = ch-'0'; // Converte ASCII em DECIMAL
747             do{
748                 erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
749             } while (erro != HAL_UART_ERROR_NONE);
750             HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2);
751             u = ch-'0';
752             h = 10 * d + u;
753             if(h>23)printf("\rHora inválida, digite novamente!\r\n");
754             HAL_Delay(30);
755             printf("\r\n");
756         }while(h>23);
```

```
757         h_end = h;
758         AONDE = NO_LCD;
759         lcd_clear();
760         lcd_goto(0,0);
761         printf("Horas config! %d\n", h_end);
762         HAL_Delay(600);
763         lcd_clear();
764         lcd_goto(0,0);
765         printf("Digite a min:\n");
766         do{// AJUSTE DO MINUTO
767             ch=0;u=0; d=0; m=0;
768             AONDE = NA_SERIAL;
769             printf("\rDigite a minutos [00-59]:\r\n");
770             do{
771                 erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
772             } while (erro != HAL_UART_ERROR_NONE);
773             HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2);
774             d = ch-'0'; // Converte ASCII em DECIMAL
775             do{
776                 erro = HAL_UART_Receive(&huart2, (uint8_t*)&ch, 1, 10);
777             } while (erro != HAL_UART_ERROR_NONE);
778             HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 2);
779             u = ch-'0';
780             m = 10 * d + u;
781             if(m>59)printf("\rMinutos inválido, digite novamente!\r\n");
782             HAL_Delay(30);
783             printf("\r\n");
784         }while(m>59);
785         m_end = m;
786         AONDE = NO_LCD;
787         lcd_clear();
788         lcd_goto(0,0);
789         printf("Min end conf:%d\n", m_end);
790         HAL_Delay(600);
791         lcd_clear();
792         AONDE = NA_SERIAL;
793         printf("\n");
794         printf("\rDef end %02d:%02d\r\n", h_end, m_end);
795         printf("\r-----------------------------------------\r\n");
796     }
797     void mostrahoras(int x){// Mostra a hora na serial ou no LCD
798         AONDE=x;
799         HAL_RTC_GetTime(&hrtc, &relogio, RTC_FORMAT_BIN);
800         HAL_RTC_GetDate(&hrtc, &calendario, RTC_FORMAT_BIN);
801 
802         hora = relogio.Hours;
803         min  = relogio.Minutes;
804         seg  = relogio.Seconds;
805         lcd_goto(1 ,8);
806         if(AONDE==NA_SERIAL) printf("\r%02d:%02d:%02d\r\n", hora, min, seg);
807         else printf("%02d:%02d:%02d\n", hora, min, seg);
808     }
809     // ---------------------------------- BUZZER ----------------------------------------
810     void acinona_se(void){// Aviasa que asiona o alarme se entrar no periodo definido do
    alarme
811         HAL_RTC_GetTime(&hrtc, &relogio, RTC_FORMAT_BIN);
812         hora = relogio.Hours;
813         min  = relogio.Minutes;
814         seg = relogio.Seconds;
815 
816         if(h_init<=h_end){//10:06 e 15:01 ou 10:00 e 10:10
817             if((hora>=h_init)&&(hora<=h_end)){
818                 if(h_init==h_end){//10:00 e 10:10
819                     if((min>=m_init)&&(min<=m_end))
820                     ok_alarme = 1;
821                 }
822                 else if(((hora>h_init)||(hora<h_end))|| //10:06 e 15:01
823                     ((hora == h_init)&&(min>=m_init))||
824                     ((hora == h_end)&&(min<=m_end))){
```

```
825                 ok_alarme = 1;
826             }
827         }
828     }
829 
830     else if(h_init>h_end){//22:00 e 8:00
831         if((hora>=h_init)||(hora<=h_end)){
832             if(((hora==h_init)&&(min>=m_init))||((hora==h_end)&&(min<=m_end))||(hora<
                h_end)||(hora>h_init)){
833                 ok_alarme = 1;
834             }
835         }
836     }
837 
838     else ok_alarme = 0;
839 }
840 
841 void alarme(void){
842     if(ok_alarme && abertura){
843         HAL_RTC_GetTime(&hrtc, &relogio, RTC_FORMAT_BIN);
844         hora = relogio.Hours;
845         min  = relogio.Minutes;
846         seg  = relogio.Seconds;
847         AONDE= NA_SERIAL;
848         printf("\rInvasão registrada ás %02d:%02d:%02d\r\n", hora, min, seg);
849         lcd_clear();
850         BUZZER_ON;
851         INVASAO=1;
852 
853         corrente = 3;// para ficar ativo ate que alguem digite a senha;
854     }
855 }
856 void desliga(void){
857     char senha_dig[5];
858     AONDE = NO_LCD;
859     lcd_clear();
860     lcd_goto(0,3);
861     printf("***ALERTA !!!\n");
862     lcd_goto(1,3);
863     printf("***INVASAO\n");
864     BUZZER_ON;
865     while(1){
866         AONDE= NA_SERIAL;
867         printf("\rDigite a senha de 4 dig para desligar o alarme\r\n");
868         for(int k=0; k<4; k++){//Digita a senha
869             do{
870                 erro = HAL_UART_Receive(&huart2, (uint8_t*)&senha_dig[k], 1, 10);
871             }while(erro != HAL_UART_ERROR_NONE);
872             HAL_UART_Transmit(&huart2, (uint8_t*)&senha_dig[k], 1, 2);
873             if((k+1)==4)senha_dig[4]='\0';
874         }
875         acinona_se();//Verifica se atingiu m_end:h_end -> TIMEOUT
876         if(strcmp(senha_dig, senha_alarme)==0){// Caso a senha seja a mesma que esta na
            senha_alarme desativa alarme
877             BUZZER_OFF;
878             INVASAO=0;
879             abertura = 0;
880             corrente = 2;
881             AONDE=NO_LCD;
882             lcd_clear();
883             lcd_goto(0,0);
884             printf("ALARME OFF\n");
885             HAL_RTC_GetTime(&hrtc, &relogio, RTC_FORMAT_BIN);
886             hora = relogio.Hours;
887             min  = relogio.Minutes;
888             seg  = relogio.Seconds;
889             AONDE= NA_SERIAL;
890             printf("\rDesligado ás %02d:%02d:%02d\r\n", hora, min, seg);
891             HAL_Delay(1000);
```

