Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Дисциплина: Проектирование вычислительных систем

Лабораторная работа 3

Вариант 2

Выполнили:
Колбасин Владислав Ильич,
Митичев Иван Дмитриевич,
группа P3416

Преподаватель:
Пинкевич Василий Юрьевич

2025 г.
Санкт-Петербург

Задание

Разработать программу, которая использует таймеры для управления яркостью светодиодов и излучателем звука (по прерыванию или с использованием аппаратных каналов). Блокирующее ожидание (функция HAL_Delay()) в программе использоваться не должно.

Стенд должен поддерживать связь с компьютером по UART и выполнять указанные действия в качестве реакции на нажатие кнопок на клавиатуре компьютера. В данной лабораторной работе каждая нажатая кнопка (символ, отправленный с компьютера на стенд) обрабатываются отдельно, ожидание ввода полной строки не требуется.

Для работы с UART на стенде можно использован один из двух вариантов драйвера (по прерыванию и по опросу) на выбор исполнителя. Поддержка двух вариантов не требуется.

Частота синхросигнала процессорного ядра и сигнала ШИМ для управления яркостью светодиодов (если используется) должны соответствовать указанным в варианте задания.

Вариант 2:

Реализовать настраиваемый пульт управления светодиодами боковой панели. Пульт должен иметь два основных режима: рабочий режим и режим настройки.

Действия стенда при получении символов от компьютера в рабочем режиме:

Символ	Действие
«1» – «9»	Зажигание светодиода в соответствии с настройками пульта. Светодиод горит до ввода следующего символа. Предыдущий светодиод выключается. Настройки по умолчанию: «1» – зеленый, 10 % яркости; «2» – зеленый, 40 % яркости; «3» – зеленый, 100 % яркости; «4» – желтый, 10 % яркости; «5» – желтый, 40 % яркости; «6» – желтый, 100 % яркости; «7» – красный, 10 % яркости;

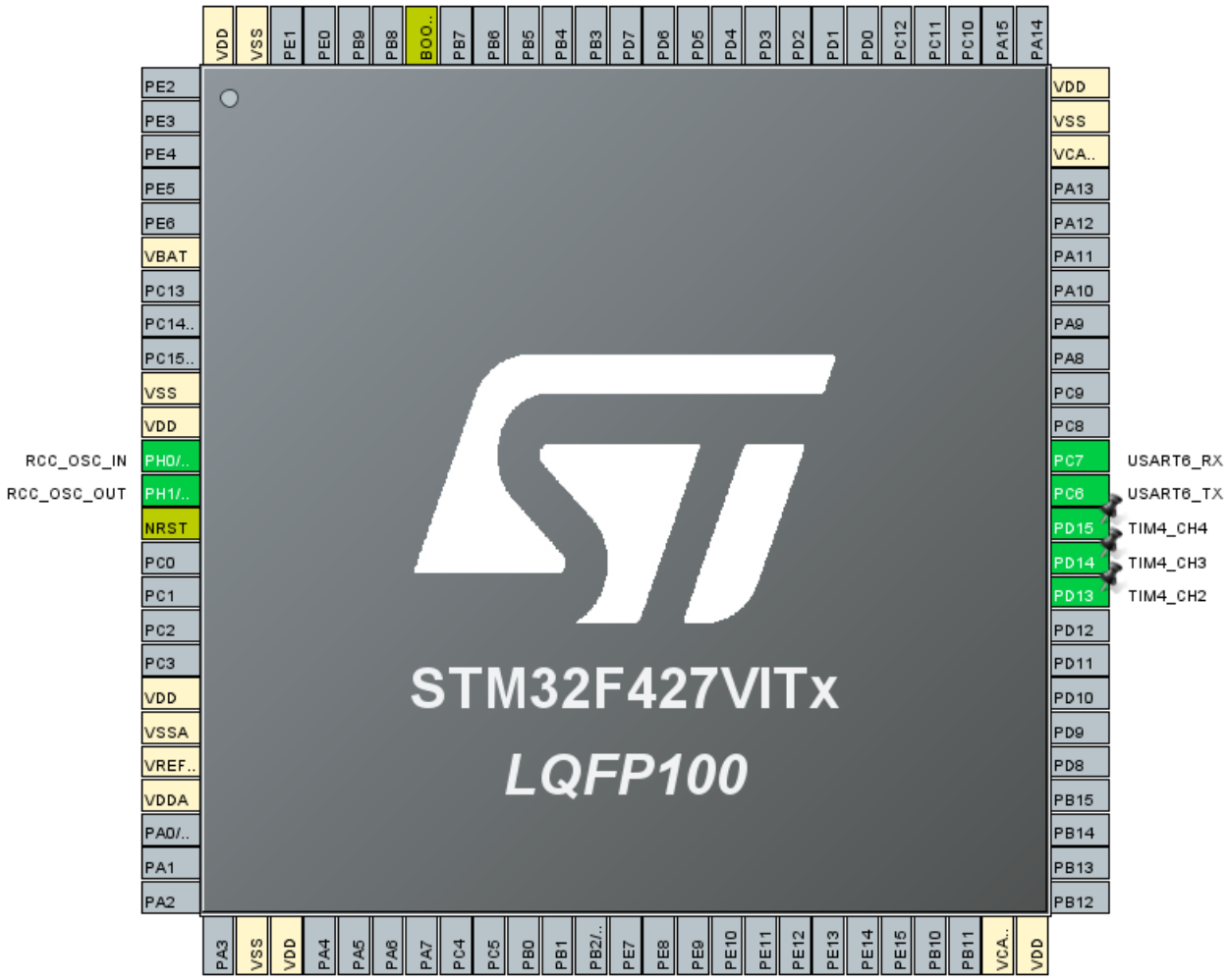
	«8» – красный, 40 % яркости; «9» – красный, 100 % яркости.
«0»	Погасить все светодиоды.
«Enter»	Вход в режим настройки.

