

ГУАП

КАФЕДРА № 43

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

доцент, к.т.н

должность, уч. степень, звание

подпись, дата

А.А. Попов

инициалы, фамилия

Пояснительная записка к курсовой работе (проекту)

Гирлянда из 8-ми светодиодов – «Разбегающиеся огоньки»

по курсу: Программирование встроенных приложений

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

4031

подпись, дата

Бобошко А.С

инициалы, фамилия

Санкт-Петербург 2023

СОДЕРЖАНИЕ

Оглавление

СОДЕРЖАНИЕ.....	2
Задание на курсовое проектирование.....	3
Техническое задание на выбор.....	4
Основные требования.....	6
Схемы и алгоритм работы	7
Функции реализации режимов движения	13
Тестирование.....	18
Тесты для контроля соответствия прибора техническому заданию	18
Заключение.....	20
Список использованной литературы	21

Задание на курсовое проектирование

Гирлянда из 8-ми светодиодов – «Разбегающиеся огоньки»

Считать, что к линиям ПВВ В подключенные светодиоды, причем высокий уровень на линии зажигает светодиод, низкий гасит. Реализовать движение «огоньков» в одном из трех режимов с заданной скоростью. Один или два «огонька» по выбору пользователя. Скорость, режим движения (с отражением влево-вправо, разбегающиеся, сталкивающиеся) и количество огоньков (1, 2), задавать через PORTA и кнопки, созданные скриптом отладчика. Текущие значения настроек выводить в окно отладчика (Debug (printf) Viewer). Решить самостоятельно, какие кнопки создать и с какими функциями

Техническое задание на выбор

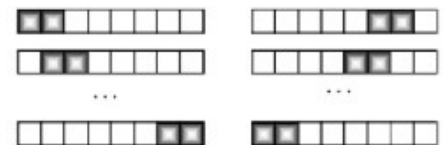
Поскольку курсовое проектирование выполняется удаленно, без отладочной платы, то необходимо выполнить проект на симуляторе в среде Keil uVision5. Таким образом, для имитации нажатия кнопок необходимо использовать скриптовый отладчик Keil.

Будем считать, что у нас подключено следующее периферийное оборудование: виртуальная клавиатура с кнопками «режим», «скорость», «Кол-во», меняющие режим работы, скорость движения и кол-во «огоньков» соответственно. Будем считать, что к линии ПВВ В подключены 8 светодиодов, которые формируют «гирлянду», светодиоды на которой переключаются в связи с определенной логикой (режимом).

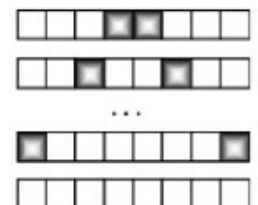
Согласно заданию, необходимо реализовать 3 режима переключения «огоньков»:

1. «Бегущий огонек»/«Отражающийся» - огонек начинает свой путь от левой части «гирлянды» и движется к правой, достигая крайнего правого положения, он движется обратно;
2. «Разбегающиеся огоньки» - 2 огонька начинают свой путь из центра «гирлянды» и движутся по направлению к крайнему левому и крайнему правому положениям. Достигая заданных позиций, «огоньки» гаснут;
3. «Сталкивающиеся огоньки» - 2 огонька начинают свой путь от крайнего левого и крайнего правого положений «гирлянды» и движутся на встречу друг к другу в центр. Достигая центра, огоньки «сталкиваются» образуя «взрыв», что зажигает все огоньки на ленте

○ «Бегущий огонек» из 2х светодиодов слева направо и обратно



○ «Разбегающиеся огоньки»



○ «Сталкивающиеся огоньки»

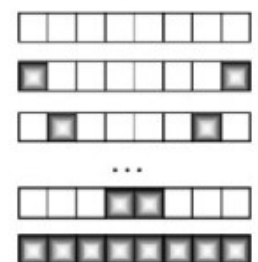


Рис 1. Иллюстрация работы 3-х режимов движения

ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

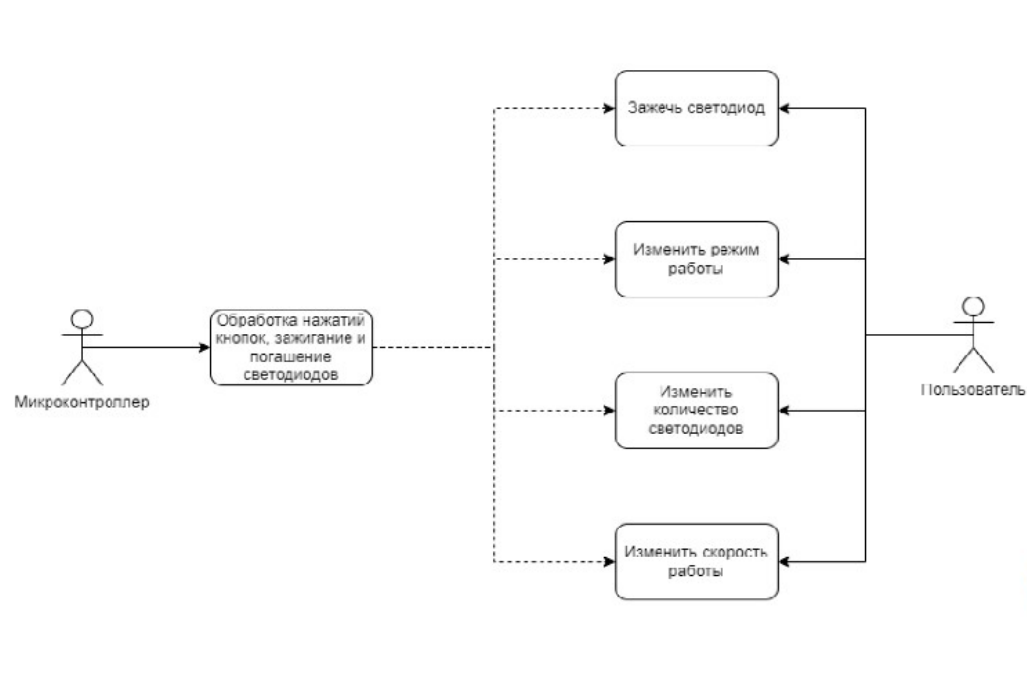


Рис 2. Диаграмма вариантов использования

Таблица 1. Назначение кнопок

Название кнопки	Режим	Скорость	Кол-во
Назначение	Последовательная смена режимов	Последовательная смена скоростей	Последовательное переключение кол-ва «огоньков»

