ОТЧЕТ ПО КУРСОВОМУ ПРОЕКТУ

Реализация управления цветом RGB-светодиода с использованием инкрементного датчика и ШИМ

1. Поставленная задача

Разработать систему управления цветом RGB-светодиода с использованием:

- инкрементного датчика (энкодера) для выбора цвета и интенсивности;
- широтно-импульсной модуляции (ШИМ) для управления яркостью каналов;
- прерываний для обработки сигналов энкодера и генерации ШИМ.

Важным уточнением является программирование микроконтроллера AVR используя язык программирования Ассемблер.

Входные данные:

- сигналы с инкрементного датчика (фаза A, фаза B);
- тактовая частота микроконтроллера 16 МГц.

Выходные данные:

- ШИМ сигналы для трёх каналов RGB-светодиода;
- динамическое изменение цвета в реальном времени.

2. Алгоритмы решения поставленной задачи

Основные алгоритмы:

- обработка прерываний от энкодера с определением направления вращения;
- преобразование положения энкодера в значения ШИМ для каналов R, G, B;
- функция имитации работы;
- функция ожидания (простоя).

Блок-схемы алгоритмов будет представлены ниже. Каждый из алгоритмов описан без адаптации к реализации на языке Ассемблер, то есть алгоритм универсален и описывает общее решение поставленной задачи.

В обрабоке сигналов с энкодера использовался код Грея, позволяет с минимальным количеством действий определить направление движения энкодера по сигналам с его каналов. На рисунке 1 представлена блок-схема алгоритма отбработки значений с энкодера.

