```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 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 "MY_NRF24.h"
#include "i2c_20x4_lcd.h"
#include "stdlib.h"
#include "flash.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 address_data_storage 0x800FC00
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;

SPI_HandleTypeDef hspi1;

TIM_HandleTypeDef htim2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
```

```
  59  static void MX_SPI1_Init(void);
  60  static void MX_I2C1_Init(void);
  61  static void MX_TIM2_Init(void);
  62  /* USER CODE BEGIN PFP */
  63  uint8_t isButtonPressed(uint16_t buttonPin, uint16_t dl);
  64  void set_temp_increase(void);
  65  void set_temp_decrease(void);
  66  void display_setTemp(void);
  67  //void display_Temp(void);
  68  void displayTemp(void);
  69  void blinking_display(void);
  70  /* USER CODE END PFP */
  71
  72  /* Private user code ---------------------------------------------------------*/
  73  /* USER CODE BEGIN 0 */
  74  uint64_t addr = 0x0011223344;
  75  uint8_t Rxdata[10];
  76  uint8_t Temp[5];
  77  uint8_t direct_set_temp[5];
  78  volatile bool setup;
  79  volatile uint8_t bdn = 0;
  80  float setTemp = 25.0;
  81  float tam;
  82  char t[5];
  83  /* USER CODE END 0 */
  84
  85  /**
  86    * @brief  The application entry point.
  87    * @retval int
  88    */
  89  int main(void)
  90  {
  91    /* USER CODE BEGIN 1 */
  92
  93    /* USER CODE END 1 */
  94
  95    /* MCU Configuration--------------------------------------------------------*/
  96
  97    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  98    HAL_Init();
  99
 100    /* USER CODE BEGIN Init */
 101
 102    /* USER CODE END Init */
 103
 104    /* Configure the system clock */
 105    SystemClock_Config();
 106
 107    /* USER CODE BEGIN SysInit */
 108
 109    /* USER CODE END SysInit */
 110
 111    /* Initialize all configured peripherals */
 112    MX_GPIO_Init();
 113    MX_SPI1_Init();
 114    MX_I2C1_Init();
 115    MX_TIM2_Init();
 116    /* USER CODE BEGIN 2 */
```

```
117     //LCD INIT
118     lcd_clear();
119     lcd_init();
120     HAL_Delay(500);
121     lcd_cgram_init();
122     //NRF24 INIT
123     NRF24_begin(GPIOA,GPIO_PIN_4,GPIO_PIN_3,hspi1);
124     NRF24_setAutoAck(true);
125     NRF24_setChannel(1);
126     NRF24_setPayloadSize(10);
127     NRF24_openReadingPipe(1,addr);
128     NRF24_enableDynamicPayloads();
129     NRF24_enableAckPayload();
130     NRF24_startListening();
131     HAL_TIM_Base_Start_IT(&htim2);
132     /* USER CODE END 2 */
133
134     /* Infinite loop */
135     /* USER CODE BEGIN WHILE */
136     setup = false;
137     lcd_put_cur(0,0);
138     lcd_send_string("Set Temp:");
139     lcd_put_cur(1,0);
140     lcd_send_string("Temp ");
141     lcd_send_data('(');
142     lcd_send_data(223);
143     lcd_send_string("C):");
144     setTemp = flashReadFloat(address_data_storage);
145     tam = setTemp;
146     while (1)
147     {
148       //hien thi thong tin lcd
149       if(setup==false)
150       {
151         while(tam != setTemp)
152         {
153           flashErase(address_data_storage);
154           flashWriteFloat(address_data_storage,setTemp);
155           tam = setTemp;
156         }
157         //truyen nhan RF24L01
158         if(NRF24_available())
159         {
160           NRF24_read(Rxdata,10);
161           sprintf((char *)direct_set_temp,"%.1f",setTemp);
162           NRF24_writeAckPayload(1,direct_set_temp,5);
163         }
164         if(Rxdata[0]!='*')
165         {
166           for(uint8_t i=0;i<5;i++)
167             Temp[i] = Rxdata[i];
168         }
169         else
170         {
171           for(uint8_t i=0;i<5;i++)
172             direct_set_temp[i] = Rxdata[i+1];
173           setTemp = atof((char *)direct_set_temp);
174         }
```

```
175         display_setTemp();
176       }
177       else
178       {
179         if(NRF24_available())
180         {
181           NRF24_read(Rxdata,10);
182           NRF24_writeAckPayload(1,"setup",5);
183         }
184         if(Rxdata[0]!='*')
185         {
186           for(uint8_t i=0;i<5;i++)
187             Temp[i] = Rxdata[i];
188         }
189         blinking_display();
190         set_temp_increase();
191         set_temp_decrease();
192       }
193         displayTemp();
194       /* USER CODE END WHILE */
195
196       /* USER CODE BEGIN 3 */
197
198     /* USER CODE END 3 */
199   }
200   }
201   /**
202     * @brief System Clock Configuration
203     * @retval None
204     */
205   void SystemClock_Config(void)
206   {
207     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
208     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
209
210     /** Initializes the RCC Oscillators according to the specified parameters
211     * in the RCC_OscInitTypeDef structure.