Основные требования

Входы	Три кнопки соответствующие заявленным функциям
Выходы	Линия ПВВ В к которой подключены светодиоды
Функции	<p>При отсутствии ввода «гирлянда» работает в стандартном режиме «бегущие огоньки», с минимальной скоростью и одним «огоньком»</p> <p>При нажатии кнопки «режим» происходит изменение режима движения на следующий в списке (порядок бегущие, разбегающиеся, сталкивающиеся)</p> <p>При нажатии кнопки «скорость» происходит изменение скорости движения огоньков на следующую</p> <p>При нажатии кнопки «Кол-во» происходит изменение кол-ва огоньков</p> <p>При переборе всех значений, перебор начинается сначала.</p> <p>После изменения параметров их применение произойдет после завершения нынешнего цикла(полностью завершится нынешний режим движения)</p>
Особенности	Отсутствуют
Питание	Питание от сети переменного тока через стандартный блок питания (прим. USB-адаптер)
Размеры и вес	Достаточно маленькие, чтобы использовать на рабочем столе, маленькой тумбе и т.п
Стоимость производства	Стоимость отладочной платы STM32F103C8 составляет 13 рублей. Стоимость 1 светодиода составляет 1 рубль. Сборка и тестирование 5 рублей. Таким образом стоимость изделия должна составить не более 26 рублей

Схемы и алгоритм работы

- Для настройки работы портов PORTA, PORTB используется регистр RCC_APB2ENR (APB2 peripheral clock enable register).
- Для настройки высокого и низкого уровня на линии ПБВ В используется регистр GPIOB_ODR
- Для настройки внешних прерываний с линий PA0-PA9 используются регистры AFIO_EXTICR1 (External interrupt configuration register 1), AFIO_EXTICR2 (External interrupt configuration register 2), AFIO_EXTICR3 (External interrupt configuration register 3).

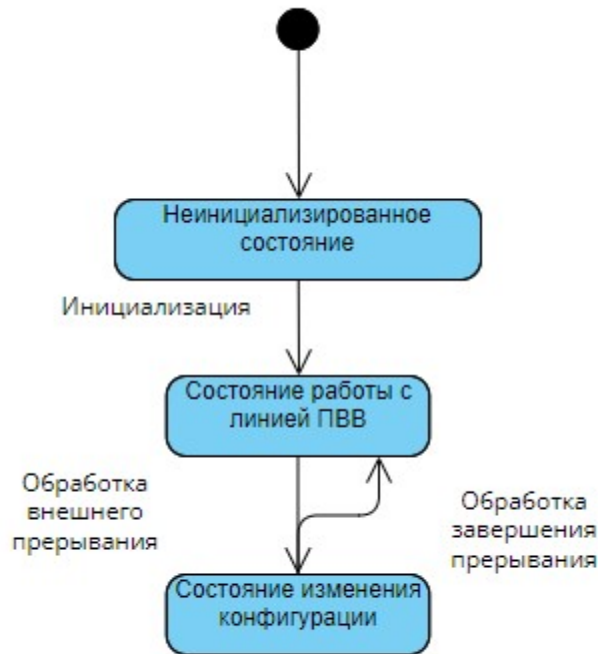


Рис 3. Диаграмма состояний системы

При поступлении внешнего сигнала (ввод с клавиатуры) система переходит в состояние изменений конфигурации, в котором происходит изменение одного из 3-х основных параметров системы: Режим движения, скорости или кол-ва огоньков. После завершения вывода сигнала система автоматически переходит в состояние работы с линией ПБВ.

Рассмотрим работу системы на обобщенной диаграмме последовательности взаимодействия:

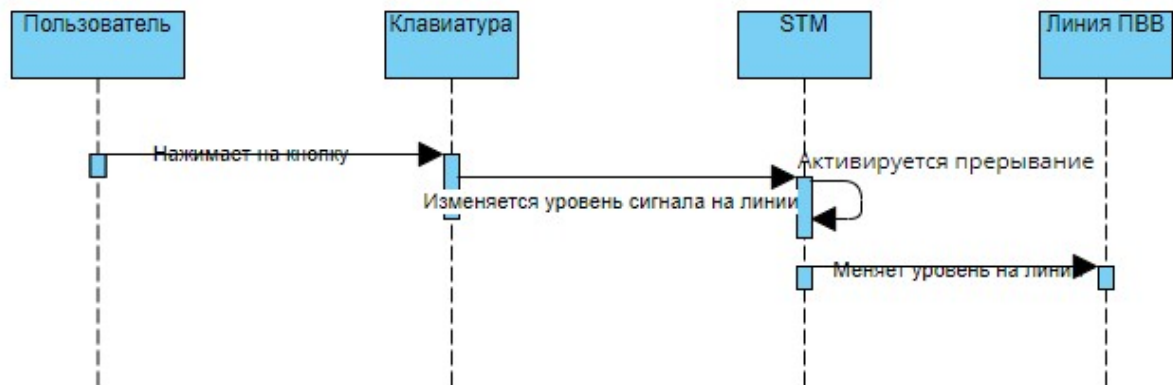


Рис 4. Диаграмма последовательности взаимодействия

Выводы по диаграмме:

1. Используются прерывания для инициализации обработки нажатия кнопок. Каждая кнопка присоединена к соответствующей линии PA0-PA2. Система прерываний микроконтроллера STM32F103C8 позволяет адресовать до 2 прерываний от входных линий, что позволит создать прерывание для каждой кнопки.
2. Необходима задерживающая функция. Некоторая функция, которая будет занимать микроконтроллер на определенное время, для того, чтобы глаз наблюдателя за гирляндой успевал заметить изменения движения огоньков.

Детализируем состояния системы на диаграмме активности функции main:

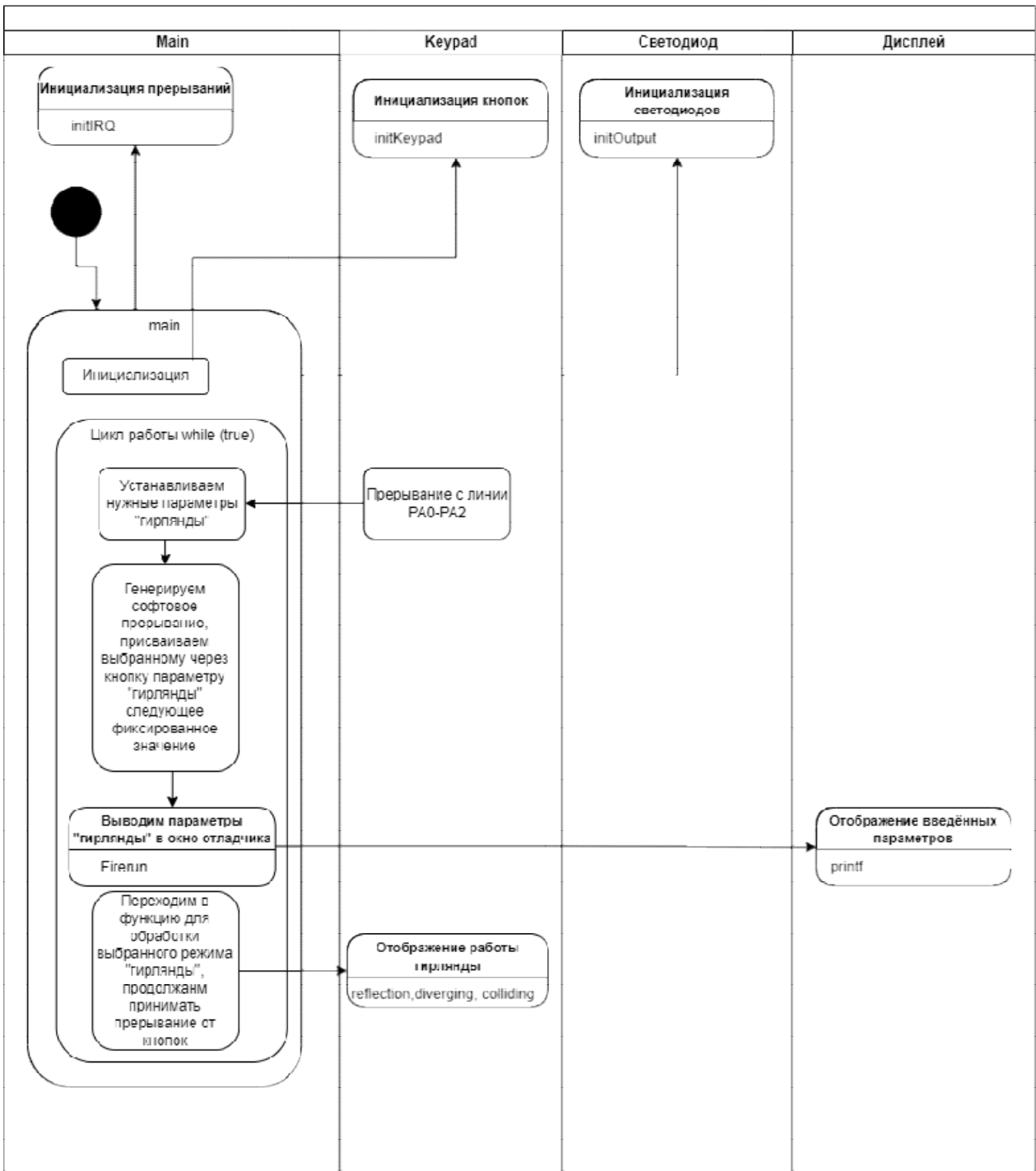


Рисунок 5. – Диаграмма активности функции main.

Отсюда следует выделить следующие основные управляющие структуры:

- 1) Функция `main()` с бесконечным циклом, в котором вызываются функции для режимов.
- 2) Функция `initKeypad()`, реализующая инициализацию кнопок (разрешение работы порта GPIO A, настройка линий PA0-PA2, настройка EXTI).
- 4) Функция `initOutput()`, реализующая инициализацию светодиода (линия PC13).
- 5) Функция `initIRQ()`, реализующая настройку прерываний.

Пример функции `main`:

```
int main()
```

```

{
    initKeypad();
    initOutput();
    initIRQ();
    while(1)
    {
        if (CurrentMode == Reflection)
        {
            reflection();
        }
        else if (CurrentMode == Diverging)
        {
            diverging();
        }
        else if (CurrentMode == Colliding)
        {
            colliding();
        }
    }
}

```

Инициализация кнопок:

```

void initKeypad() {
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN; // разрешаем работу GPIO A
    // обнуляем значения регистров
    GPIOA->CRL = 0;
    GPIOA->CRH = 0;
    //PA0-PA2 Input, Pull up
    SET_BIT(GPIOA->CRL, GPIO_CRL_CNF0_1 | GPIO_CRL_CNF1_1 | GPIO_CRL_CNF2_1 );
    // pull up
    SET_BIT(GPIOA->ODR, GPIO_ODR_ODR0 | GPIO_ODR_ODR1 | GPIO_ODR_ODR2);
    // выбираем в качестве внешних входов EXTI линии:
    // EXTIIn = PAn
    AFIO->EXTICR[0] = AFIO_EXTICR1_EXTI0_PA | AFIO_EXTICR1_EXTI1_PA |
    AFIO_EXTICR1_EXTI2_PA | AFIO_EXTICR1_EXTI3_PA;
}

```

Инициализация светодиода:

```

void initOutput() {
    RCC->APB2ENR |= RCC_APB2ENR_IOPBEN; // разрешаем работу GPIO B
    // PC13, Output mode, max speed 50 MHz, General purpose output push-pull
    GPIOB->CRH &= ~(GPIO_CRH_MODE13 | GPIO_CRH_CNF13);
    SET_BIT(GPIOB->CRH, GPIO_CRH_MODE13);
}

```

Инициализация прерываний:

```

void initIRQ() {
    // прерывание на спад сигнала
    SET_BIT(EXTI->FTSR, EXTI_FTSR_TR0 | EXTI_FTSR_TR1 | EXTI_FTSR_TR2);
    // разрешаем прерывания внешних линий 0-2,
    SET_BIT(EXTI->IMR, EXTI_IMR_MR0 | EXTI_IMR_MR1 | EXTI_IMR_MR2);
    NVIC_EnableIRQ(EXTI0_IRQn);
    NVIC_EnableIRQ(EXTI1_IRQn);
    NVIC_EnableIRQ(EXTI2_IRQn);

    NVIC_SetPriority(EXTI0_IRQn, 1);
    NVIC_SetPriority(EXTI1_IRQn, 1);
    NVIC_SetPriority(EXTI2_IRQn, 1);
}

```

Рассмотрим принцип алгоритма обработки прерывания с линий PA0 – PA2 на примере прерывания PA0 на рисунке 6.

