МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Инженерная школа информационных технологий и робототехники

Отделение автоматизации и робототехники

15.04.06 «Мехатроника и робототехника»

ЛАБОРАТОРНАЯ РАБОТА №1

Знакомство с лабораторной базой. Изучение и настройка GPIO. Применение библиотеки CMSIS при программировании микроконтроллеров

по дисциплине:

Микропроцессоры и микроконтроллеры

Исполнитель:

студент группы 8EM51 Федоренко Д. В. 07.11.2025

Руководитель:

преподаватель Поберёзкин Н. И. 07.11.2025

Томск – 2025

Содержание

Задание

Собрать схему с включением 4-х светодиодов и 2-х кнопок. К одному порту подключены кнопка и светодиод параллельно. 1) Включает светодиоды поочередно, в случае включения всех светодиодов процесс начинается сначала. 2) Каждым нажатием меняет режим работы порта 1-ой кнопки. Соответственно, включает и выключает отдельный светодиод, подключенный к тому же порту, что и эта кнопка.

Ход работы

Схема подключения

Согласно заданию необходимо соединить отладочную плату с четырьмя светодиодами и двумя кнопками, при этом одна кнопка и светодиод должны быть подключены к одному пину.

В качестве отладочной платы будет использоваться Blue Pill на микроконтроллере STM32F103C8T6, которая будет устанавливаться на специальную площадки для сборки схемы без пайки. К светодиодам будет подсоединяться резистор с целью ограничения тока, приходящего на светодиод. Одна из кнопок с светодиодом подключается параллельно к 1 порту.

В приложении А можно наблюдать принципиальную электрическую схему подключения отладочной платы к остальным компонентам. В то же время собранная физическая схема продемонстрирована на рисунке 1.

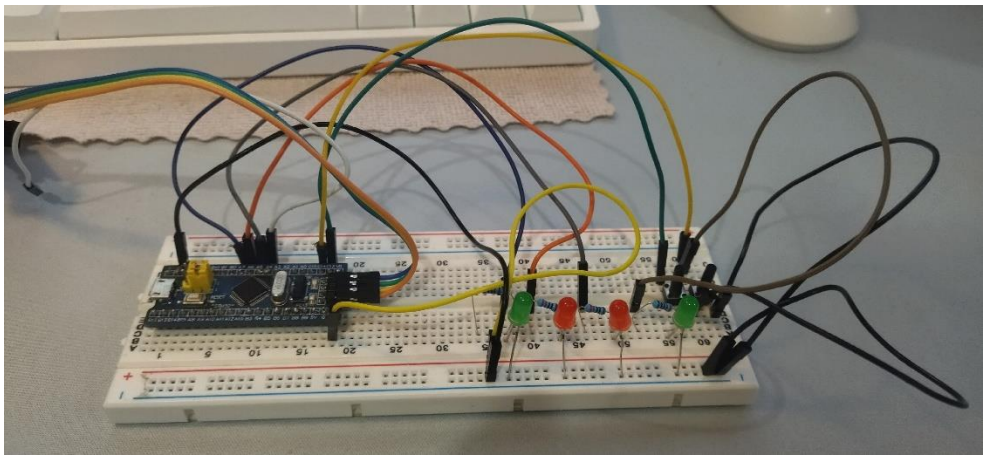


Рисунок 1 - Собранная схема

Подготовка рабочей среды

Предварительно были установлены все основные вспомогательные программы, на основании методических указаний представленных к лабораторной работе. В частности это программы:

- Make – это программа для упрощения процесса компиляции программ, фактически, запуская массу команд вместо человека.
- GNU Arm Embedded GCC – компилятор для преобразования кода C/C++ в код, понятный для ARM процессов и контроллеров.
- OpenOCD – позволяет выгружать код в память микроконтроллера и запускать режим отладки.
- MCUViewer – открытый бесплатный проект для построения графиков изменений переменных в микроконтроллерах.

В качестве IDE на основе методических материалов был взят VS Code, назначение которого будет упрощение процесса написания кода и ведения проекта.

После IDE была подключена к GitHab, чтобы было возможно выгружать программный код в облачный сервис, что даёт возможность получить доступ к проекту и его истории изменений (контролю версии) с любого персонального компьютера.

Ввиду применения неоригинального микроконтроллера, потребовалась дополнительная процедура по изменению ID допустимых контроллеров в файле конфигураций для OpenOCD, так как изначально он содержит лишь ID оригинальных контроллеров.

Разбор Datasheet на микроконтроллер STM32F103C8T6

Для программирования микроконтроллера на более низком уровне, необходимо понимать и знать, в какие адреса и что необходимо записывать, чтобы получить определённый результат.

Сам datasheet был взят с официального сайта STM под микроконтроллер STM32F103C8T6.

Для начала запустим тактирование пинов микроконтроллера, в частности набора портов GPIO A и C. Это необходимо для взаимодействия с этими пинами и их дальнейшего конфигурирования. Для этого рассмотрим раздел 7.3.7 на странице 112 и поймём, что необходимо выставить биты 4 и 2 в высокое состояние по адресу 0x4002 1000 (из раздела 3.3 на страниц 50) + 0x18, где первая часть адреса – начало раздела с адресами RCC, а 0x18 – адрес, отвечающий за секцию с тактированием групп пинов.

После рассмотрим разделы с регистрами группы GPIO, информация об этом содержится в разделе 9.2 на страницах 171-174. На странице 171 есть таблица с регистрами CRL (для пинов 0-7 с адресом 0x40010800) и CRH (для пинов 8-15 с адресом 0x40010804), где по 32 бита отвечают за порты, в частности выделяется по 4 бита на порт. Эти 4 бита позволяют сконфигурировать порт. Первый и второй бит в серии позволяют выбрать частоту работы пина, а третий и четвёртый бит выставляют режим работы порта. Все комбинации описаны на странице 161 в таблице 20.