212     */
213     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
214     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
215     RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
216     RCC_OscInitStruct.HSIState = RCC_HSI_ON;
217     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
218     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
219     RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
220     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
221     {
222       Error_Handler();
223     }
224
225     /** Initializes the CPU, AHB and APB buses clocks
226     */
227     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
228                                   |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
229     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
230     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV2;
231     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV8;
232     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
```

```
233
234     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
235     {
236       Error_Handler();
237     }
238   }
239
240   /**
241     * @brief I2C1 Initialization Function
242     * @param None
243     * @retval None
244     */
245   static void MX_I2C1_Init(void)
246   {
247
248     /* USER CODE BEGIN I2C1_Init 0 */
249
250     /* USER CODE END I2C1_Init 0 */
251
252     /* USER CODE BEGIN I2C1_Init 1 */
253
254     /* USER CODE END I2C1_Init 1 */
255     hi2c1.Instance = I2C1;
256     hi2c1.Init.ClockSpeed = 100000;
257     hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
258     hi2c1.Init.OwnAddress1 = 0;
259     hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
260     hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
261     hi2c1.Init.OwnAddress2 = 0;
262     hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
263     hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
264     if (HAL_I2C_Init(&hi2c1) != HAL_OK)
265     {
266       Error_Handler();
267     }
268     /* USER CODE BEGIN I2C1_Init 2 */
269
270     /* USER CODE END I2C1_Init 2 */
271
272   }
273
274   /**
275     * @brief SPI1 Initialization Function
276     * @param None
277     * @retval None
278     */
279   static void MX_SPI1_Init(void)
280   {
281
282     /* USER CODE BEGIN SPI1_Init 0 */
283
284     /* USER CODE END SPI1_Init 0 */
285
286     /* USER CODE BEGIN SPI1_Init 1 */
287
288     /* USER CODE END SPI1_Init 1 */
289     /* SPI1 parameter configuration*/
290     hspi1.Instance = SPI1;
```

```
291     hspi1.Init.Mode = SPI_MODE_MASTER;
292     hspi1.Init.Direction = SPI_DIRECTION_2LINES;
293     hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
294     hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
295     hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
296     hspi1.Init.NSS = SPI_NSS_SOFT;
297     hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32;
298     hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
299     hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
300     hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
301     hspi1.Init.CRCPolynomial = 10;
302     if (HAL_SPI_Init(&hspi1) != HAL_OK)
303     {
304       Error_Handler();
305     }
306     /* USER CODE BEGIN SPI1_Init 2 */
307
308     /* USER CODE END SPI1_Init 2 */
309
310   }
311
312   /**
313     * @brief TIM2 Initialization Function
314     * @param None
315     * @retval None
316     */
317   static void MX_TIM2_Init(void)
318   {
319
320     /* USER CODE BEGIN TIM2_Init 0 */
321
322     /* USER CODE END TIM2_Init 0 */
323
324     TIM_ClockConfigTypeDef sClockSourceConfig = {0};
325     TIM_MasterConfigTypeDef sMasterConfig = {0};
326
327     /* USER CODE BEGIN TIM2_Init 1 */
328
329     /* USER CODE END TIM2_Init 1 */
330     htim2.Instance = TIM2;
331     htim2.Init.Prescaler = 8999;
332     htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
333     htim2.Init.Period = 99;
334     htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
335     htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
336     if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
337     {
338       Error_Handler();
339     }
340     sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
341     if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
342     {
343       Error_Handler();
344     }
345     sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
346     sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
347     if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
348     {
```

```
349       Error_Handler();
350     }
351     /* USER CODE BEGIN TIM2_Init 2 */
352
353     /* USER CODE END TIM2_Init 2 */
354
355   }
356
357   /**
358     * @brief GPIO Initialization Function
359     * @param None
360     * @retval None
361     */
362   static void MX_GPIO_Init(void)
363   {
364     GPIO_InitTypeDef GPIO_InitStruct = {0};
365   /* USER CODE BEGIN MX_GPIO_Init_1 */
366   /* USER CODE END MX_GPIO_Init_1 */
367
368     /* GPIO Ports Clock Enable */