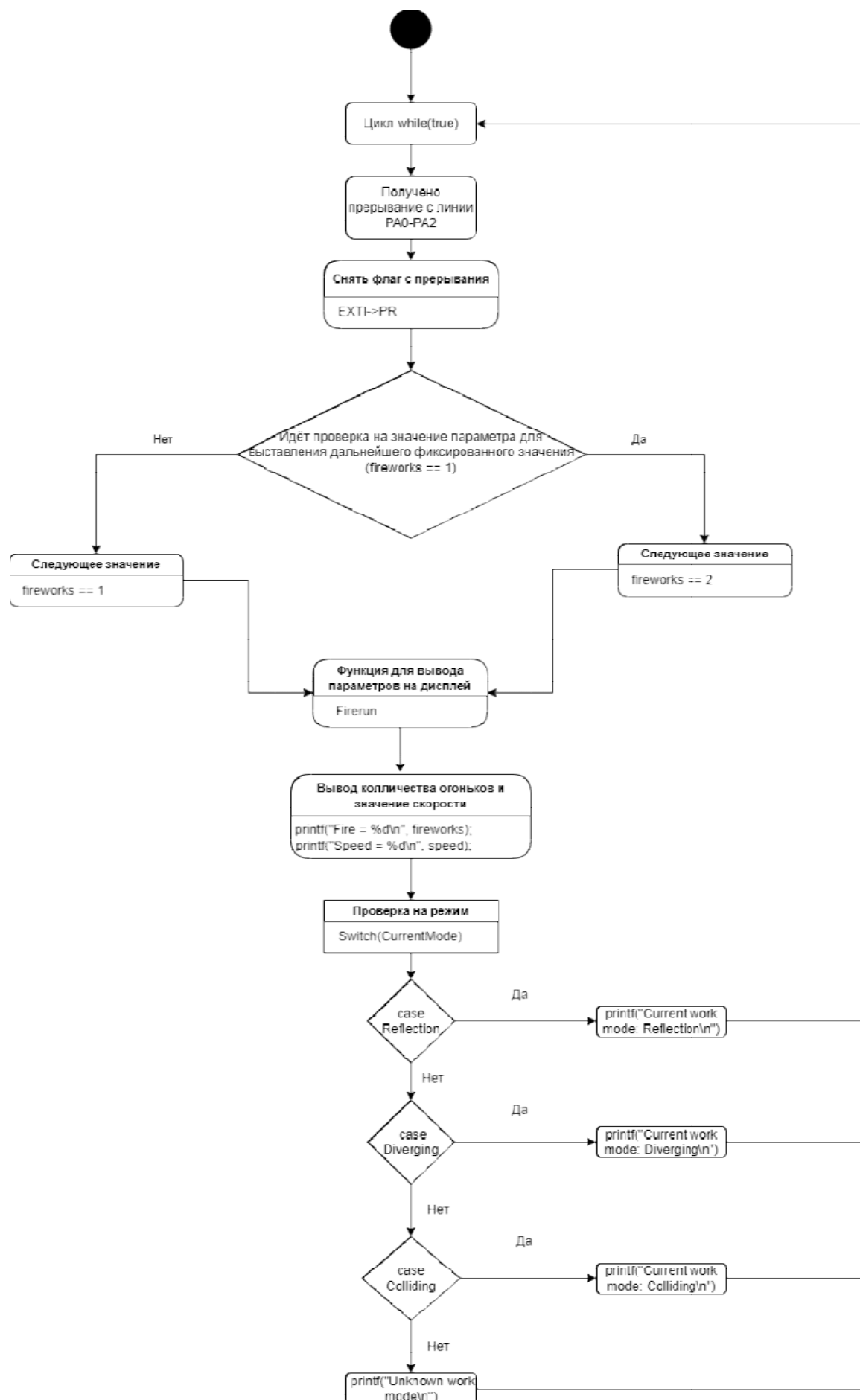


Рисунок 6. – Алгоритм работы функций обработки ввода нового значения параметра.

Пример обработчика прерываний с клавиатуры (изменение параметра скорости):

```
void EXTI1_IRQHandler(void)
{
    EXTI->PR = EXTI_PR_PR1;
    if (speed == 1){
        speed = 3;
    }
    else if (speed == 3){
        speed = 5;
    }
    else {speed = 1;}
    Firerun();
    EXTI->PR = 2;
}
```

Функция Firerun, обеспечивающая вывод параметров “гирлянды”.

```
void Firerun()
{
    printf("Fire = %d\n", fireworks);
    printf("Speed = %d\n", speed);

    switch (CurrentMode)
    {
        case Reflection:
            printf("Current work mode: Reflection\n");
            break;
        case Diverging:
            printf("Current work mode: Diverging\n");
            break;
        case Colliding:
            printf("Current work mode: Colliding\n");
            break;
        default:
            printf("Unknown work mode\n");
            break;
    }
}
```

Функции реализации режимов движения

В связи с поставленным техническим заданием, необходимо реализовать «движение» «огоньков» в 3-х различных режимах движения. Каждый из поставленных режимов движения представлен собственной функцией.

Важным моментом стоит учесть скорость обработки информации современными процессорами и процессором используемым для платы STM103. Из-за высокой скорости обработки информации человеческий глаз не будет успевать воспринимать движение огоньков. Для создания возможности восприятия необходимо определить и выделить некоторые критические состояния зажженных огоньков задерживаясь в этом состоянии на некоторое время. Для фиксирования в определенном состоянии используется функция задержки «Delay». Так же стоит отметить, что именно эта функция и является скоростью движения огоньков. Т.е чем меньше задержка, тем быстрее происходит смена критических состояний.

В связи с тем, что в работе реализован механизм прерываний, в рамках которых параметры выполнения могут измениться, в начале выполнения каждой из функций происходит кеширование параметров. В связи с этим, изменение параметров повлияет на следующий вызов функции, но никак не изменит уже выполняющийся процесс

Reflection – бегущие огоньки.

Данный режим движения это простейшее «перемещение», т.е последовательное включение «огоньков» слева направо и отключение предыдущего огонька, а так же его возвращение в стартовую позицию.

Данная функция завершается сразу после возвращения в стартовую позицию.

В качестве критического состояния выступит один шаг, т.е перемещение огонька на 1 позицию влево или вправо.

Выделение состояния в качестве критического выполняется с помощью размещения функции задержки сразу после достижения состояния, воспринимаемого как критическое. Объясняя простыми словами, огоньки сдвинулись и горят в течении некоторого времени, затем загораются следующие и всё повторяется.

Листинг функции Reflection():

```
float speedLocal = speed;
int8_t fireworksLocal = fireworks;
static volatile int counter = 0;
if (fireworksLocal == 1)
{
    // Right
    GPIOB->ODR|=GPIO_ODR_ODR2;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR2;
    GPIOB->ODR|=GPIO_ODR_ODR3;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR3;
    GPIOB->ODR|=GPIO_ODR_ODR4;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR4;
    GPIOB->ODR|=GPIO_ODR_ODR5;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR5;
    GPIOB->ODR|=GPIO_ODR_ODR6;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR6;
    GPIOB->ODR|=GPIO_ODR_ODR7;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR7;
    GPIOB->ODR|=GPIO_ODR_ODR8;
    Delay(speed);
}
```

```

        GPIOB->ODR&= ~GPIO_ODR_ODR8;
        GPIOB->ODR|=GPIO_ODR_ODR9;
        // Left
        GPIOB->ODR&= ~GPIO_ODR_ODR9;
        GPIOB->ODR|=GPIO_ODR_ODR8;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR8;
        GPIOB->ODR|=GPIO_ODR_ODR7;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR7;
        GPIOB->ODR|=GPIO_ODR_ODR6;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR6;
        GPIOB->ODR|=GPIO_ODR_ODR5;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR5;
        GPIOB->ODR|=GPIO_ODR_ODR4;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR4;
        GPIOB->ODR|=GPIO_ODR_ODR3;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR3;
        GPIOB->ODR|=GPIO_ODR_ODR2;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR2;
    }
    else if (fireworksLocal == 2)
    {
        // Right
        GPIOB->ODR|=GPIO_ODR_ODR2;
        GPIOB->ODR|=GPIO_ODR_ODR3;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR2;
        GPIOB->ODR|=GPIO_ODR_ODR4;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR3;
        GPIOB->ODR|=GPIO_ODR_ODR5;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR4;
        GPIOB->ODR|=GPIO_ODR_ODR6;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR5;
        GPIOB->ODR|=GPIO_ODR_ODR7;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR6;
        GPIOB->ODR|=GPIO_ODR_ODR8;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR7;
        GPIOB->ODR|=GPIO_ODR_ODR9;
        // Left
        GPIOB->ODR&= ~GPIO_ODR_ODR9;
        GPIOB->ODR|=GPIO_ODR_ODR7;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR8;
        GPIOB->ODR|=GPIO_ODR_ODR6;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR7;
        GPIOB->ODR|=GPIO_ODR_ODR5;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR6;
        GPIOB->ODR|=GPIO_ODR_ODR4;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR5;
        GPIOB->ODR|=GPIO_ODR_ODR3;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR4;
        GPIOB->ODR|=GPIO_ODR_ODR2;
    }
}

```

Diverging – разбегающиеся огоньки

Данный режим движения это «перемещение» огоньков друг от друга, т.е последовательное включение «огоньков» из центра влево и вправо и отключение предыдущего огонька.