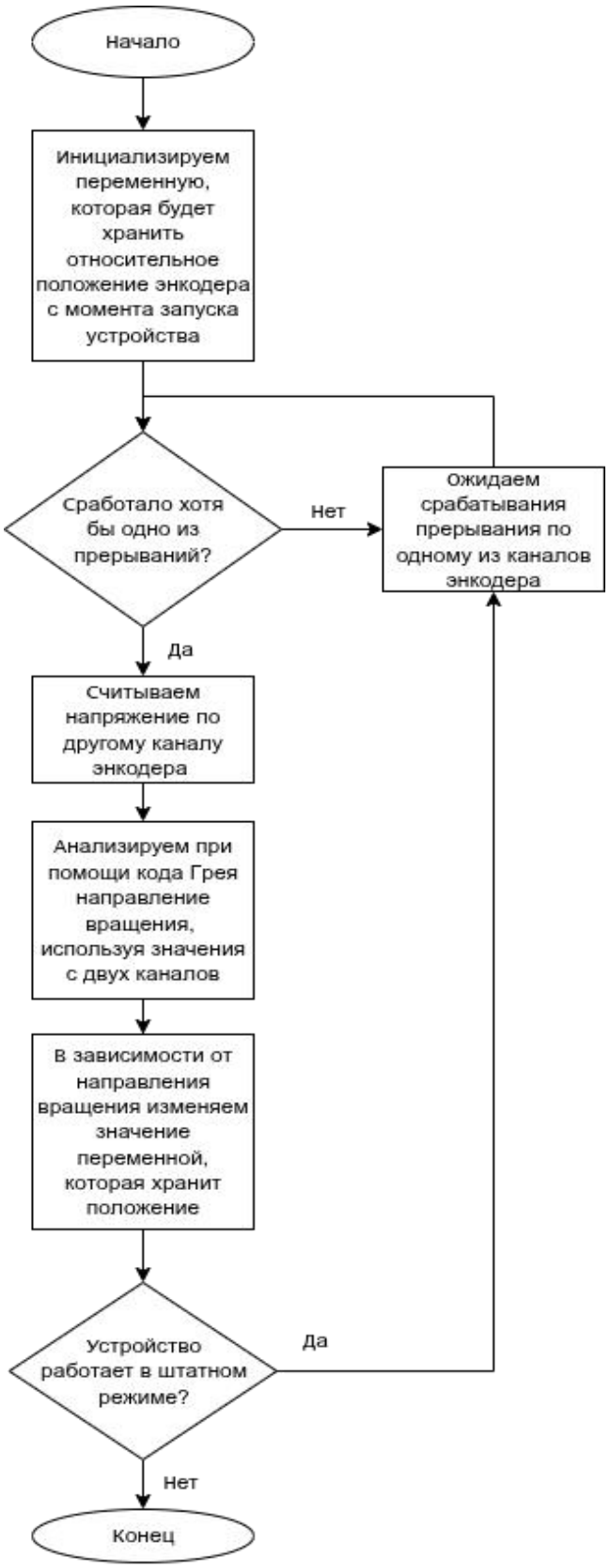

Рисунок 1 - блок-схема алгоритма отбработки значений с энкодера.

На рисунке 2 представленна диаграмма состояний, которая описывает обработку сигналов с энкодера.

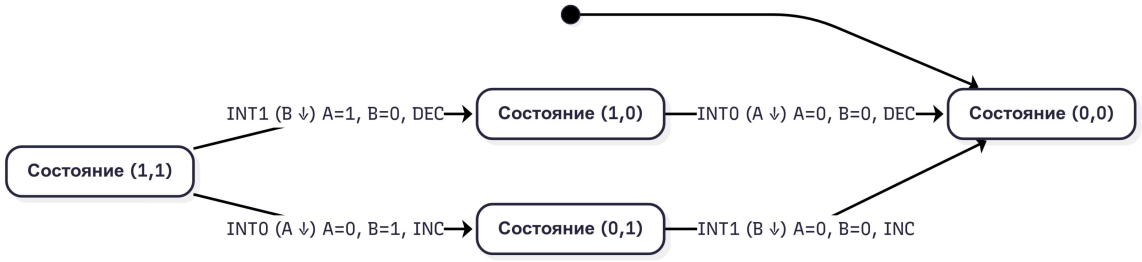

Рисунок 2 - диаграмма состояний обработки сигналов с энкодера.

В качествве проверки можно рассмотреть следущий пример вращения по часовой:

```
(0,0) -> (1,0) [Не обрабатывается] ->
(1,0) -> (1,1) [Не обрабатывается] ->
(1,1) --INT0--> (0,1) [INC] ->
(0,1) --INT1--> (0,0) [INC]
```

На рисунке 3 представлена блок-схема алгоритма отбработки сигналов с энкодера. В блоке «Вычисление...» подразумевается какие-то математические функции, которые на вход получают текущее положение энкодера и выдают на выходе значения для одного из каналов. Математическая функция может быть любой для каждого из каналов. В моей реализации использовались следущие формулы

$$R = C\%256,$$

$$B = C\%256,$$

$$G = 255 - C\%256,$$

где R, G, B - значения коэффицинета заполнения (от 0 до 255) ШИМ для соответствующего канала, C - значение счётчика, который оценивает положение энкодера.

Рисунок 3 - блок-схема алгоритма отбработки изменения состояния светодиода.

Функция иммитации работы является индикатором того, что устройство функционирует в штатном режиме и предоставляет некоторую вариативность для последующего использования данного проекта. Решение, реализованное через прерывания, позволяет не тратить ресурсы микроконтроллера на функции постоянного опроса датчиков, а реагировать по факту совершения, что является хорошей оптимизацией занимаемого процессорного времени, которое тратися основной программой в 1 итерации. В моей реализации основной программой является изменение состояния встроенного в плату контроллера светодиода, ожидание (простой), именение состояния вновь, ожидание (простой). Более наглядно алгоритм основной работы микроконтроллера представлен на рисунке 4.

Рисунок 4 - блок-схема функции имитации работы.

Функция ожидания позволяет микроконтроллеру «подождать» некоторое время. То есть зафиксировать своё состояние и не изменять его некоторое время. В самой простой реализации это счётчик, который считает до определённого числа, а потом продолжает выполнять основную программу. Блок-схема функции ожидания представлена на рисунке 5.

