Введение в микроконтроллеры

Домашнее задание №2

Евгений Зеленин

16 января 2025 г.

1. Постановка задачи

Условия задачи. *Перепишите ДЗ с прошлого урока так, чтобы светодиод переключался по прерыванию при нажатии кнопки. Вынесите в отдельный файл функции инициализации светодиода (void init_led(GPIO_TypeDef* port, uint16_t pin)) и обработчик прерывания. ** сделайте счётчик нажатий с выводом на семисегментный индикатор; функции для работы с ним вынесите в пару файлов c./.h*

2. Схема устройства

На рисунке 1 показана схема устройства. Часть компонентов была подключена к порту A, часть к порту Б.

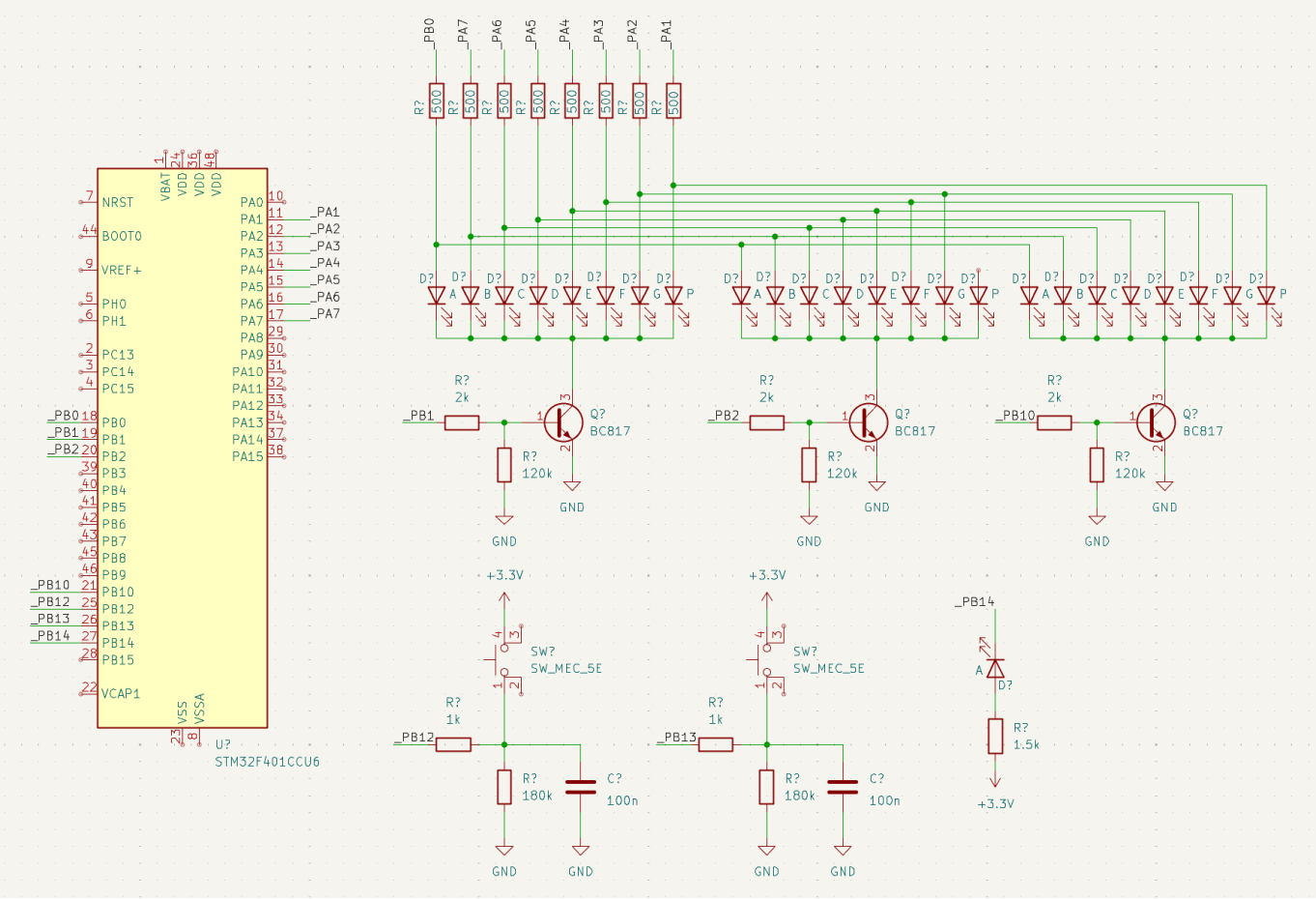

Рис. 1: Схема устройства

3. Описание работы устройства

При нажатии на кнопку 1 счетчик counter увеличивается на 1, а при нажатии на кнопку 2 - уменьшается на 1, допустимые значения: 000 - 999. К порту B, дополнительно, подключен отдельный светодиод. Каждые 10 нажатий он меняет свое состояние на противоположное. Обработка нажатий реализована с помощью прерываний, код обработчика вынесен в файл библиотеки. В файлах библиотеки сделаны функции - заглушки для возможного функционала по выводу анимации и символов отличных от цифр.

4. Исходный код проекта (main.c)

Для реализации обработки нажатий кнопок и вывода информации на 7-сегментный индикатор была создана библиотека my_L2_lib. Инициализация библиотеки происходит с помощью функции my_7seg_init(MY_7SEG seg). Параметр функции - структура с информацией о назначении портов и пинов.

Содержимое файла main.c:

```
/* 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 "mseglib.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 */

/* USER CODE END PM */

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

/* USER CODE BEGIN PV */
extern volatile int counter;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* 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();