Достигая краёв огоньки гаснут. После погашения огоньков, они снова появляются в центре
 Данная функция завершается сразу после погашения всех огоньков.

В качестве критического состояния выступит один шаг, т.е перемещение огонька на 1 позицию влево или вправо.

Выделение состояния в качестве критического выполняется с помощью размещения функции задержки сразу после достижения состояния, воспринимаемого как критическое. Объясняя простыми словами, огоньки сдвинулись и горят в течении некоторого времени, затем загораются следующие и всё повторяется.

Листинг функции Diverging():

```
float speedLocal = speed;
int8_t fireworksLocal = fireworks;
if (fireworksLocal == 1)
{
    GPIOB->ODR|=GPIO_ODR_ODR5;
    GPIOB->ODR|=GPIO_ODR_ODR6;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR6;
    GPIOB->ODR&= ~GPIO_ODR_ODR5;
    GPIOB->ODR|=GPIO_ODR_ODR4;
    GPIOB->ODR|=GPIO_ODR_ODR7;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR4;
    GPIOB->ODR&= ~GPIO_ODR_ODR7;
    GPIOB->ODR|=GPIO_ODR_ODR3;
    GPIOB->ODR|=GPIO_ODR_ODR8;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR3;
    GPIOB->ODR&= ~GPIO_ODR_ODR8;
    GPIOB->ODR|=GPIO_ODR_ODR2;
    GPIOB->ODR|=GPIO_ODR_ODR9;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR2;
    GPIOB->ODR&= ~GPIO_ODR_ODR9;
}
else if (fireworksLocal == 2)
{
    GPIOB->ODR|=GPIO_ODR_ODR5;
    GPIOB->ODR|=GPIO_ODR_ODR6;
    GPIOB->ODR|=GPIO_ODR_ODR4;
    GPIOB->ODR|=GPIO_ODR_ODR7;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR6;
    GPIOB->ODR&= ~GPIO_ODR_ODR5;
    GPIOB->ODR|=GPIO_ODR_ODR3;
    GPIOB->ODR|=GPIO_ODR_ODR8;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR4;
    GPIOB->ODR&= ~GPIO_ODR_ODR7;
    GPIOB->ODR|=GPIO_ODR_ODR2;
    GPIOB->ODR|=GPIO_ODR_ODR9;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR3;
    GPIOB->ODR&= ~GPIO_ODR_ODR8;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR2;
    GPIOB->ODR&= ~GPIO_ODR_ODR9;
    Delay(speed);
}
```

Colliding – сталкивающиеся огоньки

Данный режим движения это «перемещение» огоньков друг к другу, т.е последовательное включение «огоньков» от краев к центру и отключение предыдущего огонька. Достигая центра огоньки «сталкиваются», что характеризуется зажиганием всех огоньков на ленте, а затем их погашение. После погашения огоньков, они снова по краям.

Данная функция завершается сразу после погашения всех огоньков.

В качестве критического состояния выступит один шаг, т.е перемещение огонька на 1 позицию влево или вправо.

Выделение состояния в качестве критического выполняется с помощью размещения функции задержки сразу после достижения состояния, воспринимаемого как критическое. Объясняя простыми словами, огоньки сдвинулись и горят в течении некоторого времени, затем загораются следующие и всё повторяется.

При нынешней конфигурации после «столкновения» огоньки загораются сразу и все, а затем сразу же все гаснут. Удаление **комментариев** в блоке после столкновения, позволяет сделать этот процесс более плавным, т.е после «столкновения» огоньки плавно загорятся от центра к краям и так же плавно погаснут.

Листинг функции Colliding():




```
float speedLocal = speed;
int8_t fireworksLocal = fireworks;
if (fireworksLocal == 1)
{
    GPIOB->ODR|=GPIO_ODR_ODR2;
    GPIOB->ODR|=GPIO_ODR_ODR9;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR2;
    GPIOB->ODR&= ~GPIO_ODR_ODR9;
    GPIOB->ODR|=GPIO_ODR_ODR3;
    GPIOB->ODR|=GPIO_ODR_ODR8;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR3;
    GPIOB->ODR&= ~GPIO_ODR_ODR8;
    GPIOB->ODR|=GPIO_ODR_ODR4;
    GPIOB->ODR|=GPIO_ODR_ODR7;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR4;
    GPIOB->ODR&= ~GPIO_ODR_ODR7;
    GPIOB->ODR|=GPIO_ODR_ODR5;
    GPIOB->ODR|=GPIO_ODR_ODR6;
    Delay(speed);
    GPIOB->ODR|=GPIO_ODR_ODR4;
    GPIOB->ODR|=GPIO_ODR_ODR7;
    // Delay(speed);
    GPIOB->ODR|=GPIO_ODR_ODR3;
    GPIOB->ODR|=GPIO_ODR_ODR8;
    // Delay(speed);
    GPIOB->ODR|=GPIO_ODR_ODR2;
    GPIOB->ODR|=GPIO_ODR_ODR9;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR5;
    GPIOB->ODR&= ~GPIO_ODR_ODR6;
    // Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR4;
    GPIOB->ODR&= ~GPIO_ODR_ODR7;
    // Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR3;
    GPIOB->ODR&= ~GPIO_ODR_ODR8;
    // Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR2;
    GPIOB->ODR&= ~GPIO_ODR_ODR9;
}
else if (fireworksLocal == 2)
{
    GPIOB->ODR|=GPIO_ODR_ODR2;
    GPIOB->ODR|=GPIO_ODR_ODR9;
    GPIOB->ODR|=GPIO_ODR_ODR3;
    GPIOB->ODR|=GPIO_ODR_ODR8;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR2;
    GPIOB->ODR&= ~GPIO_ODR_ODR9;
    GPIOB->ODR|=GPIO_ODR_ODR4;
    GPIOB->ODR|=GPIO_ODR_ODR7;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR3;
    GPIOB->ODR&= ~GPIO_ODR_ODR8;
    GPIOB->ODR|=GPIO_ODR_ODR5;
    GPIOB->ODR|=GPIO_ODR_ODR6;
    Delay(speed);
    GPIOB->ODR|=GPIO_ODR_ODR3;
    GPIOB->ODR|=GPIO_ODR_ODR8;
    // Delay(speed);
```



```
GPIOB->ODR|=GPIO_ODR_ODR2;
GPIOB->ODR|=GPIO_ODR_ODR9;
Delay(speed);
GPIOB->ODR&= ~GPIO_ODR_ODR5;
GPIOB->ODR&= ~GPIO_ODR_ODR6;
// Delay(speed);
GPIOB->ODR&= ~GPIO_ODR_ODR4;
GPIOB->ODR&= ~GPIO_ODR_ODR7;
// Delay(speed);
GPIOB->ODR&= ~GPIO_ODR_ODR3;
GPIOB->ODR&= ~GPIO_ODR_ODR8;
// Delay(speed);
GPIOB->ODR&= ~GPIO_ODR_ODR2;
GPIOB->ODR&= ~GPIO_ODR_ODR9;
Delay(speed);
}
```