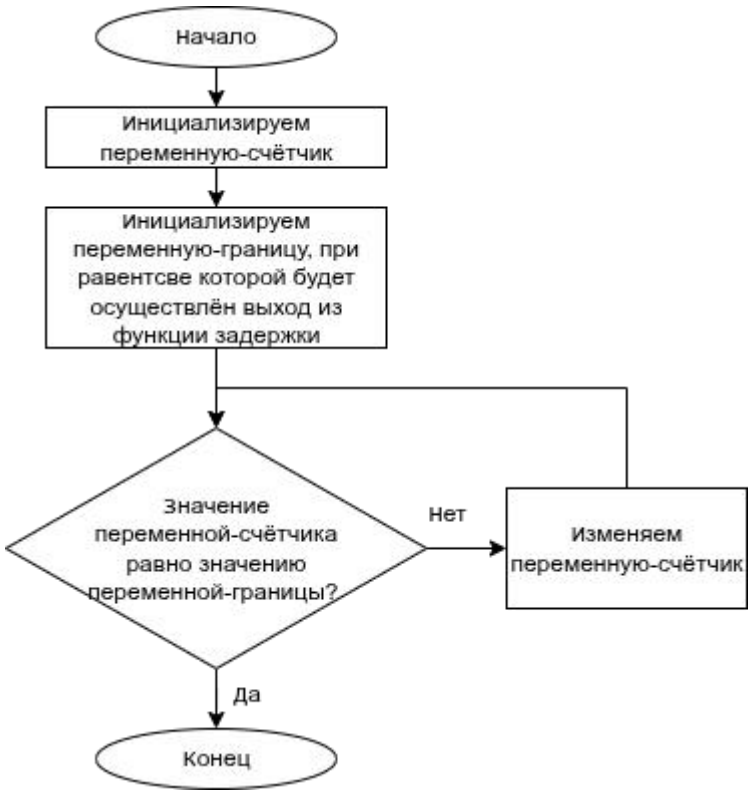

Рисунок 5 - блок-схема функции ожидания.

3. Аппаратные средства

Компоненты

1. микроконтроллер: ATmega328P;
2. модуль RGB-светодиода HW-479: общий анод, 20мА на канал, резисторы 150 Ом распаяны на плате;
3. модуль инкрементного энкодера: GSMIN AK291, RC-цепи для уменьшения дребезга контактов распаяны на плате.
4. светодиод, встроенный в плату Arduino Nano. Привязан к пину PB5 (пин D13 в абсолютной нумерации);
5. провода для подключения компонентов;
6. беспаечная макетная плата.

Схема подключения

- энкодер: фазы A и B к выводам PD2 (пин D2 в абсолютной нумерации), PD3 (пин D3 в абсолютной нумерации) без подтяжки к питанию, также подключено питание 5В и земля (GND);
- RGB-светодиод: каналы R, G, B к выводам OCR0A (пин PD6)(пин D6 в абсолютной нумерации), OCR0B (пин PD5)(пин D5 в абсолютной нумерации), OCR1AL (пин PB1)(пин D9 в абсолютной нумерации).

Более детальная схема подключения представлена на рисунке 6.