Описанное выше актуально для группы A, группа C настраивается по другим адресам, найти которые можно на тех же страницах.

Настройка кнопок происходит аналогичным образом, только необходимо дополнительно изменить биты ODR для активации подтяжек, информацию об этих битах можно найти по тем же страницам.

Программирование микроконтроллера

Инициализация пинов

На основе методических указаний и с целью упрощения процесса работы с кодом, проект был разбит на несколько файлов, в частности на фалы main и init, в файле main содержится основное тело программы, в то время как в init содержится код для инициализации пинов в микроконтроллере.

Согласно заданию, было необходимо реализовать инициализацию пинов через макросы и путём прямого обращения к регистрам.

В качестве примера инициализации напрямую на основе метода обращения к регистрам будет проведена настройка пина PA5, к которому будет подключён светодиод через резистр. На рисунке 2 проведён пример кода для инициализации этого пина.

```
void Init_GPIO(void) {
    // PA5 (LED2) - ИНИЦИАЛИЗАЦИЯ ЧЕРЕЗ РЕГИСТРОВ
    // Output push-pull 50MHz: MODE=11 (0x3), CNF=00
    *(volatile uint32_t*)(0x40010800U + 0x00) = (*(volatile uint32_t*)(0x40010800U + 0x00) & ~(0xF << 20)) | (0x3 << 20);
```

Рисунок 2 - Инициализация LED2 через обращение к памяти

Все другие пины инициализируются через макросы в этом же файле init.c.

```
// PA6-PA7 (LED3, LED4)
GPIOA->CRL = (GPIOA->CRL & ~(0xF << 24)) | (PIN_MODE_OUTPUT_50MHz << 24);
GPIOA->CRL = (GPIOA->CRL & ~(0xF << 28)) | (PIN_MODE_OUTPUT_50MHz << 28);


// PC13 как вход с подтяжкой
GPIOC->CRH = (GPIOC->CRH & ~(0xF << 20)) | ((PIN_MODE_INPUT | CNF_PULL_UPDOWN) << 20);
GPIOC->ODR |= (1 << 13);

// PC14 как вход с подтяжкой
GPIOC->CRH = (GPIOC->CRH & ~(0xF << 24)) | ((PIN_MODE_INPUT | CNF_PULL_UPDOWN) << 24);
GPIOC->ODR |= (1 << 14);
```

Рисунок 3 - Инициализация остальных пинов через макросы

Сами макросы расположены в init.h и представлены в приложении Б.

Основное тело программы

Под основным телом программы подразумевается код, находящийся в файле main.c и отвечающий за логику переключения светодиодов, обработку кнопок.

Фактически, основной код должен будет производить обработку двух кнопок, подключённых к разным пинам. В частности, происходи следующее:

Кнопка 1 отвечает за переключение режимов работы светодиодов, с каждым нажатием зажигая новый светодиод (с 2 по 4), но если все эти светодиоды уже горя, то данные светодиоды гасятся. Это возможно благодаря конфигурации портов на выдачу напряжения (порты светодиодов) и конфигурирование на получение сигнала (порт кнопки). Соответственно, чтобы зажечь светодиоды – портам разрешается выдавать напряжение, если гасить светодиоды – запрещаем выдавать напряжение.

Кнопка 2 отвечает за изменение режима работы кнопки 1, делая так, что если кнопка не нажата – светодиод, подключённый параллельно к кнопке, горит. Если кнопка нажата – светодиод гаснет.

Выполнение данного пункта возможно благодаря двум конфигурациях пина, к которым подключены светодиод и кнопка. Если кнопка в режиме управления светодиодами (группой), то у нас пин сконфигурирован в режиме приёма сигнала. Если же кнопка в режиме управления одним светодиодом, то пин переконфигурируется на выдачу напряжения, которое питает светодиод. В случай зажимания кнопки, напряжение переходит на неё, так как данный электрический контур имеет меньшее сопротивление, соответственно, светодиод остаётся без питания и гаснет. Более детально это показано на рисунке 4 и рисунке 5.

Программу можно рассмотреть в виде алгоритма, представленного в приложении Г.