Тестирование

Тесты для контроля соответствия прибора техническому заданию

№	Описание	Ожидаемый результат	Фактический результат
1	При нажатии на кнопку 0, при изначальных параметрах (Fire = 1; Speed = 1; Current work mode: Reflection), изменится параметр Fire	В Debug выводятся следующие параметры: Fire = 2; Speed = 1; Current work mode: Reflection, в регистрах начала “гирлянды” загорятся два последовательно стоящих друг за другом огоньк	<pre>Fire = 2 Speed = 1 Current work mode: Reflection</pre> 
2	При нажатии на кнопку 0 изменится параметр Fire	В Debug выводятся следующие параметры: Fire = 1; Speed = 1; Current work mode: Reflection, а в регистре начала “гирлянды” загорится один огонёк	<pre>Fire = 1 Speed = 1 Current work mode: Reflection</pre> 
3	При нажатии на кнопку 0 изменится параметр Fire, при нажатии на кнопку 1 изменится параметр speed, при нажатии на кнопку 2 изменится режим	В Debug выводятся следующие параметры: Fire = 2; Speed = 3; Current work mode: Diverging. Увеличится время задержки. В регистрах загорятся две группы огоньков,	<pre>Fire = 2 Speed = 3 Current work mode: Diverging</pre> 

4	При нажатии на кнопку 0 изменится параметр Fire	стоящих друг напротив друга в центре “гирлянды” В Debug выводятся следующие параметры: Fire = 1; Speed = 3; Current work mode: Diverging. Увеличится время задержки. В регистрах загорятся два огонька, стоящих друг напротив друга в центре в “центре гирлянды”	<pre>Fire = 1 Speed = 3 Current work mode: Diverging</pre> <table><tr><td>ODR2</td><td><input type="checkbox"/></td></tr><tr><td>ODR3</td><td><input type="checkbox"/></td></tr><tr><td>ODR4</td><td><input type="checkbox"/></td></tr><tr><td>ODR5</td><td><input checked="" type="checkbox"/></td></tr><tr><td>ODR6</td><td><input checked="" type="checkbox"/></td></tr><tr><td>ODR7</td><td><input type="checkbox"/></td></tr><tr><td>ODR8</td><td><input type="checkbox"/></td></tr><tr><td>ODR9</td><td><input type="checkbox"/></td></tr></table>	ODR2	<input type="checkbox"/>	ODR3	<input type="checkbox"/>	ODR4	<input type="checkbox"/>	ODR5	<input checked="" type="checkbox"/>	ODR6	<input checked="" type="checkbox"/>	ODR7	<input type="checkbox"/>	ODR8	<input type="checkbox"/>	ODR9	<input type="checkbox"/>
ODR2	<input type="checkbox"/>																		
ODR3	<input type="checkbox"/>																		
ODR4	<input type="checkbox"/>																		
ODR5	<input checked="" type="checkbox"/>																		
ODR6	<input checked="" type="checkbox"/>																		
ODR7	<input type="checkbox"/>																		
ODR8	<input type="checkbox"/>																		
ODR9	<input type="checkbox"/>																		
5	При нажатии на кнопку 0 изменится параметр Fire, при нажатии на кнопку 2 изменится режим	В Debug выводятся следующие параметры: Fire = 2; Speed = 3; Current work mode: Colliding. В регистрах загорятся две группы огоньков, стоящих друг напротив друга на разных концах “гирлянды”	<pre>Fire = 2 Speed = 3 Current work mode: Colliding</pre> <table><tr><td>ODR2</td><td><input checked="" type="checkbox"/></td></tr><tr><td>ODR3</td><td><input checked="" type="checkbox"/></td></tr><tr><td>ODR4</td><td><input type="checkbox"/></td></tr><tr><td>ODR5</td><td><input type="checkbox"/></td></tr><tr><td>ODR6</td><td><input type="checkbox"/></td></tr><tr><td>ODR7</td><td><input type="checkbox"/></td></tr><tr><td>ODR8</td><td><input checked="" type="checkbox"/></td></tr><tr><td>ODR9</td><td><input checked="" type="checkbox"/></td></tr></table>	ODR2	<input checked="" type="checkbox"/>	ODR3	<input checked="" type="checkbox"/>	ODR4	<input type="checkbox"/>	ODR5	<input type="checkbox"/>	ODR6	<input type="checkbox"/>	ODR7	<input type="checkbox"/>	ODR8	<input checked="" type="checkbox"/>	ODR9	<input checked="" type="checkbox"/>
ODR2	<input checked="" type="checkbox"/>																		
ODR3	<input checked="" type="checkbox"/>																		
ODR4	<input type="checkbox"/>																		
ODR5	<input type="checkbox"/>																		
ODR6	<input type="checkbox"/>																		
ODR7	<input type="checkbox"/>																		
ODR8	<input checked="" type="checkbox"/>																		
ODR9	<input checked="" type="checkbox"/>																		
6	При нажатии на кнопку 0 изменится параметр Fire, при нажатии на кнопку 1 изменится параметр speed	В Debug выводятся следующие параметры: Fire = 1; Speed = 5; Current work mode: Colliding. В регистрах	<pre>Fire = 1 Speed = 5 Current work mode: Colliding</pre>																

загорятся два
огонька,
стоящих друг
напротив
друга на
разных
концах
“гирлянды”

ODR2
ODR3
ODR4
ODR5
ODR6
ODR7
ODR8
ODR9

<input checked="" type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input checked="" type="checkbox"/>

Заключение

В результате выполнения данного проекта было осуществлено проектирование устройства на базе микроконтроллера. Были получены навыки в области архитектуры и программного обеспечения встроенных систем, а так же знания о структуре, функциях и основах программирования микроконтроллеров, позволяющих решать вопросы анализа функционирования программного обеспечения встраиваемых систем

Список использованной литературы

1. Микроконтроллеры. Разработка встраиваемых приложений: учебное пособие / А.Е. Васильев; С.-Петербургский государственный политехнический ун-т. - СПб. : Изд-во СПбГПУ, 2003. - 211 с.
2. The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors. Third Edition. Joseph Yiu. ARM Ltd., Cambridge, UK. [электронный ресурс] // URL: <https://www.pdfdrive.com/the-definitive-guide-to-arm-cortex-m3-and-cortex-m4-processors-e187111520.html> (дата обращения 12.05.2020).
3. Джозеф Ю. Ядро Cortex-M3 компании ARM. Полное руководство. 2012. ISBN: 978- 5-94120-243-0. [электронный ресурс] // URL: <https://b-ok.xyz/book/2373589/b5c3ad> (дата обращения 12.05.2020).
4. RM0008. Reference manual STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs [электронный ресурс] // URL: https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf (дата обращения 12.05.2020).
5. Datasheet STM32F103x8 STM32F103xB Medium-density performance line ARM®- based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. Interfaces. [электронный ресурс] // URL: <https://www.st.com/resource/en/datasheet/stm32f103c8.pdf> (дата обращения 12.05.2020).
6. Мартин М. Инсайдерское руководство по STM32 [электронный ресурс] // URL: <https://istarik.ru/file/STM32.pdf> (дата обращения 12.05.2020).