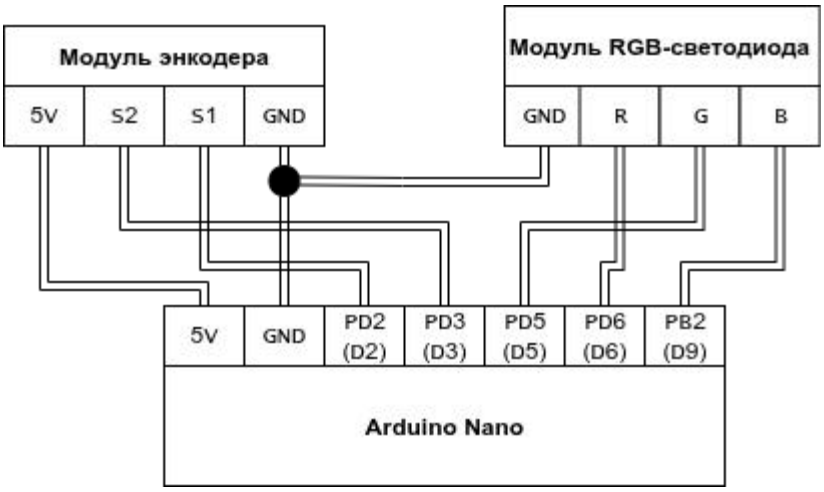

Рисунок 6 - детальная схема подключения.

Обоснование выбора

- ATmega328P имеет достаточное количество таймеров для реализации 3-канального ШИМ;
- Энкодер GSMIN AK291 обеспечивает точное позиционирование и надежность, а также имеет аппаратную защиту от дребезга (RC-цепи).

- Модуль RGB-светодиода уже имеет резисторы, что позволяет не устанавливать резисторы на макетную плату, освобождаяя место.

Собранное устройство

Фото собранного устройства представлены на рисунках 7, 8.

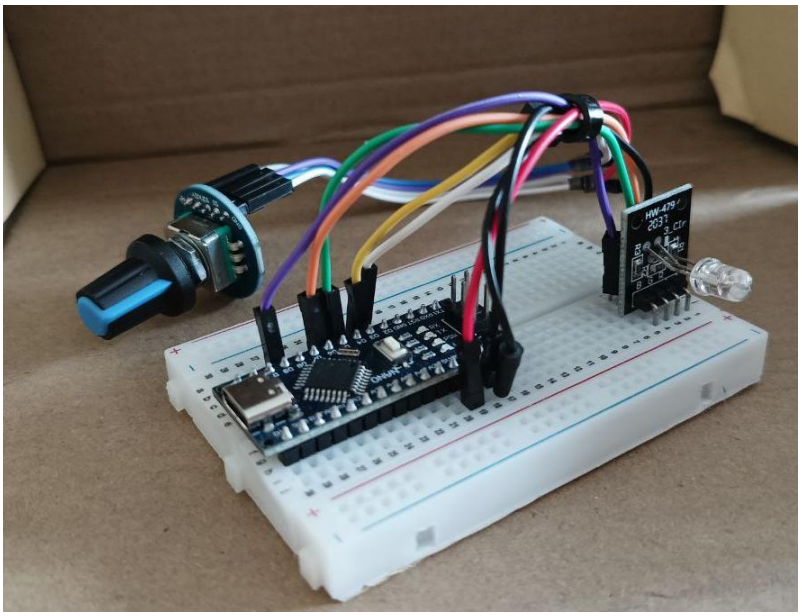

Рисунок 7 - фото устройства с ракурса №1.