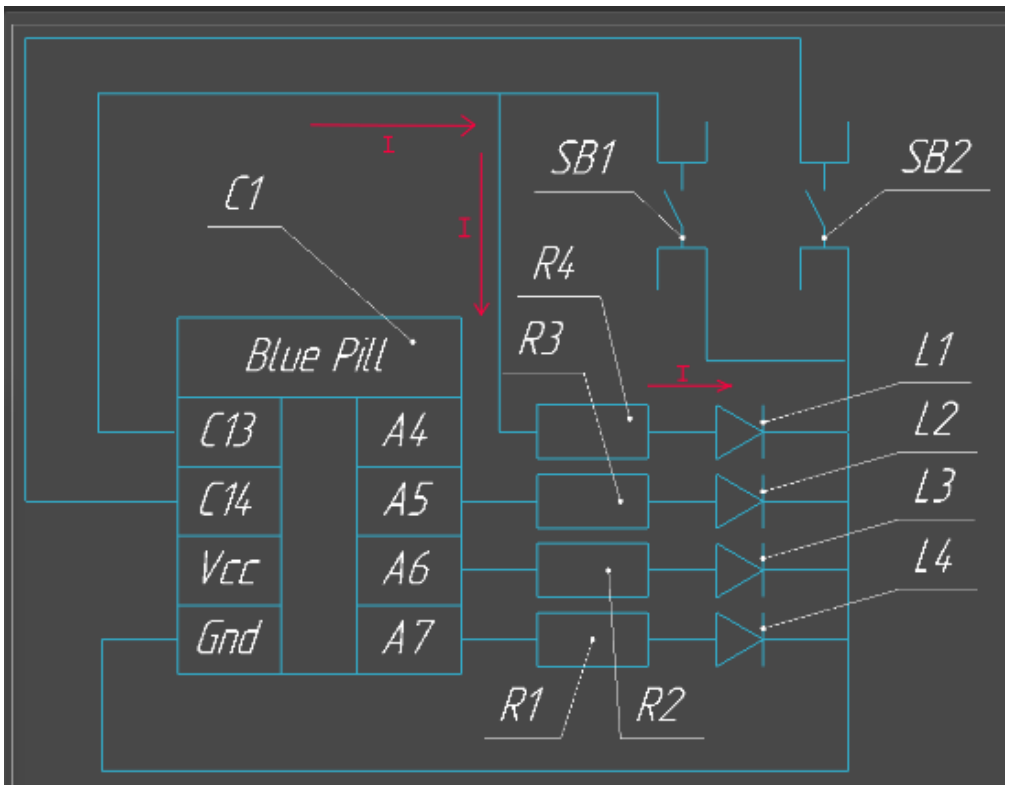


Рисунок 4 - Направление тока при отпущенной кнопки 1

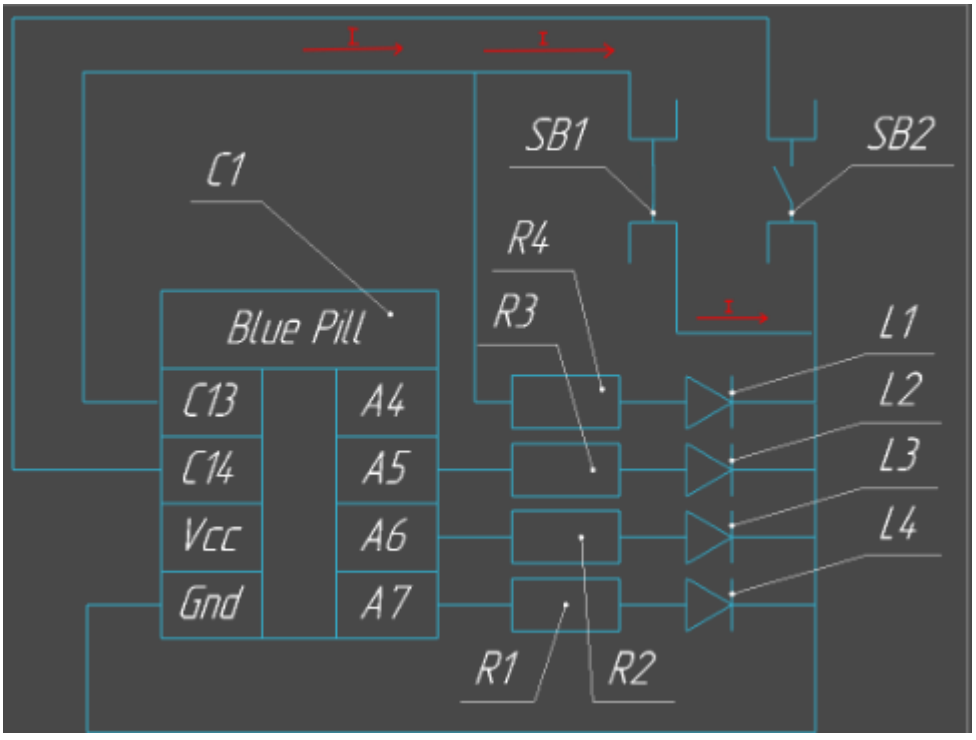


Рисунок 5 - Направление тока при зажатой кнопки 1

Рассмотрим более подробно код на основе представленной блок схемы. Первым делом вызывается функция Init_RCC, которая подключает тактирование пинов микроконтроллера из группы GPIO A и C.

Следующая функция Init_GPIO производит инициализацию всех портов, участвующих в проекте, в частности это инициализация портов PA3, PA5, PA6 как выходные, но без выдачи напряжения. И порты PA0 и PC14 как порты с подтяжкой (для обработки нажатий кнопок).

После происходит процесс инициализации переменных-состояний, назначение которых отслеживать состояния кнопок и светодиодов.

Следующим этапом программа ожидает нажатия одной из кнопок. Рассмотрим сперва процессы, происходящие при нажатии кнопки 1. В случай первого нажатия (при условии, что кнопка в состоянии управления группой светодиодов) – кнопка будет увеличивать счётчик, который определяет состояние группы светодиодов в зависимости от своего значения:

- При значении 1: зажигается светодиод LED2;
- При значении 2: зажигается светодиод LED2 и LED3;
- При значении 3: зажигается светодиод LED2 и LED3, LED4;
- При значении 4: погасить все светодиоды LED2, LED3, LED4.

Для зажигания и гашения светодиодов применялось установление пинов в состояние разрешения выдачи напряжения или запрет на выдачу напряжения.

Если была нажата кнопка 2, которая позволяет менять режим работы кнопки 1, то вызывается функция, назначение которой – изменить параметры конфигурации пинов с режима подтяжки (получения сигналов) на конифигурацию выходного пина. Или же производить обратное переключение, если кнопка 2 была нажата повторна.

Снятие данных с STMViewer

Применение программы STMViewer позволяет удобно в режиме реального времени видеть значение переменных в виде графиков изменений их во времени.

