МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

Базовая кафедра авиационных приборов, интерфейсов и систем

ОТЧЁТ ПО ПРАКТИКЕ
ЗАЩИЩЁН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ
Заведующий базовой кафедрой АПИиС
д-р техн. наук, доцент — подпись, дата — Хвощ С. Т.
должность, уч. степень, звание — подпись, дата — инициалы, фамилия

ОТЧЁТ ПО ПРАКТИКЕ

вид практики: производственная

тип практики: технологическая

на тему индивидуального задания: Разработка программ для микроконтроллера STM32103C8T6

выполнен: Гридиным Артёмом Максимовичем
фамилия, имя, отчество обучающегося в творительном падеже

по направлению подготовки: 09.03.01 (код) — Информатика и вычислительная техника (наименование направления)

направленности: 02 (код) — компьютерные технологии, системы и сети (наименование направленности)

Обучающийся группы № 4143 (номер) — 23.08.24 (подпись, дата) — А. М. Гридин (инициалы, фамилия)

Санкт–Петербург 2024

Содержание

1. Задание на практику

Написать программы для выполнения поставленных задач.

2. Описание задач и требований к программам

1) Светодиод

Описание задачи:

- Подключите светодиод к микроконтроллеру.
- Используйте функцию `HAL_Delay()` для создания задержек.
- Напишите программу, которая заставляет светодиод мигать с интервалом 1 секунда (1000 миллисекунд).

2) Кнопка

Описание задачи:

- Подключите 2 светодиода и 2 кнопки к микроконтроллеру.
- Напишите программу, которая при нажатии на первую кнопку зажигает первый светодиод, а при нажатии на вторую кнопку гасит второй светодиод.
- При отпускании кнопок светодиоды должны возвращаться в начальное состояние (первый не горит, второй горит).

3) Таймер

Описание задачи:

- Используйте таймер для управления миганием светодиода.
- Напишите программу, которая заставляет светодиод мигать с интервалом 1 секунда (1000 миллисекунд) через таймер.

4) ШИМ (Широтно-импульсная модуляция)

Описание задачи:

- Подключите светодиод и используйте ШИМ для управления его яркостью.
- Напишите программу, которая плавно уменьшает и

увеличивает яркость светодиода.

5) Энкодеры

Описание задачи:

- Подключите энкодер и 4 светодиода.
- Напишите программу, которая зажигает светодиоды в зависимости от направления вращения энкодера. При вращении вправо загорается следующий справа светодиод, при вращении влево — следующий слева.

6) Энкодер и светодиод

Описание задачи:

- Подключите энкодер и светодиод.
- Используйте энкодер для управления яркостью светодиода через ШИМ.

7) АЦП (Аналого-цифровой преобразователь)

Описание задачи:

- Напишите программу, использующую АЦП для измерения напряжения с делителя напряжения и фоторезистора.
- Используйте полученные данные для какого-либо действия, например, управления яркостью светодиода.

8) Матричная клавиатура

Описание задачи:

- Подключите матричную клавиатуру 3x3 и матричный индикатор.
- Напишите программу, которая отображает номер нажатой клавиши на индикаторе.
- Оптимизируйте код сканирования клавиатуры, сделав его максимально кратким.

9) Подключение нагрузки через транзистор 2N3904

Описание задачи:

- Подключите лампочку на 12V через транзистор 2N3904.
- Напишите программу, которая управляет включением и выключением лампочки через кнопку.

3. Пример программы

Пример кода для выполнения задачи «Матричная клавиатура»:

```
/* 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 */

/* 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 */
GPIO_InitTypeDef GPIO_InitStructPrivate = {0};
  uint32_t previousMillis = 0;
```

```
  uint32_t currentMillis = 0;
  uint8_t keyPressed = 0;

/* 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 */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5, 1);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* 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};

  /** 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_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
                    |GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, GPIO_PIN_RESET);
```

```
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB,              GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5,
GPIO_PIN_RESET);

  /*Configure GPIO pins : PA0 PA1 PA2 PA3
                         PA4 PA5 PA6 */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
                      |GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6;
  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 : PB3 PB4 PB5 */
  GPIO_InitStruct.Pin = GPIO_PIN_3|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);

  /*Configure GPIO pins : PB6 PB7 PB8 PB9 */
  GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8|GPIO_PIN_9;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
```