369     __HAL_RCC_GPIOC_CLK_ENABLE();
370     __HAL_RCC_GPIOD_CLK_ENABLE();
371     __HAL_RCC_GPIOA_CLK_ENABLE();
372     __HAL_RCC_GPIOB_CLK_ENABLE();
373
374     /*Configure GPIO pin Output Level */
375     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
376
377     /*Configure GPIO pin Output Level */
378     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3|GPIO_PIN_4, GPIO_PIN_RESET);
379
380     /*Configure GPIO pin : PC13 */
381     GPIO_InitStruct.Pin = GPIO_PIN_13;
382     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
383     GPIO_InitStruct.Pull = GPIO_NOPULL;
384     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
385     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
386
387     /*Configure GPIO pins : PA3 PA4 */
388     GPIO_InitStruct.Pin = GPIO_PIN_3|GPIO_PIN_4;
389     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
390     GPIO_InitStruct.Pull = GPIO_NOPULL;
391     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
392     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
393
394     /*Configure GPIO pin : PB12 */
395     GPIO_InitStruct.Pin = GPIO_PIN_12;
396     GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
397     GPIO_InitStruct.Pull = GPIO_PULLUP;
398     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
399
400     /*Configure GPIO pins : PB13 PB14 */
401     GPIO_InitStruct.Pin = GPIO_PIN_13|GPIO_PIN_14;
402     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
403     GPIO_InitStruct.Pull = GPIO_PULLUP;
404     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
405
406     /* EXTI interrupt init*/
```

```
407     HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
408     HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
409
410   /* USER CODE BEGIN MX_GPIO_Init_2 */
411   /* USER CODE END MX_GPIO_Init_2 */
412   }
413
414   /* USER CODE BEGIN 4 */
415   void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
416   {
417     if(GPIO_Pin == GPIO_PIN_12)
418     {
419       setup = !setup;
420       for(int i = 500000; i>0; i--);
421       __HAL_GPIO_EXTI_CLEAR_IT(GPIO_PIN_12);
422       HAL_NVIC_ClearPendingIRQ(EXTI15_10_IRQn);
423     }
424   }
425   void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
426   {
427     if(htim->Instance == TIM2)
428     {
429       bdn++;
430       if(bdn==10) bdn=0;
431     }
432   }
433   //ham kiem tra nut nhan
434   uint8_t isButtonPressed(uint16_t buttonPin, uint16_t dl)
435   {
436     if(!(HAL_GPIO_ReadPin(GPIOB, buttonPin)))
437     {
438       HAL_Delay(20);
439       if(!(HAL_GPIO_ReadPin(GPIOB, buttonPin)))
440       {
441         HAL_Delay(dl);
442         return 1;
443       }
444       return 0;
445     }
446     return 0;
447   }
448   void set_temp_increase(void)
449   {
450     if(isButtonPressed(GPIO_PIN_13,100))
451     {
452       setTemp += 1;
453     }
454   }
455   void set_temp_decrease(void)
456   {
457     if(isButtonPressed(GPIO_PIN_14,100))
458     {
459       setTemp -= 1;
460     }
461   }
462
463   void display_setTemp(void)
464   {
```

```
465     sprintf(t,"%.0f",setTemp);
466     lcd_put_cur(0,9);
467     lcd_send_string(t);
468     lcd_put_cur(0,11);
469     lcd_send_data(223);
470     lcd_put_cur(0,12);
471     lcd_send_data('C');
472   }
473
474   void displayTemp(void)
475   {
476     if(atof((char *)Temp)<50)
477     {
478       lcd_ht_so_to(Temp[0],2,0);
479       lcd_ht_so_to(Temp[1],2,3);
480       lcd_ht_so_to(Temp[2],2,7);
481       lcd_ht_so_to(Temp[3],2,10);
482     }
483
484   }
485   void blinking_display(void)
486   {
487     if(bdn>5 && (HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_13)) &&
      HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_14))
488     {
489       for(uint8_t i = 0; i<5; i++)
490       {
491         lcd_put_cur(0,9+i);
492         lcd_send_data(' ');
493       }
494     }
495     else
496     {
497       display_setTemp();
498     }
499   }
500   /* USER CODE END 4 */
501
502   /**
503     * @brief  This function is executed in case of error occurrence.
504     * @retval None
505     */
506   void Error_Handler(void)
507   {
508     /* USER CODE BEGIN Error_Handler_Debug */
509     /* User can add his own implementation to report the HAL error return state */
510     __disable_irq();
511     while (1)
512     {
513     }
514     /* USER CODE END Error_Handler_Debug */
515   }
516
517   #ifdef  USE_FULL_ASSERT
518   /**
519     * @brief  Reports the name of the source file and the source line number
520     *         where the assert_param error has occurred.
521     * @param  file: pointer to the source file name
```

```
522     * @param  line: assert_param error line source number
523     * @retval None
524     */
525   void assert_failed(uint8_t *file, uint32_t line)
526   {
527     /* USER CODE BEGIN 6 */
528     /* User can add his own implementation to report the file name and line number,
529        ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
530     /* USER CODE END 6 */
531   }
532   #endif /* USE_FULL_ASSERT */
533
```