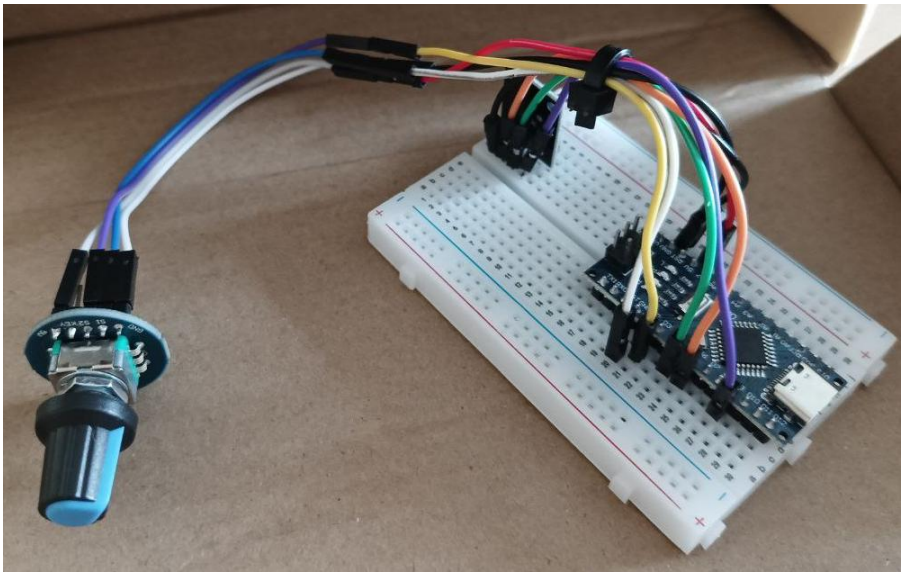

Рисунок 8 - фото устройства с ракурса №2.

4. Использование микроконтроллера

Используемые выводы

- D2 (INT0) - фаза A энкодера;
- D3 (INT1) - фаза B энкодера;
- D6 (OCR0A)(PD6) - канал Red;
- D5 (OCR0B)(PD5) - канал Green;
- D9 (OCR1AL)(PB1) - канал Blue;
- D13 (PB5) - отладочный встроенный светодиод.

Периферийные модули

- Таймер/Счетчик 0: генерация ШИМ для канала R и G;
- Таймер/Счетчик 1: генерация ШИМ для каналов B;
- Внешние прерывания INT0, INT1: обработка энкодера;
- Отладочный встроенный светодиод: пин PB5 (пин D13 в абсолютной нумерации);

Расчет параметров ШИМ

Частота ШИМ: 7812.5 Гц (Fast PWM mode)
Разрешение: 8 бит
Коэффициент деления: 8

5. Общие сведения о программе

Наименование: RGB Encoder Control System

Язык программирования: Ассемблер AVR

Среда разработки: Neovim

Дата создания: 15.05.2023

Автор: Грачев Александр Витальевич

Группа: КРБО-03-23

Функциональное назначение

- Управление цветом RGB-светодиода через энкодер;
- Плавное изменение цвета и интенсивности.

Аппаратное обеспечение

- Плата Arduino Nano с микроконтроллером ATmega328P;
- RGB-светодиод;
- Инкрементный энкодер GSMIN AK291;

6. Структура программного обеспечения

Модули программы:

- Инициализация периферии (таймеры, прерывания);
- Обработчик прерываний энкодера;
- Алгоритм преобразования положения в цвет;
- Генерация ШИМ сигналов;
- Основной цикл программы (индикация работы).

Программа содержит множество процедур, которые облегчают понимание кода, повышают удобство программирования, придают код стуктуру. В программе содержатся следующие процедуры:

- reset - в ней производится настройка всей периферии и установна начальных значений переменных;
- main_loop - основной цикл программы, иммитирует решение какой-то задачи микроконтроллером;
- update_leds - устанавливает изменяет состояния светодиодов;
- delay_100ms, delay_loop - реализовывают процедуру ожидания (простоя);
- ISR_INT0, ISR_INT1 - процедуры, которые запускаются при срабатывании прерывания;
- case_A_1, case_A_0, inc_pos_A, dec_pos_A, end_int0, case_B_1, case_B_0, inc_pos_B, dec_pos_B, end_int1 - вспомогательные процедуры для удобной обработки изменения состояния энкодера.

7. Структура данных

Используемые регистры

- R16 - регистр, используемый как буффер для взаимодействия с другими регистрами (сумма, вычитание и т.д.)
- R17-R19: регистры-счётчики, необходимы для реализации функции задержки
- R20, R21, R22: значения ШИМ (от 0 до 255) для каналов R, G, B соответственно
- R23: текущее положение энкодера

Организация памяти

- Данные цвета хранятся в регистрах общего назначения
- Позиция энкодера хранится в регистре общего назначения

Выходные сигналы

- ШИМ сигналы на выводах OCR0A (PD6), OCR0B (PD5), OCR1AL (PB2)

8. Методика и результаты тестирования

Методика тестирования

- Проверка реакции на вращение энкодера
- Проверка цветовых переходов
- Тест на стабильность работы при разных скоростях вращения
- Проверка функционирования основного цикла программы

Фото во время тестирования

На рисунках 9-15 изображена реакция системы при разных положениях энкодера.