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

/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  currentMillis = HAL_GetTick();
  if (currentMillis - previousMillis > 10) {
    /*Configure GPIO pins : PB6 PB7 PB8 PB9 to GPIO_INPUT*/
    GPIO_InitStructPrivate.Pin                                                    =
GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8|GPIO_PIN_9;
    GPIO_InitStructPrivate.Mode = GPIO_MODE_INPUT;
    GPIO_InitStructPrivate.Pull = GPIO_NOPULL;
    GPIO_InitStructPrivate.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStructPrivate);

    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4|GPIO_PIN_5, 0);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, 1);
    if(GPIO_Pin == GPIO_PIN_7 && HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_7))
Disp_CA(3);
    else if(GPIO_Pin == GPIO_PIN_8 && HAL_GPIO_ReadPin(GPIOB,
GPIO_PIN_8)) Disp_CA(6);
    else if(GPIO_Pin == GPIO_PIN_9 && HAL_GPIO_ReadPin(GPIOB,
GPIO_PIN_9)) Disp_CA(9);


    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3|GPIO_PIN_5, 0);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 1);
    if(GPIO_Pin == GPIO_PIN_6 && HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_6))
```

```
Disp_CA(0);
   else if(GPIO_Pin == GPIO_PIN_7 && HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_7)) Disp_CA(2);
   else if(GPIO_Pin == GPIO_PIN_8 && HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_8)) Disp_CA(5);
   else if(GPIO_Pin == GPIO_PIN_9 && HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_9)) Disp_CA(8);

   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3|GPIO_PIN_4, 0);
   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 1);
   if(GPIO_Pin == GPIO_PIN_7 && HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_7)) Disp_CA(1);
   else if(GPIO_Pin == GPIO_PIN_8 && HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_8)) Disp_CA(4);
   else if(GPIO_Pin == GPIO_PIN_9 && HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_9)) Disp_CA(7);

   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5, 1);
   /*Configure GPIO pins : PB6 PB7 PB8 PB9 back to EXTI*/
   GPIO_InitStructPrivate.Mode = GPIO_MODE_IT_RISING;
   GPIO_InitStructPrivate.Pull = GPIO_PULLDOWN;
   HAL_GPIO_Init(GPIOB, &GPIO_InitStructPrivate);
   previousMillis = currentMillis;
  }
}

void Disp_CA(unsigned char var_1)
{
      unsigned                                                    char
disp_arr[]={0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x10,0x08,0x03,0x46,
```

```
0x21,0x06,0x0e,0xff};
        unsigned char data1, bit_var;
        data1=disp_arr[var_1];
        bit_var=data1 & 0x01;
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_0,bit_var);

        bit_var=(data1>>1) & 0x01;
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_1,bit_var);

        bit_var=(data1>>2) & 0x01;
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_2,bit_var);

        bit_var=(data1>>3) & 0x01;
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_3,bit_var);

        bit_var=(data1>>4) & 0x01;
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_4,bit_var);

        bit_var=(data1>>5) & 0x01;
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,bit_var);