```
            break;
        }
        else if(ok_alarme==0){//Atingiu m_end:h_end
            BUZZER_OFF;
            INVASAO=0;
            abertura = 0;
            corrente = 2;
            HAL_RTC_GetTime(&hrtc, &relogio, RTC_FORMAT_BIN);
            hora = relogio.Hours;
            min  = relogio.Minutes;
            seg  = relogio.Seconds;
            AONDE= NA_SERIAL;
            printf("\rTIMEOUT ás %02d:%02d:%02d\r\n", hora, min, seg);
            AONDE=NO_LCD;
            lcd_clear();
            lcd_goto(0,0);
            printf("TIMEOUT\n");
            corrente = 2;
            HAL_Delay(1000);
            break;
        }
        else{// Caso erre a senha pergunta de novo
            memset(senha_dig, 0, sizeof(senha_dig));
            AONDE= NA_SERIAL;
            printf("\rSenha incorreta!\r\n");
            HAL_Delay(50);
        }
    }
}

//---------------------------------LCD-----------------------------
void lcd_backlight (uint8_t light){
    if(light == 0){
        GPIOB -> BRR = 1<<3;
    } else {
        GPIOB -> BSRR= 1<<3;
    }
}
void lcd_init(uint8_t cursor){
    lcd_wrcom4(3);
    lcd_wrcom4(3);
    lcd_wrcom4(3);
    lcd_wrcom4(2);
    lcd_wrcom(0x28);
    lcd_wrcom(cursor);
    lcd_wrcom(0x06);
    lcd_wrcom(0x01);
}
void lcd_wrcom4(uint8_t com4){
    lcd_senddata(com4); //D4...d0
    RS_0;
    EN_1;
    delayus(5);
    EN_0;
    HAL_Delay(5);
}
void lcd_wrcom(uint8_t com){
    lcd_senddata(com>>4); //0000D7...D4
    RS_0;
    EN_1;
    delayus(5);
    EN_0;
    delayus(5);

    lcd_senddata(com & 0x0F); //0000D3...d0
    EN_1;
    delayus(5);
    EN_0;
    HAL_Delay(5);
```

```
961   }
962   void lcd_clear(void){
963       lcd_wrcom(0x01);
964   }
965   //goto para 16x2
966   void lcd_goto(uint8_t x, uint8_t y){
967       uint8_t com = 0x80;
968       if (x==0 && y<16) com = 0x80 + y;
969       else if (x==1 && y<16) com = 0xC0 + y;
970       else com = 0x80;
971       lcd_wrcom(com);
972   }
973   void lcd_wrchar(char ch){
974       lcd_senddata(ch>>4); //D7...D4
975       RS_1;
976       EN_1;
977       delayus(5);
978       EN_0;
979       delayus(5);
980
981       lcd_senddata(ch & 0x0F); //D3...D0
982       RS_1;
983       EN_1;
984       delayus(5);
985       EN_0;
986       HAL_Delay(5);
987   }
988
989   void lcd_wrstr(char *str){
990       while(*str) lcd_wrchar(*str++);
991   }
992
993   void udelay(void){
994       int tempo = 7;
995       while(tempo--);
996   }
997
998   void delayus(int tempo){
999       while(tempo--) udelay();
1000  }
1001
1002  void lcd_wr2dig(uint8_t valor){
1003      lcd_wrchar(valor/10 + '0'); // ou +48 -> dezena
1004      lcd_wrchar(valor%10 + '0'); // ou +48 -> unidade
1005  }
1006
1007  void lcd_senddata(uint8_t data){
1008      if((data & (1<<3))==0) D7_0; else D7_1;
1009      if((data & (1<<2))==0) D6_0; else D6_1;
1010      if((data & (1<<1))==0) D5_0; else D5_1;
1011      if((data & (1<<0))==0) D4_0; else D4_1;
1012  }
1013  /* USER CODE END 4 */
1014
1015  /**
1016    * @brief  This function is executed in case of error occurrence.
1017    * @retval None
1018    */
1019  void Error_Handler(void)
1020  {
1021    /* USER CODE BEGIN Error_Handler_Debug */
1022    /* User can add his own implementation to report the HAL error return state */
1023    __disable_irq();
1024    while (1)
1025    {
1026    }
1027    /* USER CODE END Error_Handler_Debug */
1028  }
1029
```

```
1030  #ifdef  USE_FULL_ASSERT
1031  /**
1032    * @brief  Reports the name of the source file and the source line number
1033    *         where the assert_param error has occurred.
1034    * @param  file: pointer to the source file name
1035    * @param  line: assert_param error line source number
1036    * @retval None
1037    */
1038  void assert_failed(uint8_t *file, uint32_t line)
1039  {
1040    /* USER CODE BEGIN 6 */
1041    /* User can add his own implementation to report the file name and line number,
1042       ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
1043    /* USER CODE END 6 */
1044  }
1045  #endif /* USE_FULL_ASSERT */
1046
```