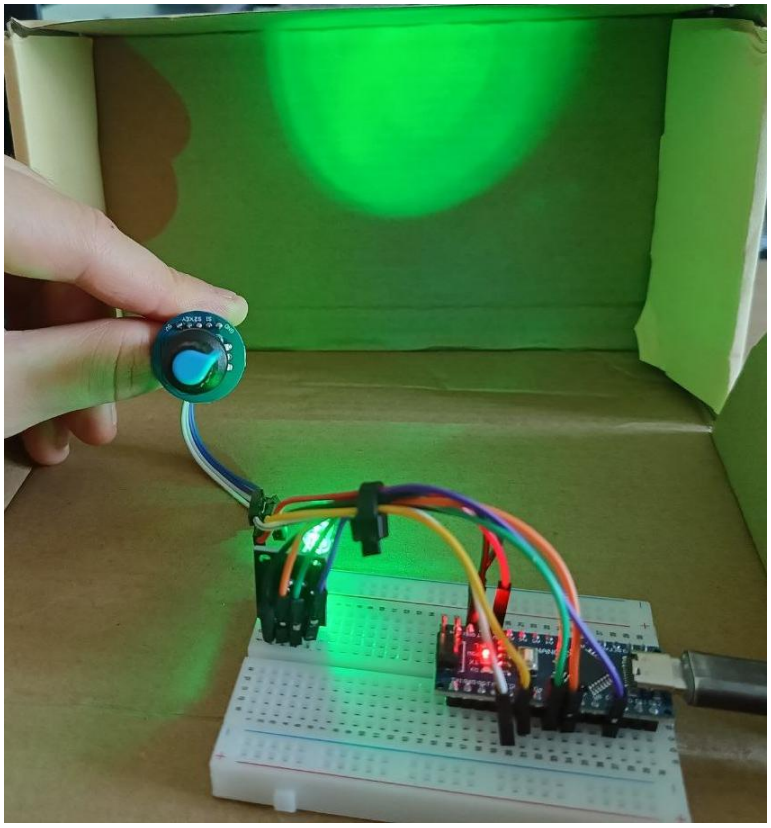

Рисунок 9 - реакция системы на положение энкодера №1.

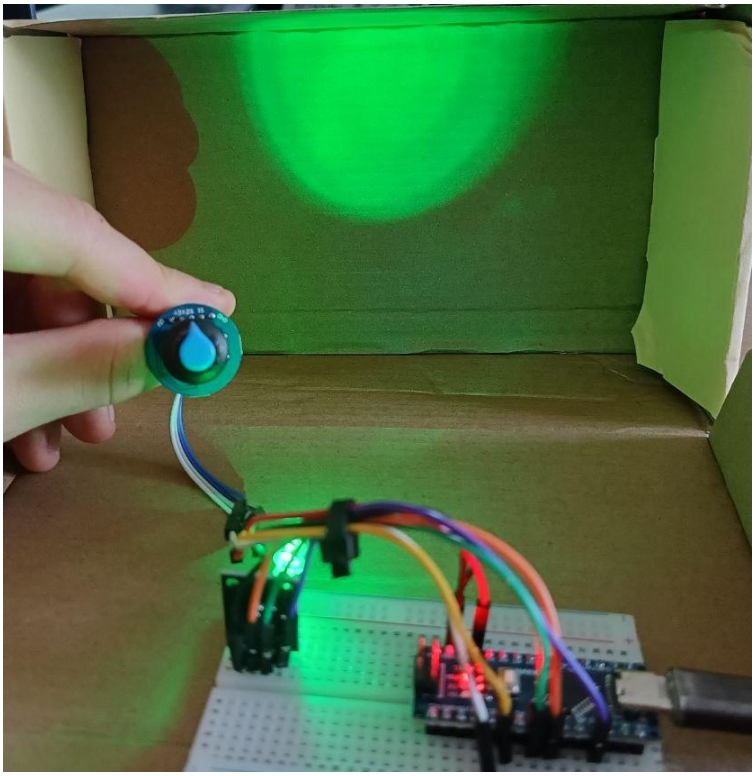

Рисунок 10 - реакция системы на положение энкодера №2.