Запустим систему и запустим программу, в итоге нажимая кнопки, мы можем получить следующий графики переключения группы светодиодов из-за нажатия кнопки.

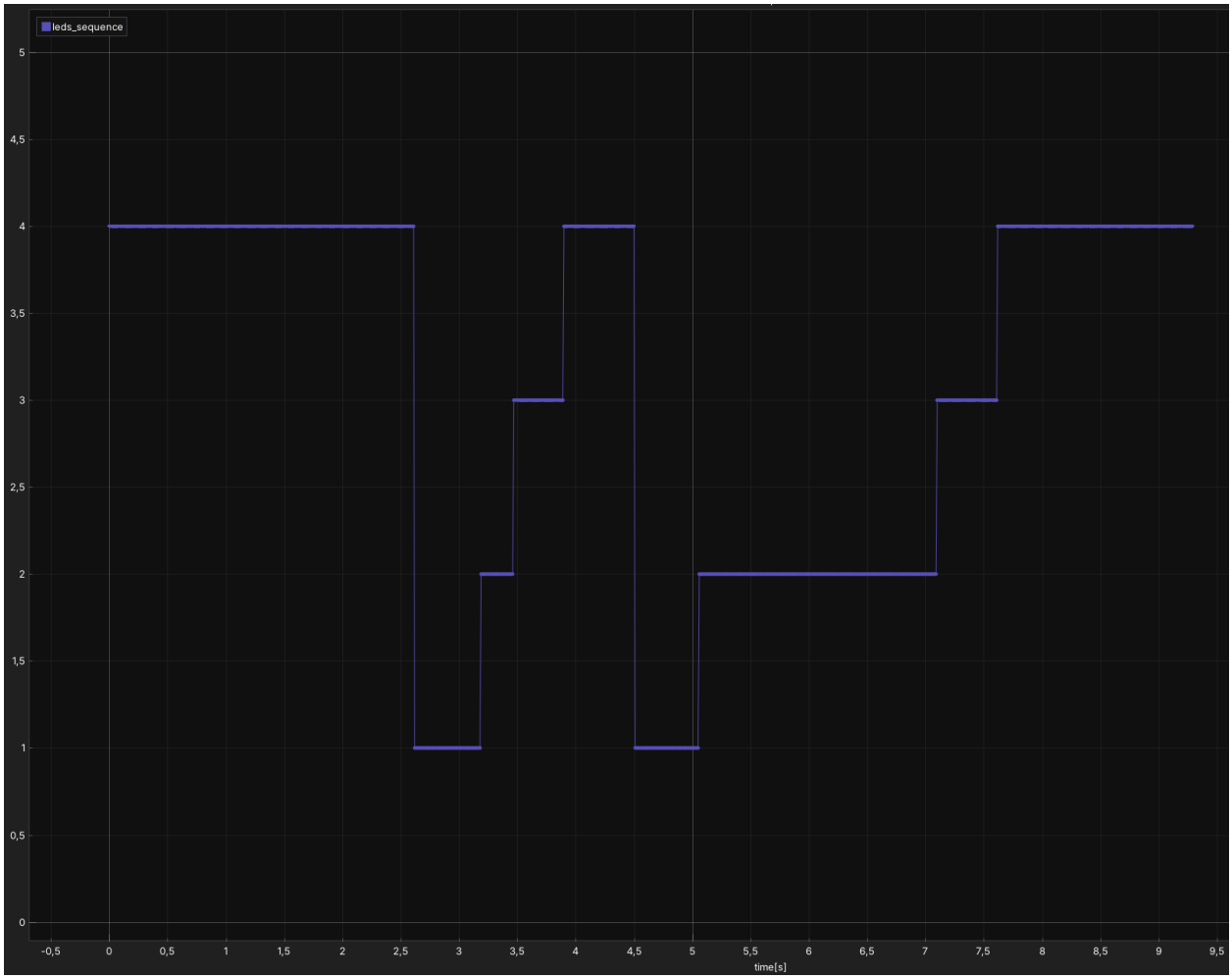


Рисунок 6 - Переключение группы светодиодов

Состояние 4 – ни один светодиод не горит, состояние 3 – горят все три светодиода, состояние 2 – горят два светодиода, состояние 1 – горит один светодиод.

Теперь рассмотрим график переключения кнопки, где состояние 0 – кнопка не нажата, 1 – нажата.