 /* USER CODE BEGIN 2 */
 my_7seg_init(( MY_7SEG ) { {
   LED_A_GPIO_Port, LED_A_Pin }, {
   LED_B_GPIO_Port, LED_B_Pin }, {
   LED_C_GPIO_Port, LED_C_Pin }, {
   LED_D_GPIO_Port, LED_D_Pin }, {
   LED_E_GPIO_Port, LED_E_Pin }, {
   LED_F_GPIO_Port, LED_F_Pin }, {
   LED_G_GPIO_Port, LED_G_Pin }, {
   LED_P_GPIO_Port, LED_P_Pin }, {
   Q1_GPIO_Port, Q1_Pin }, {
   Q2_GPIO_Port, Q2_Pin }, {
   Q3_GPIO_Port, Q3_Pin } });
 /* USER CODE END 2 */

 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
 static int prev = 0;
 while (1) {
  //output to 7-seg indicator
  my_7seg_disp_num(counter);

  // switch every 10 counts
  if (prev != counter && !(counter % 10)) {
   HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
   prev = counter;
   }

  /* USER CODE END WHILE */

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

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void) {
```

```
 RCC_OscInitTypeDef RCC_OscInitStruct = { 0 };
 RCC_ClkInitTypeDef RCC_ClkInitStruct = { 0 };

 /** Configure the main internal regulator output voltage
  */
 __HAL_RCC_PWR_CLK_ENABLE();
 __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);

 /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
 RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
 RCC_OscInitStruct.HSIState = RCC_HSI_ON;
 RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
 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_HSI;
 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_0) != HAL_OK) {
  Error_Handler();
 }
}

/**
 * @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_GPIOH_CLK_ENABLE();
 __HAL_RCC_GPIOA_CLK_ENABLE();
```

```
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA,
  LED_P_Pin | LED_G_Pin | LED_F_Pin | LED_E_Pin | LED_D_Pin
    | LED_C_Pin | LED_B_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOB, LED_A_Pin | Q3_Pin | Q2_Pin | Q1_Pin | LED_Pin,