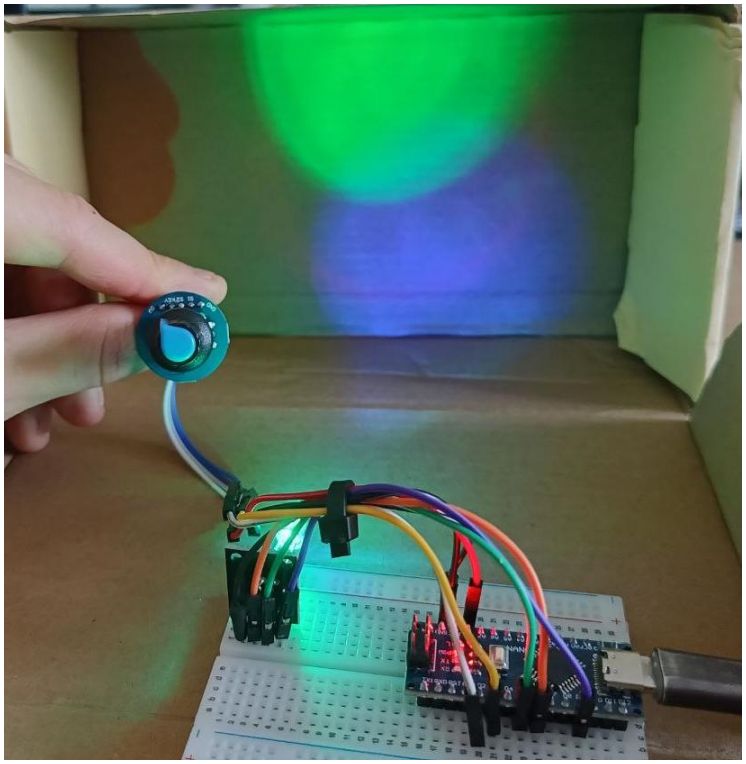

Рисунок 11 - реакция системы на положение энкодера №3.

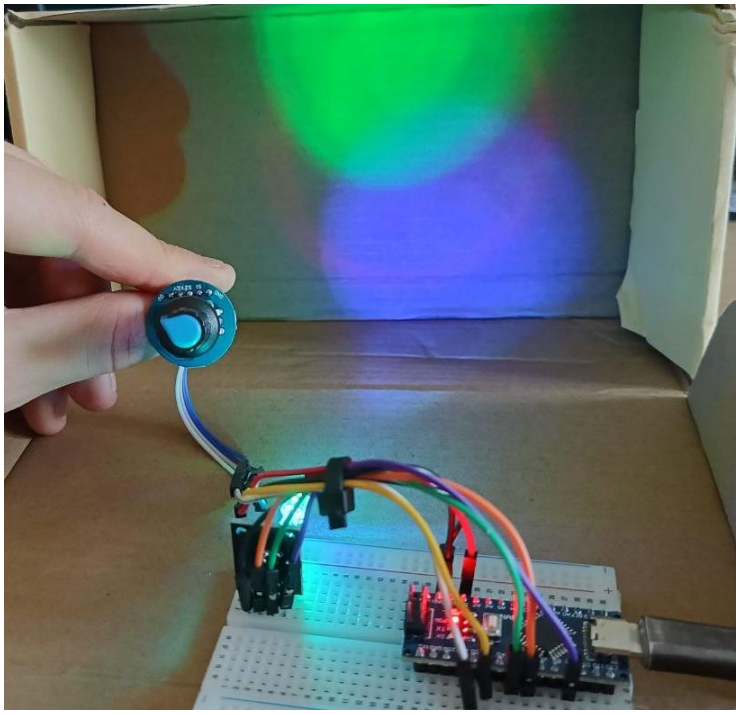

Рисунок 12 - реакция системы на положение энкодера №4.