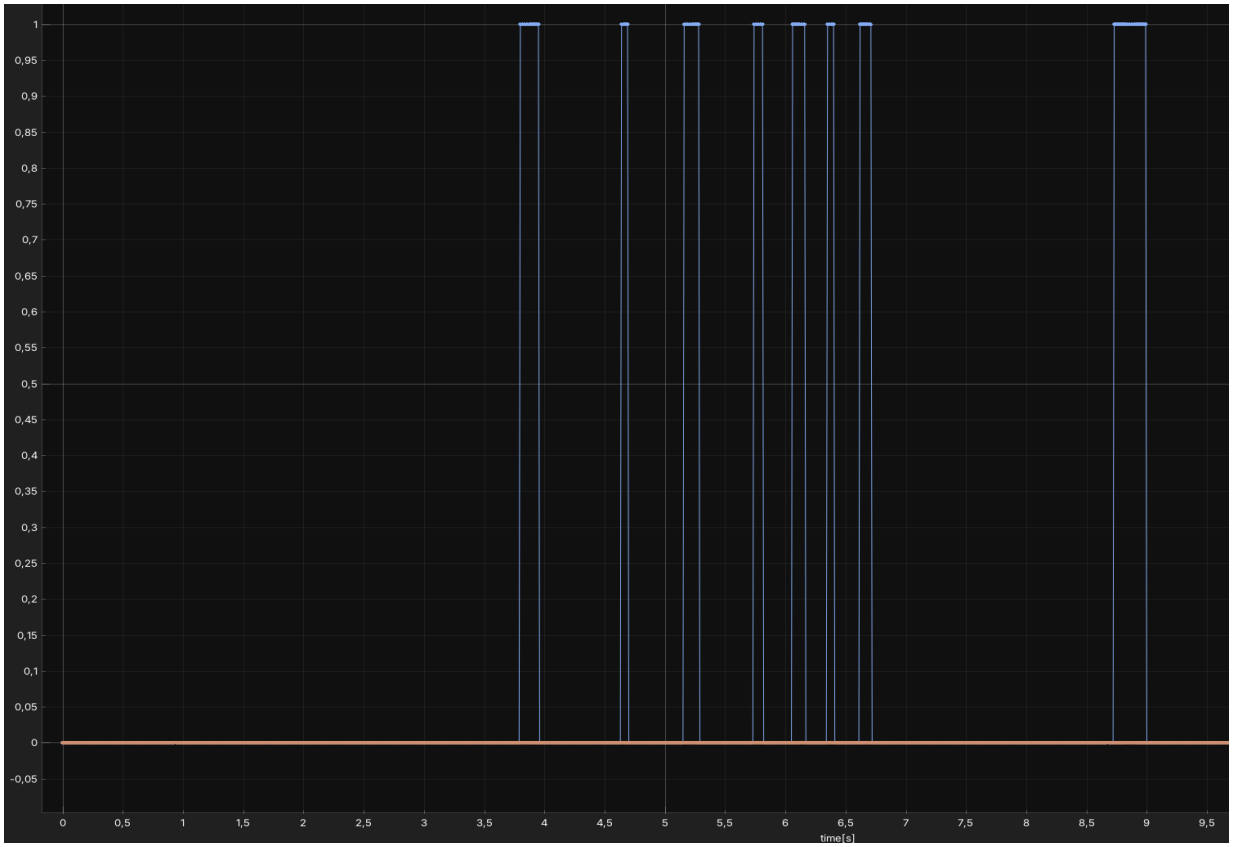


Рисунок 7 - График нажатий кнопки 1

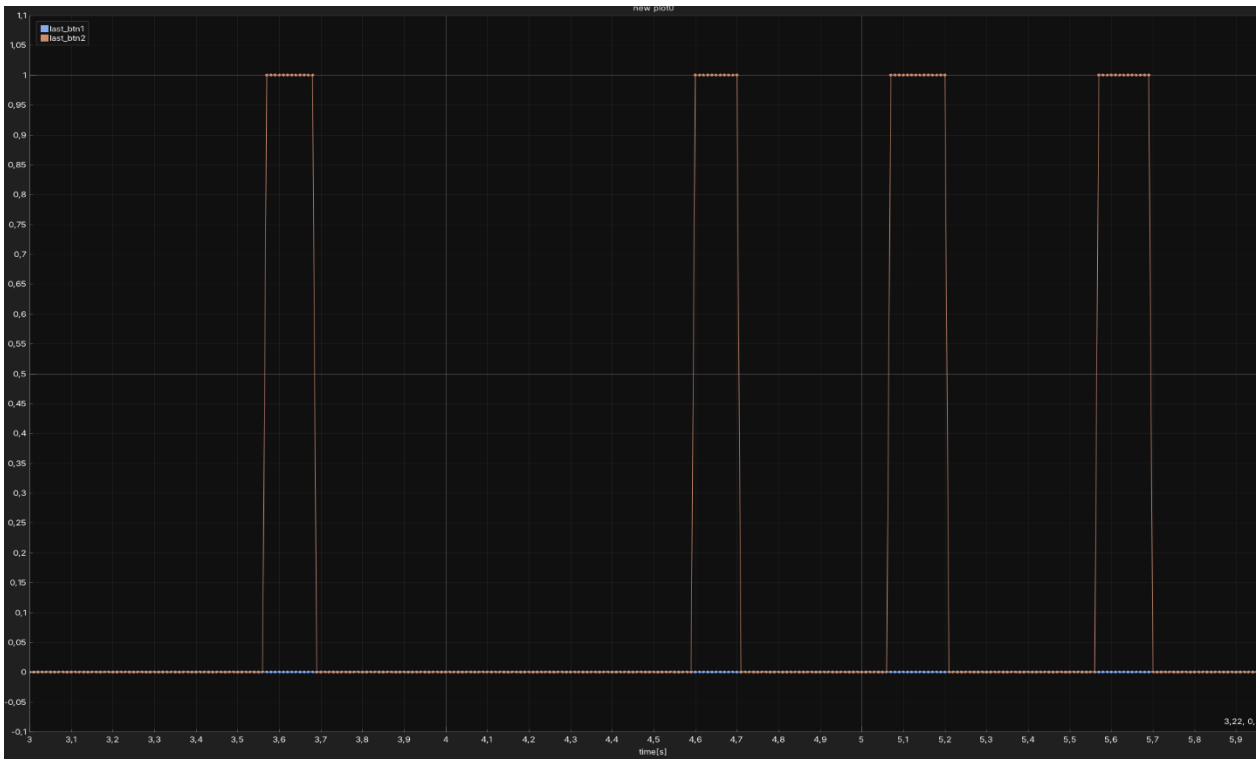


Рисунок 8- График нажатий кнопки 2

Аналогично был сделан график нажатий кнопки 2. То есть, система полностью работоспособна и можно наблюдать, как она реагирует на нажатия кнопки.