  GPIO_PIN_RESET);

/*Configure GPIO pins : PC13 PC14 PC15 */
GPIO_InitStruct.Pin = GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_15;
GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pins : PH0 PH1 */
GPIO_InitStruct.Pin = GPIO_PIN_0 | GPIO_PIN_1;
GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOH, &GPIO_InitStruct);

/*Configure GPIO pins : PA0 PA8 PA9 PA10
 PA11 PA12 PA15 */
GPIO_InitStruct.Pin = GPIO_PIN_0 | GPIO_PIN_8 | GPIO_PIN_9 | GPIO_PIN_10
  | GPIO_PIN_11 | GPIO_PIN_12 | GPIO_PIN_15;
GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pins : LED_P_Pin LED_G_Pin LED_F_Pin LED_E_Pin
 LED_D_Pin LED_C_Pin LED_B_Pin */
GPIO_InitStruct.Pin = LED_P_Pin | LED_G_Pin | LED_F_Pin | LED_E_Pin
  | LED_D_Pin | LED_C_Pin | LED_B_Pin;
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 : LED_A_Pin Q3_Pin Q2_Pin Q1_Pin
 LED_Pin */
GPIO_InitStruct.Pin = LED_A_Pin | Q3_Pin | Q2_Pin | Q1_Pin | LED_Pin;
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);
```

```
 /*Configure GPIO pins : BTN_1_Pin BTN_2_Pin */
 GPIO_InitStruct.Pin = BTN_1_Pin | BTN_2_Pin;
 GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
 GPIO_InitStruct.Pull = GPIO_NOPULL;
 HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

 /*Configure GPIO pins : PB15 PB3 PB4 PB5
  PB6 PB7 PB8 PB9 */
 GPIO_InitStruct.Pin = GPIO_PIN_15 | GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5
   | GPIO_PIN_6 | GPIO_PIN_7 | GPIO_PIN_8 | GPIO_PIN_9;
 GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
 GPIO_InitStruct.Pull = GPIO_NOPULL;
 HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

 /* EXTI interrupt init*/
 HAL_NVIC_SetPriority(EXTI15_10_IRQn, 3, 0);
 HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

 /* 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 */
```