        bit_var=(data1>>6) & 0x01;
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_6,bit_var);
}
/* 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                  */
```

4. Результаты выполнения

Запуск микроконтроллера осуществляется в среде Proteus 8.

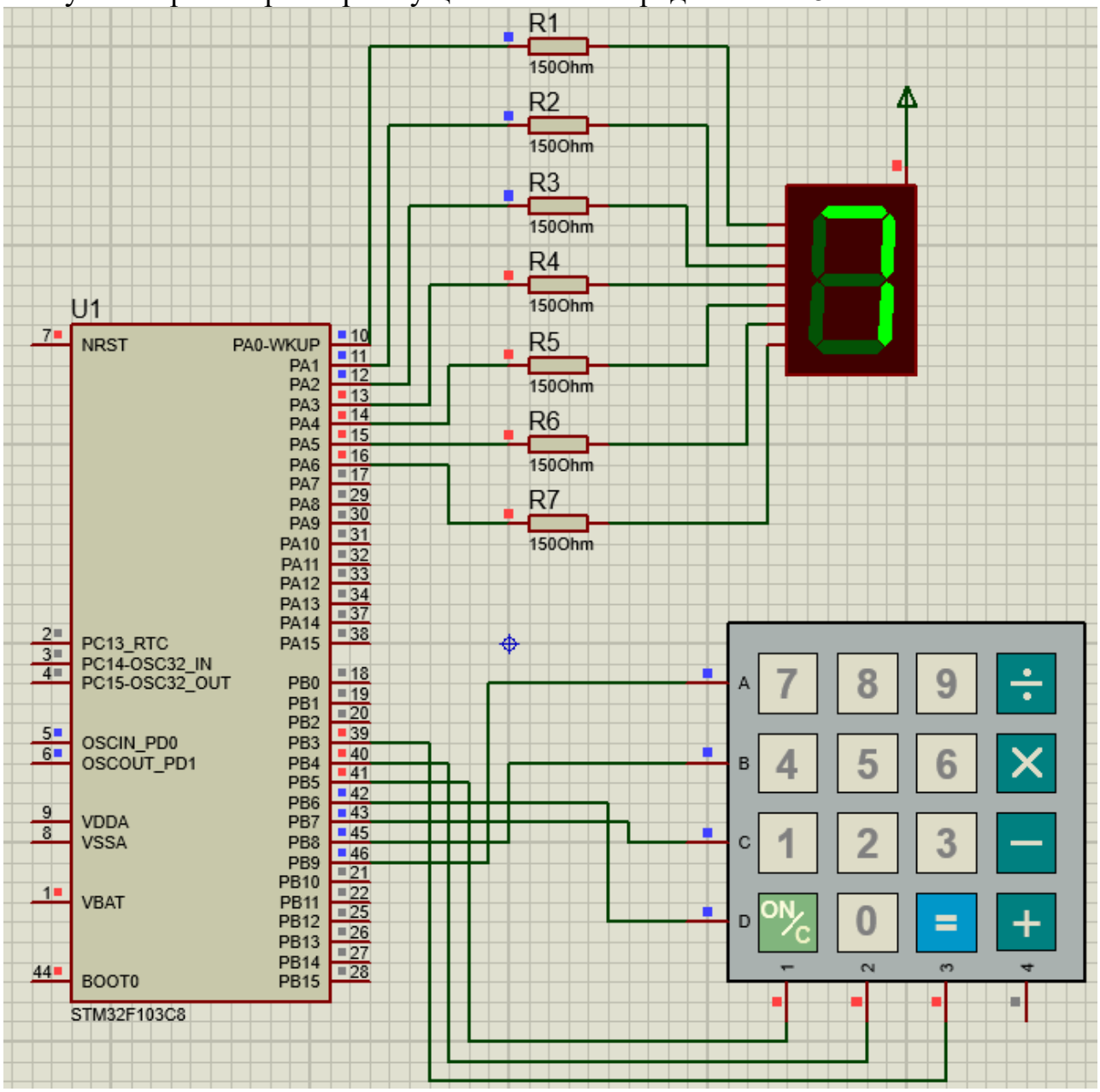

Рисунок 1 – Пример работы схемы для задачи «Матричная клавиатура» в среде Proteus 8

Результат работы всех программ предоставлен в виде видеоматериалов по ссылке: URL: https://drive.google.com/drive/folders/1cRvHdvwiR05op55vUWk2yqEpixA7-koN?usp=sharing

5. Используемые технические средства для разработки и выполнения программ

Для разработки, отладки и выполнения программ были использованы следующие технические средства:

• Язык программирования C: Код написан на языке программирования C, который широко используется для разработки системного и прикладного программного обеспечения;

• Библиотека HAL;

• Программы STM32CubeIDE и Proteus 8;

• Микроконтроллер STM32F103C8T6;

6. Выводы по работе

Были разработаны программы для решения описанных задач. Результаты работы соответствуют поставленным задачам.

Список использованных источников

1 STM32F103C8T6 Datasheet [Электронный ресурс], URL - https://www.st.com/resource/en/datasheet/stm32f103c8.pdf

2 Уроки по STM32 [Электронный ресурс], URL - https://www.micropeta.com/

3 74LS148 Datasheet [Электронный ресурс], URL - https://www.futurlec.com/74LS/74LS148.shtml