Для чистоты опыта предоставляется график на рисунке 9, в котором все переменные выводятся на один экран параллельно.

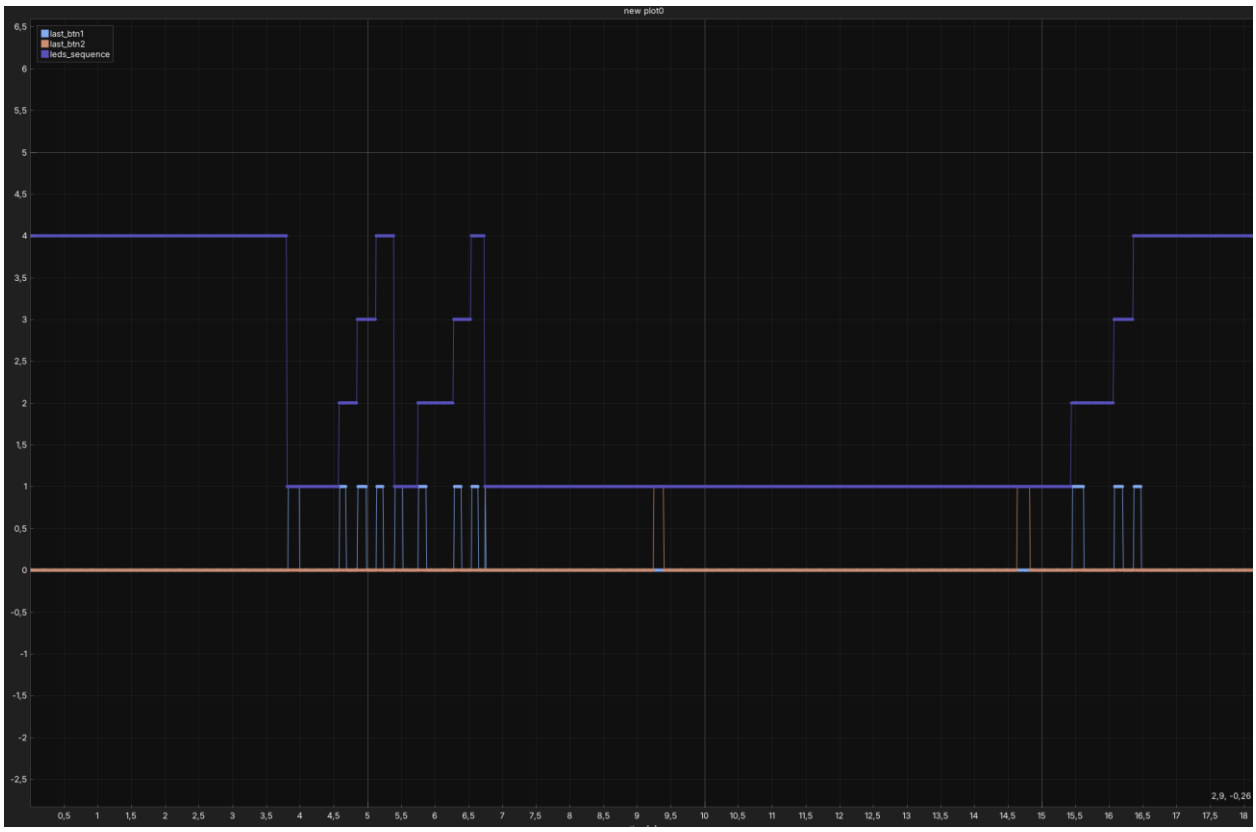


Рисунок 9 - Паралельный вывод всех переменных

Так же здесь будет оставлена ссылка на основной репрезиторий проекта.

На этом основная задача лабораторной работы выполнена.

Дополнительное задание

В качестве дополнительного задания было необходимо реализовать возможность настройки частоты мигания трёх светодиодов. Для этого кнопка 1 должна была отвечать за выбор режима мигания светодиода (короткое нажатие) и за выбор мигающего светодиода (длительное нажатие).

Кнопка 2 должна выдавать разрешение на работу этим же трём светодиодам.

Основной сложностью задания была реализация механизма различия длительного и короткого нажатия, а также реализация изменения частот мигания группы светодиодов без применения аппаратных таймеров микроконтроллера.

Решение обоих задач осуществляется через реализацию программных таймеров, представлявших из себя во многом просто циклы с счётчиком. В задачи определения зажатия кнопки – мы находимся в цикле счётчика ровно до тех пор, пока идёт нажатие (но не более 1000 итераций, что равняется около 0,1 секунде), после выход из цикла из-за прекращения нажатия кнопки, происходит анализ значения счётчика. Если оно менее 1000, то было осуществлено кратковременное нажатие, если же более 1000, то было длительное нажатие. После вывода об длительности нажатия счётчик обнуляется и вызываются функции, которые должны вызываться при данных видах нажатия.

Вторая задача, мигание светодиодов, реализуется через общий цикл, внутри которого находится несколько условий, зависящих от одного счётчика внутри этого цикла. Если счётчик переступает порог, то происходит переключение состояния светодиода в противоположное, соответственно, он гаснет или загорается. Каждое из условий отвечает за отдельный светодиод.

Фактически, эта вся программная реализация дополнительного задания.