Содержимое файла mseglib.h:

```
  /*
 * 7seglib.h
 *
 *  Created on: Jan 14, 2025
 *      Author: EZ-GPRO
 */
#include "stm32f4xx_hal.h"

#ifndef SEGLIB_H_
#define SEGLIB_H_

typedef struct{
 GPIO_TypeDef* port;
 uint16_t pin;

} pin_info;

typedef struct {
 pin_info A;
 pin_info B;
 pin_info C;
 pin_info D;
 pin_info E;
 pin_info F;
 pin_info G;
 pin_info P;
 pin_info Q1;
 pin_info Q2;
 pin_info Q3;
} MY_7SEG;

int my_7seg_init(MY_7SEG seg);
void my_7seg_disp_num(int num);
void my_7seg_disp_char(char letter, int pos);
```

```
void my_7seg_disp_anim(int num);


#endif /* SEGLIB_H_ */
```

Содержимое файла mseglib.c:

```
 /*
 * 7seglib.c
 *
 *  Created on: Jan 14, 2025
 *      Author: EZ-GPRO
 */
#include "mseglib.h"
#include "main.h"

volatile int counter = 0;

static int flag = -1;
static MY_7SEG disp = { };

void static set_pos(int pos);
void static print_digit(int num);
void static print_letter(char letter);
/**
 * @brief  This function make initialization of the lib
 * @retval flag {0,1}.
 */
int my_7seg_init(MY_7SEG seg) {
  flag = 1;
  if (seg.A.pin && seg.A.port) {
    disp.A = seg.A;
  } else {
    flag = 0;
  }
  if (seg.B.pin && seg.B.port) {
    disp.B = seg.B;
  } else {
    flag = 0;
  }
  if (seg.C.pin && seg.C.port) {
    disp.C = seg.C;
  } else {
    flag = 0;
  }
  if (seg.D.pin && seg.D.port) {
```

```
    disp.D = seg.D;
  } else {
    flag = 0;
  }
  if (seg.E.pin && seg.E.port) {
    disp.E = seg.E;
  } else {
    flag = 0;
  }
  if (seg.F.pin && seg.F.port) {
    disp.F = seg.F;
  } else {
    flag = 0;
  }
  if (seg.G.pin && seg.G.port) {
    disp.G = seg.G;
  } else {
    flag = 0;
  }
  if (seg.P.pin && seg.P.port) {
    disp.P = seg.P;
  } else {
    flag = 0;
  }
  if (seg.Q1.pin && seg.Q1.port) {
    disp.Q1 = seg.Q1;
  } else {
    flag = 0;
  }
  if (seg.Q2.pin && seg.Q2.port) {
    disp.Q2 = seg.Q2;
  } else {
    flag = 0;
  }
  if (seg.Q3.pin && seg.Q3.port) {
    disp.Q3 = seg.Q3;
  } else {
    flag = 0;
  }
  return flag;
}

/**
 * @brief this function print nums to 7seg indicator
 * @retval none
 */
```

```
void my_7seg_disp_num(int num) {
  if (flag) {
    for (int pos = 0; pos < 3; pos++) {
      set_pos(-1);
      print_digit(num % 10);
      set_pos(pos);
      HAL_Delay(1);
      num /= 10;
    }

  }
}

/**
 * @brief this function enables specified segment
 * pos = 0, 1, 2 - enable Q1, Q2, Q3. Any other - disable all.
 * @retval none
 */
void static set_pos(int pos) {
  switch (pos) {
  case 2:
    HAL_GPIO_WritePin(disp.Q1.port, disp.Q1.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.Q2.port, disp.Q2.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.Q3.port, disp.Q3.pin, GPIO_PIN_RESET);
    break;
  case 1:
    HAL_GPIO_WritePin(disp.Q1.port, disp.Q1.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.Q2.port, disp.Q2.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.Q3.port, disp.Q3.pin, GPIO_PIN_RESET);
    break;
  case 0:
    HAL_GPIO_WritePin(disp.Q1.port, disp.Q1.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.Q2.port, disp.Q2.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.Q3.port, disp.Q3.pin, GPIO_PIN_SET);
    break;
  default:
    HAL_GPIO_WritePin(disp.Q1.port, disp.Q1.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.Q2.port, disp.Q2.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.Q3.port, disp.Q3.pin, GPIO_PIN_RESET);
    break;
  }
}

/**
 * @brief this function print digit to 7seg indicator
 * @retval none
 */
```

```
void static print_digit(int num) {
  switch (num) {
  case 0:
    HAL_GPIO_WritePin(disp.A.port, disp.A.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.B.port, disp.B.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.C.port, disp.C.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.D.port, disp.D.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.E.port, disp.E.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.F.port, disp.F.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.G.port, disp.G.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.P.port, disp.P.pin, GPIO_PIN_RESET);
    break;
  case 1:
    HAL_GPIO_WritePin(disp.A.port, disp.A.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.B.port, disp.B.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.C.port, disp.C.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.D.port, disp.D.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.E.port, disp.E.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.F.port, disp.F.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.G.port, disp.G.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.P.port, disp.P.pin, GPIO_PIN_RESET);
    break;
  case 2:
    HAL_GPIO_WritePin(disp.A.port, disp.A.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.B.port, disp.B.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.C.port, disp.C.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.D.port, disp.D.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.E.port, disp.E.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.F.port, disp.F.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.G.port, disp.G.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.P.port, disp.P.pin, GPIO_PIN_RESET);
    break;
  case 3:
    HAL_GPIO_WritePin(disp.A.port, disp.A.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.B.port, disp.B.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.C.port, disp.C.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.D.port, disp.D.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.E.port, disp.E.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.F.port, disp.F.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.G.port, disp.G.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.P.port, disp.P.pin, GPIO_PIN_RESET);
    break;
  case 4:
    HAL_GPIO_WritePin(disp.A.port, disp.A.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.B.port, disp.B.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.C.port, disp.C.pin, GPIO_PIN_SET);
```

```
  HAL_GPIO_WritePin(disp.D.port, disp.D.pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(disp.E.port, disp.E.pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(disp.F.port, disp.F.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.G.port, disp.G.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.P.port, disp.P.pin, GPIO_PIN_RESET);
  break;
case 5:
  HAL_GPIO_WritePin(disp.A.port, disp.A.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.B.port, disp.B.pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(disp.C.port, disp.C.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.D.port, disp.D.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.E.port, disp.E.pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(disp.F.port, disp.F.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.G.port, disp.G.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.P.port, disp.P.pin, GPIO_PIN_RESET);
  break;
case 6:
  HAL_GPIO_WritePin(disp.A.port, disp.A.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.B.port, disp.B.pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(disp.C.port, disp.C.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.D.port, disp.D.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.E.port, disp.E.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.F.port, disp.F.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.G.port, disp.G.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.P.port, disp.P.pin, GPIO_PIN_RESET);
  break;
case 7:
  HAL_GPIO_WritePin(disp.A.port, disp.A.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.B.port, disp.B.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.C.port, disp.C.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.D.port, disp.D.pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(disp.E.port, disp.E.pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(disp.F.port, disp.F.pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(disp.G.port, disp.G.pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(disp.P.port, disp.P.pin, GPIO_PIN_RESET);
  break;
case 8:
  HAL_GPIO_WritePin(disp.A.port, disp.A.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.B.port, disp.B.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.C.port, disp.C.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.D.port, disp.D.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.E.port, disp.E.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.F.port, disp.F.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.G.port, disp.G.pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(disp.P.port, disp.P.pin, GPIO_PIN_RESET);
  break;
case 9:
```

```
    HAL_GPIO_WritePin(disp.A.port, disp.A.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.B.port, disp.B.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.C.port, disp.C.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.D.port, disp.D.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.E.port, disp.E.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.F.port, disp.F.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.G.port, disp.G.pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(disp.P.port, disp.P.pin, GPIO_PIN_RESET);
    break;
  default:
    HAL_GPIO_WritePin(disp.A.port, disp.A.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.B.port, disp.B.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.C.port, disp.C.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.D.port, disp.D.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.E.port, disp.E.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.F.port, disp.F.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.G.port, disp.G.pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(disp.P.port, disp.P.pin, GPIO_PIN_RESET);
    break;
  }
}

/**
 * @brief this function print available chars to 7seg
 * @retval none
 */
void my_7seg_disp_char(char letter, int pos) {
  if (flag) {
    switch (pos) {
    case 0:
      print_letter(letter);
      break;
    case 1:
      print_letter(letter);
      break;
    case 2:
      print_letter(letter);
      break;
    }
  }
}

/**
 * @brief this function print letter to 7-seg indicator
 * @retval none
 */
void static print_letter(char letter) {
```

```
}

/**
 * @brief this function runs animation at 7-seg display
 * @retval none
 */
void my_7seg_disp_anim(int num) {

}

/**
 * @brief external interrupt to handle two buttons
 * @retval none
 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
  switch (GPIO_Pin) {
  case BTN_1_Pin:
    if (counter < 999) {
      counter++;
    } else {
      counter = 0;
    }
    break;
  case BTN_2_Pin:
    if (counter > 0) {
      counter--;
    } else {
      counter = 999;
    }
    break;
  };
}
```