Приложение 1. Код программы

```
#include "RTE_Components.h"
#include CMSIS_device_header
#include <stdio.h>
#include <stdlib.h>

// Огоньки
static volatile int8_t fireworks =1;

// режимы гирлянды
typedef enum
{
    Reflection,
    Diverging,
    Colliding
} WorkMode;

WorkMode CurrentMode =Reflection;

// скорость
volatile int speed=3;

void initKeypad() {
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN; // разрешаем работу GPIO A
    // обнуляем значения регистров
    GPIOA->CRL = 0;
    GPIOA->CRH = 0;
    //PA0-PA9 Input, Pull up
    SET_BIT(GPIOA->CRL, GPIO_CRL_CNF0_1 | GPIO_CRL_CNF1_1 | GPIO_CRL_CNF2_1 );
    // pull up
    SET_BIT(GPIOA->ODR, GPIO_ODR_ODR0 | GPIO_ODR_ODR1 | GPIO_ODR_ODR2);
    // выбираем в качестве внешних входов EXTI линии:
    // EXTIn = PAn
    AFIO->EXTICR[0] = AFIO_EXTICR1_EXTI0_PA | AFIO_EXTICR1_EXTI1_PA |
    AFIO_EXTICR1_EXTI2_PA | AFIO_EXTICR1_EXTI3_PA;
}

void initOutput() {
    RCC->APB2ENR |= RCC_APB2ENR_IOPBEN; // разрешаем работу GPIO B
    // PC13, Output mode, max speed 50 MHz, General purpose output push-pull
    GPIOB->CRH &= ~(GPIO_CRH_MODE13 | GPIO_CRH_CNF13);
    SET_BIT(GPIOB->CRH, GPIO_CRH_MODE13);
}

void initIRQ() {
    // прерывание на спад сигнала
    SET_BIT(EXTI->FTSR, EXTI_FTSR_TR0 | EXTI_FTSR_TR1 | EXTI_FTSR_TR2);
    // разрешаем прерывания внешних линий 0-2
    SET_BIT(EXTI->IMR, EXTI_IMR_MR0 | EXTI_IMR_MR1 | EXTI_IMR_MR2);
    NVIC_EnableIRQ(EXTI0_IRQn);
    NVIC_EnableIRQ(EXTI1_IRQn);
    NVIC_EnableIRQ(EXTI2_IRQn);
    NVIC_EnableIRQ(EXTI15_10_IRQn);
    NVIC_SetPriority(EXTI0_IRQn, 1);
    NVIC_SetPriority(EXTI1_IRQn, 1);
    NVIC_SetPriority(EXTI2_IRQn, 1);
    NVIC_SetPriority(EXTI15_10_IRQn, 3);
}

void Delay(volatile int speeds)
{
    volatile int counters = 0;
    volatile int speedcounter = 0;
    while(counters != 10)
    {
        while(speedcounter != speeds)

            {speedcounter++;}
        speedcounter = 0;
        counters++;
    }
    counters = 0;
}
```

```

}

void reflection()
{
    float speedLocal = speed;
    int8_t fireworksLocal = fireworks;
    if (fireworksLocal == 1)
    {
        // Right
        GPIOB->ODR|=GPIO_ODR_ODR2;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR2;
        GPIOB->ODR|=GPIO_ODR_ODR3;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR3;
        GPIOB->ODR|=GPIO_ODR_ODR4;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR4;
        GPIOB->ODR|=GPIO_ODR_ODR5;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR5;
        GPIOB->ODR|=GPIO_ODR_ODR6;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR6;
        GPIOB->ODR|=GPIO_ODR_ODR7;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR7;
        GPIOB->ODR|=GPIO_ODR_ODR8;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR8;
        GPIOB->ODR|=GPIO_ODR_ODR9;
        // Left
        GPIOB->ODR&= ~GPIO_ODR_ODR9;
        GPIOB->ODR|=GPIO_ODR_ODR8;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR8;
        GPIOB->ODR|=GPIO_ODR_ODR7;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR7;
        GPIOB->ODR|=GPIO_ODR_ODR6;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR6;
        GPIOB->ODR|=GPIO_ODR_ODR5;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR5;
        GPIOB->ODR|=GPIO_ODR_ODR4;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR4;
        GPIOB->ODR|=GPIO_ODR_ODR3;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR3;
        GPIOB->ODR|=GPIO_ODR_ODR2;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR2;
    }
    else if (fireworksLocal == 2)
    {
        // Right
        GPIOB->ODR|=GPIO_ODR_ODR2;
        GPIOB->ODR|=GPIO_ODR_ODR3;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR2;
        GPIOB->ODR|=GPIO_ODR_ODR4;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR3;
        GPIOB->ODR|=GPIO_ODR_ODR5;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR4;
        GPIOB->ODR|=GPIO_ODR_ODR6;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR5;
        GPIOB->ODR|=GPIO_ODR_ODR7;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR6;
        GPIOB->ODR|=GPIO_ODR_ODR8;
        Delay(speed);
    }
}

```

```

        GPIOB->ODR&= ~GPIO_ODR_ODR7;
        GPIOB->ODR|=GPIO_ODR_ODR9;
        // Left
        GPIOB->ODR&= ~GPIO_ODR_ODR9;
        GPIOB->ODR|=GPIO_ODR_ODR7;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR8;
        GPIOB->ODR|=GPIO_ODR_ODR6;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR7;
        GPIOB->ODR|=GPIO_ODR_ODR5;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR6;
        GPIOB->ODR|=GPIO_ODR_ODR4;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR5;
        GPIOB->ODR|=GPIO_ODR_ODR3;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR4;
        GPIOB->ODR|=GPIO_ODR_ODR2;
    }
}

void diverging()
{
    float speedLocal = speed;
    int8_t fireworksLocal = fireworks;
    if (fireworksLocal == 1)
    {
        GPIOB->ODR|=GPIO_ODR_ODR5;
        GPIOB->ODR|=GPIO_ODR_ODR6;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR6;
        GPIOB->ODR&= ~GPIO_ODR_ODR5;
        GPIOB->ODR|=GPIO_ODR_ODR4;
        GPIOB->ODR|=GPIO_ODR_ODR7;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR4;
        GPIOB->ODR&= ~GPIO_ODR_ODR7;
        GPIOB->ODR|=GPIO_ODR_ODR3;
        GPIOB->ODR|=GPIO_ODR_ODR8;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR3;
        GPIOB->ODR&= ~GPIO_ODR_ODR8;
        GPIOB->ODR|=GPIO_ODR_ODR2;
        GPIOB->ODR|=GPIO_ODR_ODR9;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR2;
        GPIOB->ODR&= ~GPIO_ODR_ODR9;
    }
    else if (fireworksLocal == 2)
    {
        GPIOB->ODR|=GPIO_ODR_ODR5;
        GPIOB->ODR|=GPIO_ODR_ODR6;
        GPIOB->ODR|=GPIO_ODR_ODR4;
        GPIOB->ODR|=GPIO_ODR_ODR7;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR6;
        GPIOB->ODR&= ~GPIO_ODR_ODR5;
        GPIOB->ODR|=GPIO_ODR_ODR3;
        GPIOB->ODR|=GPIO_ODR_ODR8;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR4;
        GPIOB->ODR&= ~GPIO_ODR_ODR7;
        GPIOB->ODR|=GPIO_ODR_ODR2;
        GPIOB->ODR|=GPIO_ODR_ODR9;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR3;
        GPIOB->ODR&= ~GPIO_ODR_ODR8;
        Delay(speed);
        GPIOB->ODR&= ~GPIO_ODR_ODR2;
        GPIOB->ODR&= ~GPIO_ODR_ODR9;
        Delay(speed);
    }
}

void colliding()
{