После входа в меню настройки сначала надо ввести символ от «1» до «9», настройки для которого требуется изменить. Далее символом «a», «b» или «c» выбирается светодиод (зеленый, желтый, красный соответственно). После этого несколькими нажатиями кнопок «+» и «-» задается коэффициент заполнения от 0 до 100 % с шагом 10 %. По нажатию кнопки «Enter» сохраняется новая настройка и происходит переход в рабочий режим.

По вводу каждого символа в UART должно выводиться сообщение о том, какой режим активирован, или текущие вводимые настройки.

Частота процессорного ядра – 45 МГц, частота ШИМ-сигнала для светодиодов – 250 Гц.

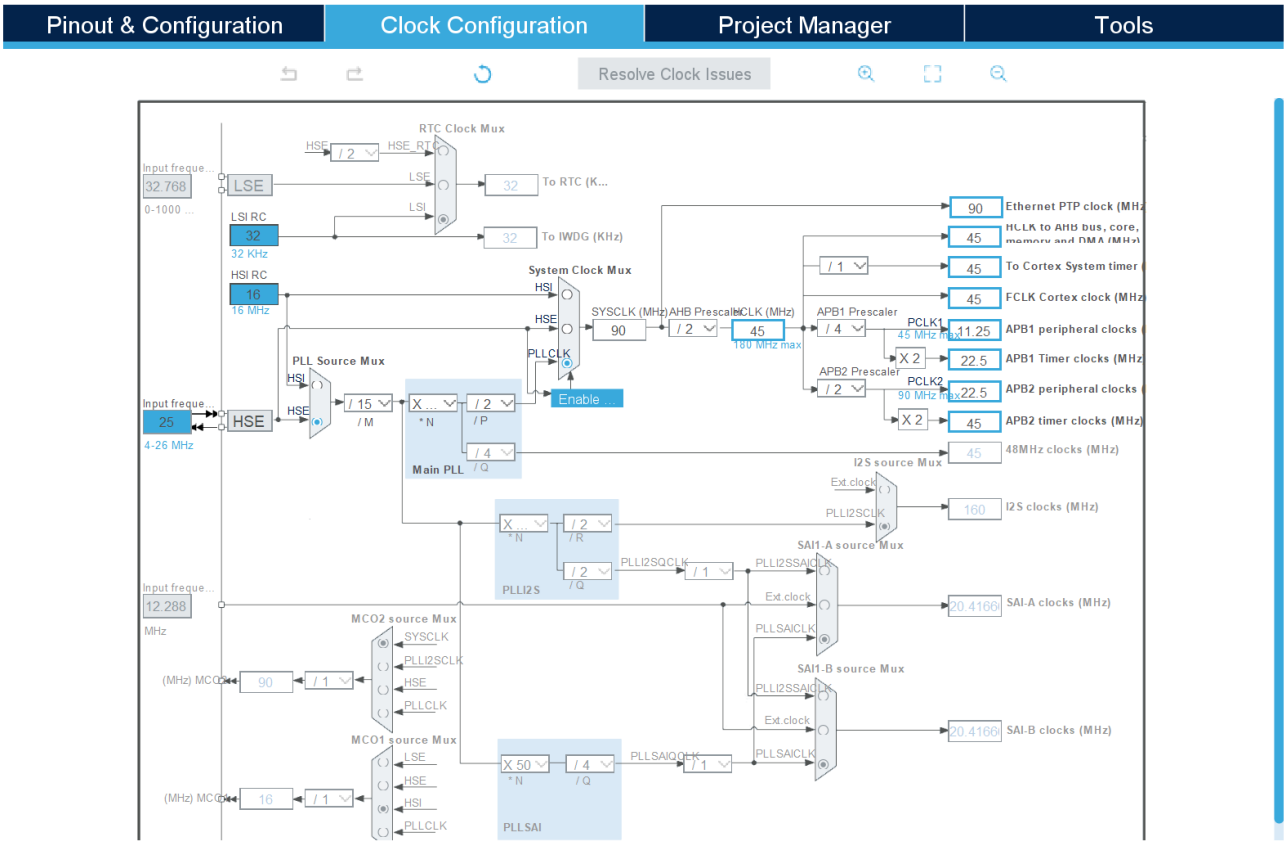
Конфигурация пинов



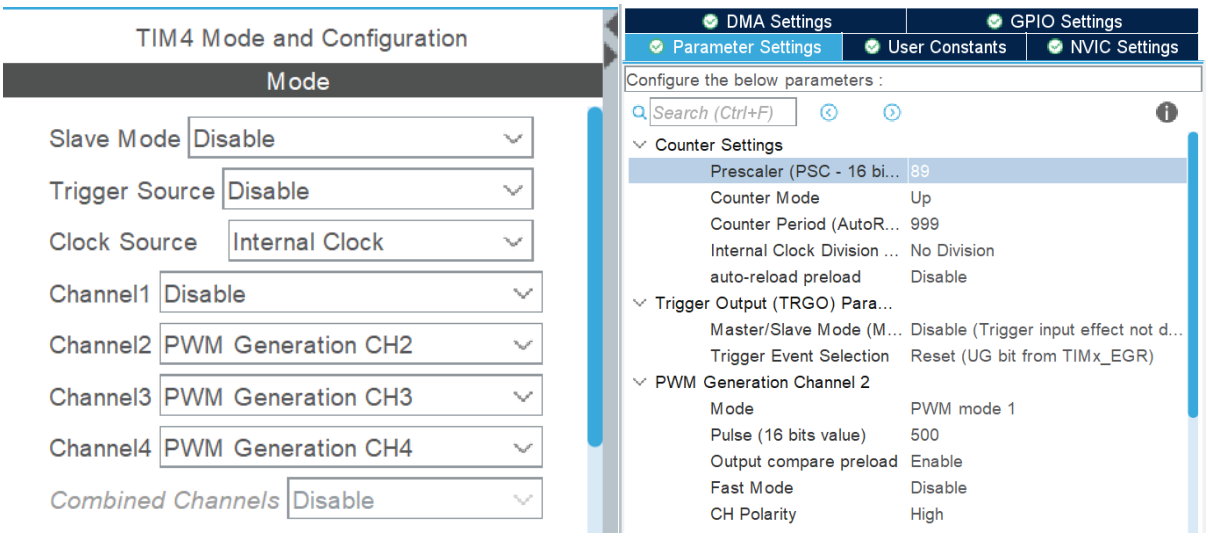
Описание контактов

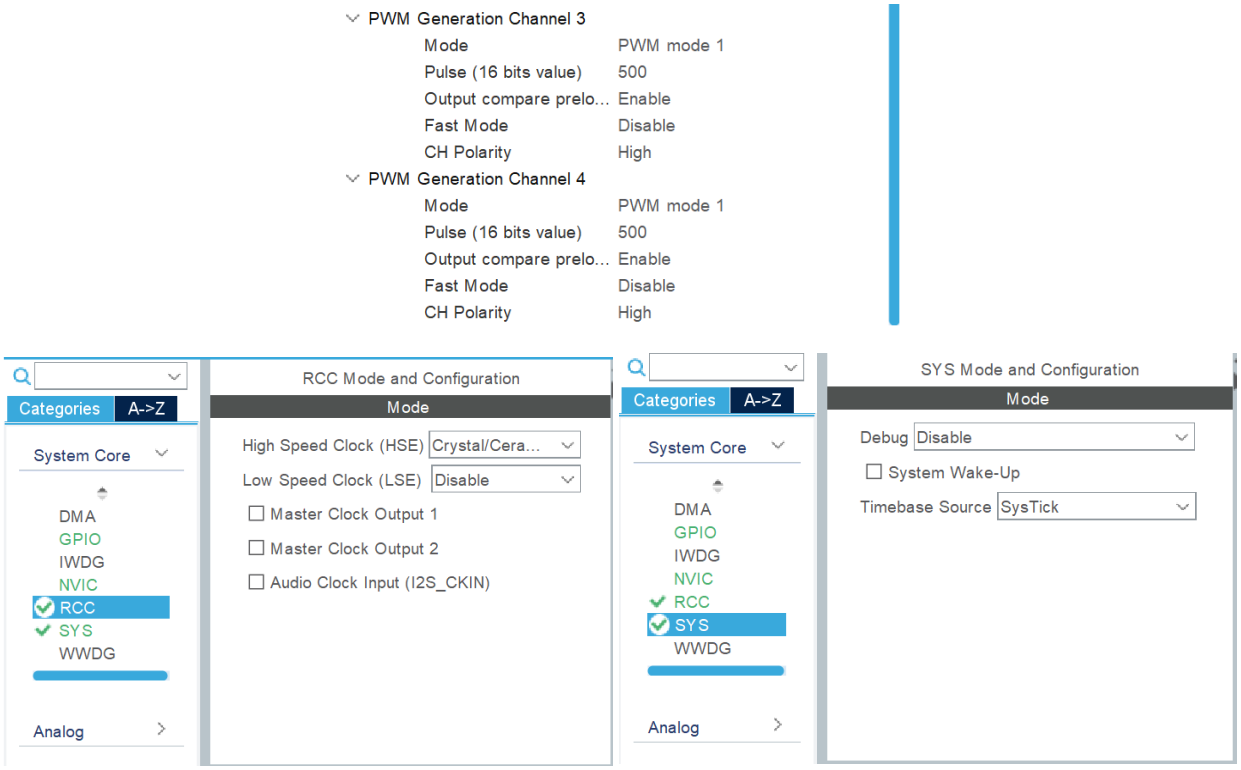
- **PH0** – RSS_OSC_IN
- **PH1** – RSS_OSC_OUT
- **PD13** – TIM4_CH2 (зеленый светодиод)
- **PD14** – TIM4_CH3 (желтый светодиод)
- **PD15** – TIM4_CH4 (красный светодиод)
- **PC6** – USART6_TX
- **PC7** – USART6_RX

По условию задания необходимо выставить частоту процессорного ядра 45 МГц. Тогда по шине APB1 частота таймера будет 22.5 МГц (в программе используется TIM4).