Вывод по работе

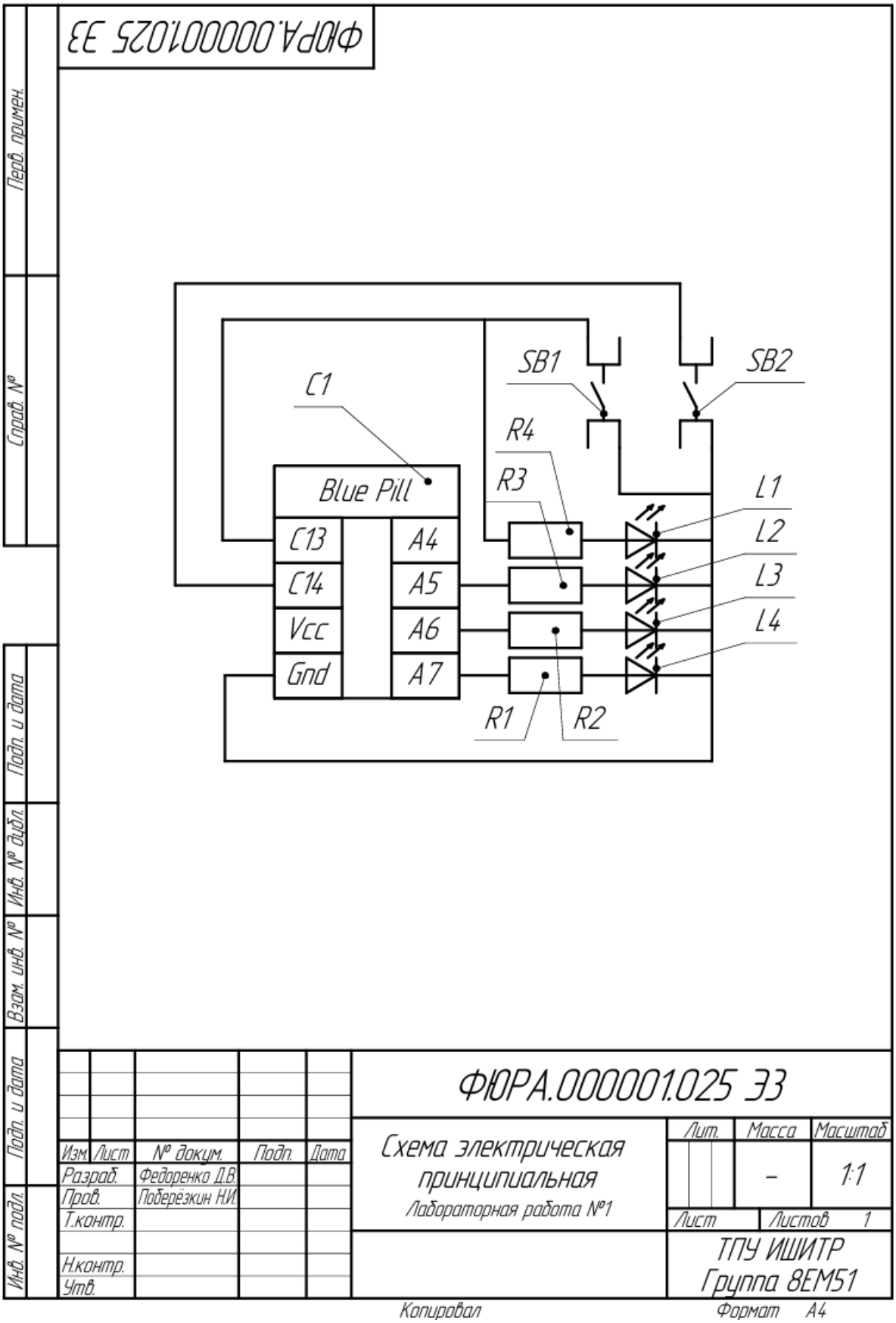
В рамках лабораторной работы была настроена программная среда для написания программ под микроконтроллеры серии STM32 и их прошивки без применения программных продуктов компании STM, с применением makefile-ов.

Был освоен метод программирования микроконтроллеров серии STM32 с использованием регистров и макросов, что позволяет управлять контроллером на самом низком доступном уровне. При этом Datasheet становится основным средством получения информации об назначении тех или иных регистров.

Также было выполнено задание по управлению светодиодами с помощью нескольких кнопок, задание было получено согласно своему варианту. После чего было выполнено дополнительное задание, заключающееся в управлении теми же светодиодами с возможностью изменения частоты мигания светодиодов.

Приложение А

Электрическая принципиальная схема



Формат	Зона	Поз.	Обозначение	Наименование	Кол.	Приме-чание
				Документация		
A4			ФЮРА.000001.025 Э3	Схема электрическая прициписльая		
				Лабораторная работа №1		
				Компоненты		
			C1	Blue Pill STM32F103C8T6	1	
			L1–L4	Светодиод GNL–3012	4	
			SB1–SB2	Кнопка KLS7–TS6601	2	
			R1–R4	Резистор 1к Ом	4	

Перв. примен.

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Федоренко Д.В.		
Пров.		Поберёзкин Н.И.		
Н.контр.				
Утв.				