5. Демонстрация работы

6. На рисунке 2 показана индикация после нажатия кнопки "−".

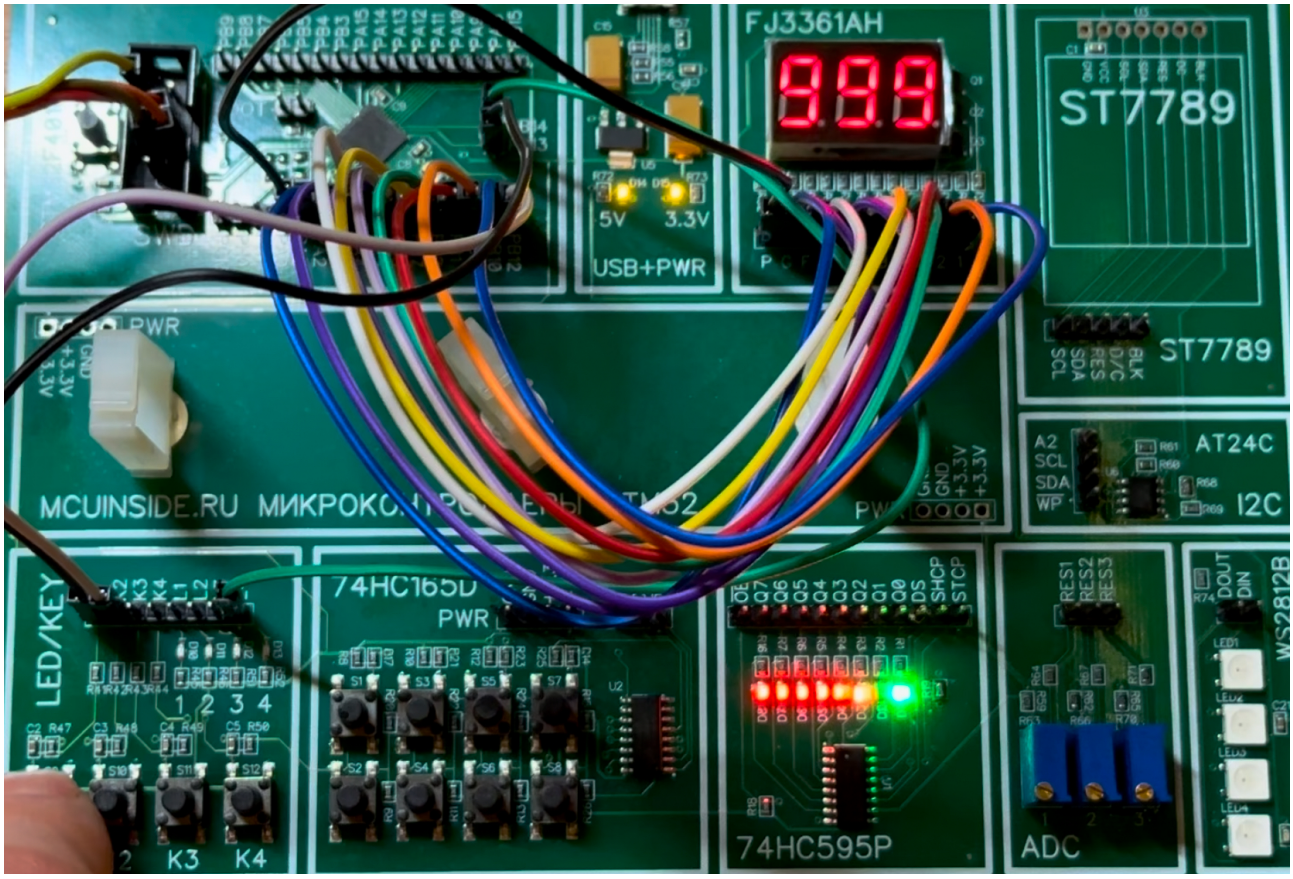

Рис. 2: Демонстрация работы

7. Дополнительные материалы

Демонстрация работы устройства проект расположены в папке на google диск по следующей ссылке: Материалы к ДЗ №02