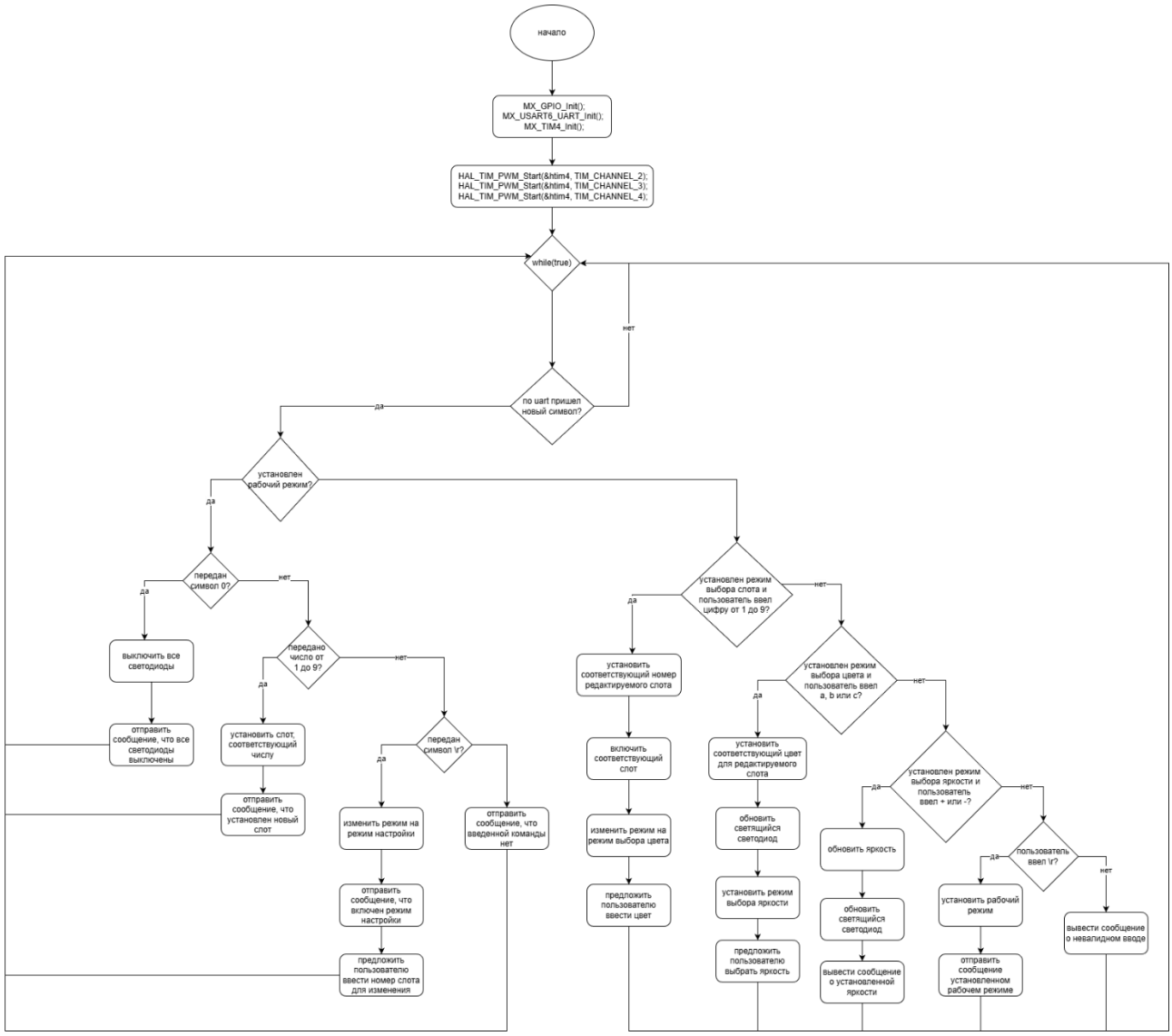
Рассчитаем значения для прескейлера и AutoReload Register. По условию частота ШИМ-сигнала для светодиодов должна быть 250 Гц. Значит нужно уменьшить частоту таймера в 22500000/250 = 90000 раз. Выберем прескейлер 89 и ARR = 999:




PWM Generation Channel 3
Mode PWM mode 1
Pulse (16 bits value) 500
Output compare prelo... Enable
Fast Mode Disable
CH Polarity High
PWM Generation Channel 4
Mode PWM mode 1
Pulse (16 bits value) 500
Output compare prelo... Enable
Fast Mode Disable
CH Polarity High
Categories A->Z
System Core
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
RCC Mode and Configuration
Mode
High Speed Clock (HSE) Crystal/Cera...
Low Speed Clock (LSE) Disable
Master Clock Output 1
Master Clock Output 2
Audio Clock Input (I2S_CKIN)
Categories A->Z
System Core
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
SYS Mode and Configuration
Mode
Debug Disable
System Wake-Up
Timebase Source SysTick

Блок-схемы

Основная программа:



Код программы

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"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "led_driver.h"
#include "usart_drv.h"
#include <stdio.h>
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
typedef enum
{
  MODE_WORK,
  MODE_CONFIG_SLOT,
  MODE_CONFIG_COLOR,
  MODE_CONFIG_BRIGHTNESS
} Mode_t;
Mode_t mode = MODE_WORK;

typedef enum
{
  LED_GREEN,
  LED_YELLOW,
  LED_RED
} LedColor_t;

typedef struct
{
  LedColor_t color;
```

```
  uint8_t duty;
} LedSlot_t;