ФЮРА.0000025.002

Схема электрическая принципиальная

Лабораторная работа №1

Лит.	Лист	Листов
		1

ТПУ ИШИТР
Группа 8ЕМ51

Копировал

Формат А4

Приложение Б

Макросы пинов

```c
#ifndef INIT_H
#define INIT_H

#include <stdint.h>

// Базовые адреса
#define GPIOA_BASE          0x40010800U
#define GPIOC_BASE          0x40011000U
#define RCC_BASE            0x40021000U

// Структуры периферии
typedef struct {
    volatile uint32_t CRL;
    volatile uint32_t CRH;
    volatile uint32_t IDR;
    volatile uint32_t ODR;
    volatile uint32_t BSRR;
    volatile uint32_t BRR;
    volatile uint32_t LCKR;
} GPIO_TypeDef;

typedef struct {
    volatile uint32_t CR;
    volatile uint32_t CFGR;
    volatile uint32_t CIR;
    volatile uint32_t APB2RSTR;
    volatile uint32_t APB1RSTR;
    volatile uint32_t AHBENR;
    volatile uint32_t APB2ENR;
    volatile uint32_t APB1ENR;
    volatile uint32_t BDCR;
    volatile uint32_t CSR;
} RCC_TypeDef;

// Макросы для доступа к периферии
#define GPIOA               ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOC               ((GPIO_TypeDef *) GPIOC_BASE)
#define RCC                 ((RCC_TypeDef *) RCC_BASE)

// Макросы для настройки пинов
#define PIN_MODE_INPUT      0x8
#define PIN_MODE_OUTPUT_10MHz 0x1
#define PIN_MODE_OUTPUT_2MHz  0x2
#define PIN_MODE_OUTPUT_50MHz 0x3

#define CNF_ANALOG          0x0
#define CNF_FLOATING        0x4
#define CNF_PULL_UPDOWN     0x8
#define CNF_OPEN_DRAIN      0xC

// Функции для настройки портов
void Init_RCC(void);
void Init_GPIO(void);

```

Приложение В

Основное тело программы

```c
#include "init.h"

void delay(uint32_t count) {
    for(volatile uint32_t i = 0; i < count; i++);
}

// Мигание встроенным светодиодом платы (PC13)
void blink_internal_led(uint8_t times) {
    // Временно переключаем PC13 в выход для мигания
    uint32_t temp_crh = GPIOC->CRH;
    GPIOC->CRH = (GPIOC->CRH & ~(0xF << 20)) | (0x3 << 20); // Push-pull выход

    for(int i = 0; i < times; i++) {
        GPIOC->BSRR = (1 << 13); // Включить светодиод (1 - выключен, т.к. подключен к
        delay(100000);
        GPIOC->BSRR = (1 << (13 + 16)); // Выключить светодиод (0 - включен)
        delay(100000);
    }

    // Возвращаем в режим входа с подтяжкой
    GPIOC->CRH = temp_crh;
    GPIOC->ODR |= (1 << 13); // Подтяжка к питанию
}

int main(void) {
    Init_RCC();
    Init_GPIO();

    uint8_t sequence_step = 0;
    uint8_t port_mode = 0; // 0=кнопка, 1=светодиод
    uint8_t last_btn1 = 1;
    uint8_t last_btn2 = 1;

    // Начальное состояние
    GPIOA->BSRR = 0xE0 << 16; // PA5-PA7 выкл

    // Убедимся что PC14 в режиме кнопки
    GPIOC->CRH = (GPIOC->CRH & ~(0xF << 24)) | (0x8 << 24); // Вход с подтяжкой
    GPIOC->ODR |= (1 << 14); // Подтяжка к питанию

    blink_internal_led(1); // 1 мигание - старт

    while(1) {
        uint8_t btn1 = !(GPIOC->IDR & (1 << 14)); // PC14
        uint8_t btn2 = !(GPIOC->IDR & (1 << 13)); // PC13

        // Кнопка 1 (PC14): смена режима
        if (btn1 && !last_btn1) {
            port_mode = !port_mode;
            blink_internal_led(3); // 3 мигания - режим сменился

            if (port_mode == 1) {
                // Переключаем в режим светодиода PUSH-PULL
```

```
                GPIOC->CRH = (GPIOC->CRH & ~(0xF << 24)) | (0x3 << 24); // Push-pull 50MHz
                // Включаем LED1 если должен гореть
                if (sequence_step >= 1 && sequence_step <= 4) {
                    GPIOC->BSRR = (1 << 14); // Включить (1 для push-pull)
                } else {
                    GPIOC->BSRR = (1 << (14 + 16)); // Выключить (0)
                }
            } else {
                // Возвращаем в режим кнопки
                GPIOC->CRH = (GPIOC->CRH & ~(0xF << 24)) | (0x8 << 24); // Вход
                GPIOC->ODR |= (1 << 14); // Подтяжка
            }

            delay(20000);
        }

        // Кнопка 2 (PC13): последовательность
        if (btn2 && !last_btn2) {
            sequence_step = (sequence_step + 1) % 5;
            blink_internal_led(2); // 2 мигания - последовательность сменилась

            // Выключить все PA5-PA7
            GPIOA->BSRR = 0xE0 << 16;

            // Включить нужные светодиоды
            switch(sequence_step) {
                case 1:
                    // Только LED1
                    if (port_mode == 1) GPIOC->BSRR = (1 << 14); // Включить LED1
                    break;
                case 2:
                    // LED1 + LED2
                    if (port_mode == 1) GPIOC->BSRR = (1 << 14); // LED1
                    GPIOA->BSRR = (1 << 5); // LED2
                    break;
                case 3:
                    // LED1 + LED2 + LED3
                    if (port_mode == 1) GPIOC->BSRR = (1 << 14); // LED1
                    GPIOA->BSRR = (1 << 5) | (1 << 6); // LED2, LED3
                    break;
                case 4:
                    // Все светодиоды
                    if (port_mode == 1) GPIOC->BSRR = (1 << 14); // LED1
                    GPIOA->BSRR = (1 << 5) | (1 << 6) | (1 << 7); // LED2, LED3, LED4
                    break;
                case 0:
                    // Все выключено
                    if (port_mode == 1) GPIOC->BSRR = (1 << (14 + 16)); // Выключить LED1
                    break;
            }

            delay(200000);
        }

        last_btn1 = btn1;
        last_btn2 = btn2;
        delay(10000);
    }
}
```

Приложение Г

Блок-схема алгоритма программы