```



```

float speedLocal = speed;
int8_t fireworksLocal = fireworks;
if (fireworksLocal == 1)
{
    GPIOB->ODR|=GPIO_ODR_ODR2;
    GPIOB->ODR|=GPIO_ODR_ODR9;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR2;
    GPIOB->ODR&= ~GPIO_ODR_ODR9;
    GPIOB->ODR|=GPIO_ODR_ODR3;
    GPIOB->ODR|=GPIO_ODR_ODR8;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR3;
    GPIOB->ODR&= ~GPIO_ODR_ODR8;
    GPIOB->ODR|=GPIO_ODR_ODR4;
    GPIOB->ODR|=GPIO_ODR_ODR7;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR4;
    GPIOB->ODR&= ~GPIO_ODR_ODR7;
    GPIOB->ODR|=GPIO_ODR_ODR5;
    GPIOB->ODR|=GPIO_ODR_ODR6;
    Delay(speed);
    GPIOB->ODR|=GPIO_ODR_ODR4;
    GPIOB->ODR|=GPIO_ODR_ODR7;
    GPIOB->ODR|=GPIO_ODR_ODR3;
    GPIOB->ODR|=GPIO_ODR_ODR8;
    GPIOB->ODR|=GPIO_ODR_ODR2;
    GPIOB->ODR|=GPIO_ODR_ODR9;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR5;
    GPIOB->ODR&= ~GPIO_ODR_ODR6;
    GPIOB->ODR&= ~GPIO_ODR_ODR4;
    GPIOB->ODR&= ~GPIO_ODR_ODR7;
    GPIOB->ODR&= ~GPIO_ODR_ODR3;
    GPIOB->ODR&= ~GPIO_ODR_ODR8;
    GPIOB->ODR&= ~GPIO_ODR_ODR2;
    GPIOB->ODR&= ~GPIO_ODR_ODR9;
}
else if (fireworksLocal == 2)
{
    GPIOB->ODR|=GPIO_ODR_ODR2;
    GPIOB->ODR|=GPIO_ODR_ODR9;
    GPIOB->ODR|=GPIO_ODR_ODR3;
    GPIOB->ODR|=GPIO_ODR_ODR8;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR2;
    GPIOB->ODR&= ~GPIO_ODR_ODR9;
    GPIOB->ODR|=GPIO_ODR_ODR4;
    GPIOB->ODR|=GPIO_ODR_ODR7;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR3;
    GPIOB->ODR&= ~GPIO_ODR_ODR8;
    GPIOB->ODR|=GPIO_ODR_ODR5;
    GPIOB->ODR|=GPIO_ODR_ODR6;
    Delay(speed);
    GPIOB->ODR|=GPIO_ODR_ODR3;
    GPIOB->ODR|=GPIO_ODR_ODR8;
    GPIOB->ODR|=GPIO_ODR_ODR2;
    GPIOB->ODR|=GPIO_ODR_ODR9;
    Delay(speed);
    GPIOB->ODR&= ~GPIO_ODR_ODR5;
    GPIOB->ODR&= ~GPIO_ODR_ODR6;
    GPIOB->ODR&= ~GPIO_ODR_ODR4;
    GPIOB->ODR&= ~GPIO_ODR_ODR7;
    GPIOB->ODR&= ~GPIO_ODR_ODR3;
    GPIOB->ODR&= ~GPIO_ODR_ODR8;
    GPIOB->ODR&= ~GPIO_ODR_ODR2;
    GPIOB->ODR&= ~GPIO_ODR_ODR9;
    Delay(speed);
}
};

int main()
{
    initKeypad();
    initOutput();
    initIRQ();

```

```

        while(1)
        {
            if (CurrentMode == Reflection)
            {
                reflection();
            }
            else if (CurrentMode == Diverging)
            {
                diverging();
            }
            else if (CurrentMode == Colliding)
            {
                colliding();
            }
        }
    }

void Firerun()
{
    printf("Fire = %d\n", fireworks);
    printf("Speed = %d\n", speed);

    switch (CurrentMode)
    {
        case Reflection:
            printf("Current work mode: Reflection\n");
            break;
        case Diverging:
            printf("Current work mode: Diverging\n");
            break;
        case Colliding:
            printf("Current work mode: Colliding\n");
            break;
        default:
            printf("Unknown work mode\n");
            break;
    }
}

void EXTI0_IRQHandler(void)
{
    EXTI->PR = EXTI_PR_PR0;

    if (fireworks == 1){
        fireworks = 2;
    }
    else {fireworks = 1;
    }

    Firerun();
    // снимаем флаг прерывания
    EXTI->PR = 1;
}

void EXTI1_IRQHandler(void)
{
    EXTI->PR = EXTI_PR_PR1;
    if (speed == 1){
        speed = 3;
    }
    else if (speed == 3){
        speed = 5;
    }
    else {speed = 1;}
    Firerun();
    EXTI->PR = 2;
}

void EXTI2_IRQHandler(void)
{
    EXTI->PR = EXTI_PR_PR2;
    if (CurrentMode == Reflection){
        CurrentMode =Diverging;
    }
    else if (CurrentMode ==Diverging){
        CurrentMode =Colliding;
    }
}

```

```

    }
    else {CurrentMode = Reflection;}
    Firerun();
    EXTI->PR = 3;
}

```

Код отладчика Keil для имитации кнопок (скрипт ini файла):

```

OSC = 8000000; //Установили частоту внешнего генератор Xtal (Hz)
PORTA &= ~0x3FF;
SIGNAL void toggle_A0_pin() {
    PORTA |= 0x01;
    swatch (0.01);
    PORTA ^= 0x01;}
SIGNAL void toggle_A1_pin() {
    PORTA |= 0x02;
    swatch (0.01);
    PORTA ^= 0x02;}
SIGNAL void toggle_A2_pin() {
    PORTA |= 0x04;
    swatch (0.01);
    PORTA ^= 0x04;}
DEFINE BUTTON "Key0 PA0", "toggle_A0_pin()"
DEFINE BUTTON "Key1 PA1", "toggle_A1_pin()"
DEFINE BUTTON "Key2 PA2", "toggle_A2_pin()"

```