LedSlot_t slots[9] = {
    {LED_GREEN, 10}, {LED_GREEN, 40}, {LED_GREEN, 100}, {LED_YELLOW, 10}, {LED_YELLOW,
40}, {LED_YELLOW, 100}, {LED_RED, 10}, {LED_RED, 40}, {LED_RED, 100}};

uint8_t selected_slot = 0;

/* 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 */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
void uart_println(const char *s)
{
  USART_DRV_TxStr("\n");
  USART_DRV_TxStr(s);
  USART_DRV_TxStr("\n");
}
void set_led_pwm(LedSlot_t cfg)
{
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, 0);
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_3, 0);
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_4, 0);

  uint32_t value = cfg.duty * 10;

  switch (cfg.color)
  {
  case LED_GREEN:
    __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, value);
    break;
  case LED_YELLOW:
    __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_3, value);
    break;
  case LED_RED:
    __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_4, value);
    break;
  }
}
```

```
static uint8_t edit_slot = 0;

void handle_config_char(char c)
{
  if ((mode == MODE_CONFIG_SLOT) && (c >= '1' && c <= '9'))
  {
    edit_slot = c - '1';
    set_led_pwm(slots[edit_slot]);
    uart_println("Select LED color: a(green), b(yellow), c(red)");
    mode = MODE_CONFIG_COLOR;
  }
  else if ((mode == MODE_CONFIG_COLOR) && (c == 'a' || c == 'b' || c == 'c'))
  {
    switch (c)
    {
    case 'a':
      slots[edit_slot].color = LED_GREEN;
      break;
    case 'b':
      slots[edit_slot].color = LED_YELLOW;
      break;
    case 'c':
      slots[edit_slot].color = LED_RED;
      break;
    }
    set_led_pwm(slots[edit_slot]);
    mode = MODE_CONFIG_BRIGHTNESS;
    uart_println("Use +/- to change brightness, Enter to save");
  }
  else if ((mode == MODE_CONFIG_BRIGHTNESS) && (c == '+' || c == '-'))
  {
    int delta = (c == '+') ? 10 : -10;
    int result = slots[edit_slot].duty + delta;
    if (result > 100)
    {
      result = 100;
    }
    else if (result < 0)
    {
      result = 0;
    }
    slots[edit_slot].duty = result;
    set_led_pwm(slots[edit_slot]);

    char buf[32];
    sprintf(buf, "Brightness: %d%%", slots[edit_slot].duty);
    uart_println(buf);
  }
  else if (c == '\r')
  {
    uart_println("Saved, back to WORK mode");
    mode = MODE_WORK;
  }
  else
  {
    uart_println("Invalid input");
  }
```

```
}

/* 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();
  MX_USART6_UART_Init();
  MX_TIM4_Init();
  /* USER CODE BEGIN 2 */
  USART_DRV_Init(&huart6);
  USART_DRV_EnableInterrupts(true);
  USART_DRV_SetEcho(true);

  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2);
  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3);
  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4);

  set_led_pwm(slots[selected_slot]);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    uint8_t rx;
    while (USART_DRV_PollGetByte(&rx))
```

```
    {
      if (mode != MODE_WORK)
      {
        handle_config_char(rx);
        continue;
      }
      switch (rx)
      {
      case '0':
        __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, 0);
        __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_3, 0);
        __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_4, 0);
        uart_println("All switched off");
        break;
      case '1' ... '9':
        selected_slot = rx - '1';
        set_led_pwm(slots[selected_slot]);
        char buf[32];
        sprintf(buf, "Set mode %d", selected_slot + 1);
        uart_println(buf);
        break;
      case '\r':
        mode = MODE_CONFIG_SLOT;
        USART_DRV_TxStr("CONFIG_MODE\nSelect slot (from 1 to 9):\n");
        break;
      default:
        uart_println("UNKNOWN COMMAND");
        break;
      }
    }
    /* 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_SCALE3);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
```

```
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 15;
  RCC_OscInitStruct.PLL.PLLN = 108;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  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_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV2;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
}

/* 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 */
```

Вывод

В ходе лабораторной работы мы применили на практике шим сигнал различной скважности для управления яркостью светодиодов, научились настраивать пины на выход тактового генератора, рассчитывать прескейлер и ARR.