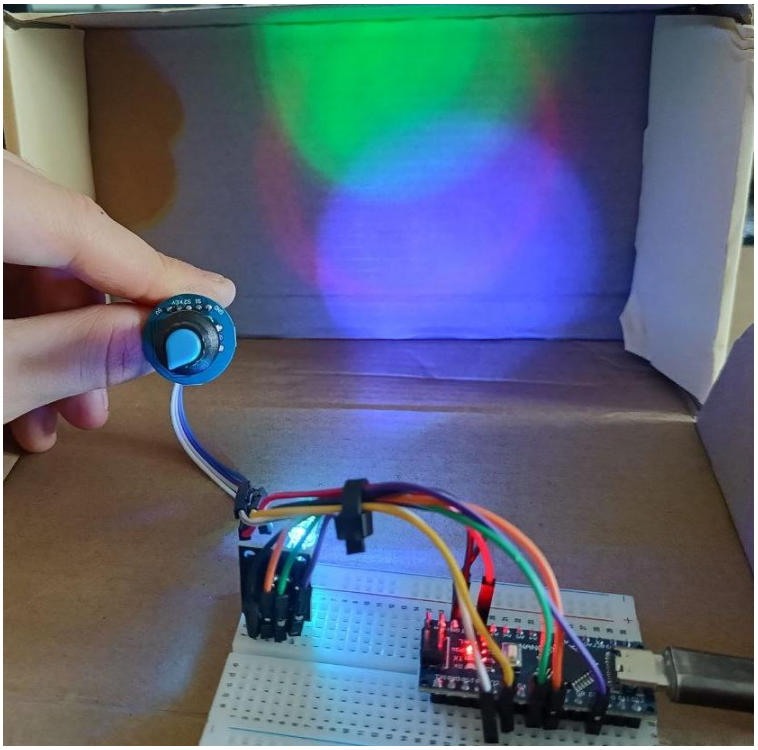

Рисунок 13 - реакция системы на положение энкодера №5.

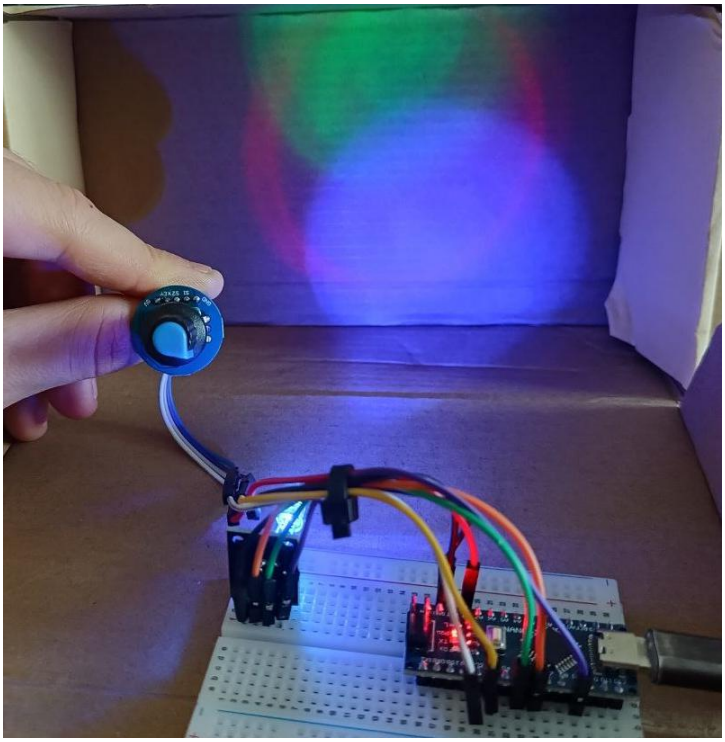

Рисунок 14 - реакция системы на положение энкодера №6.

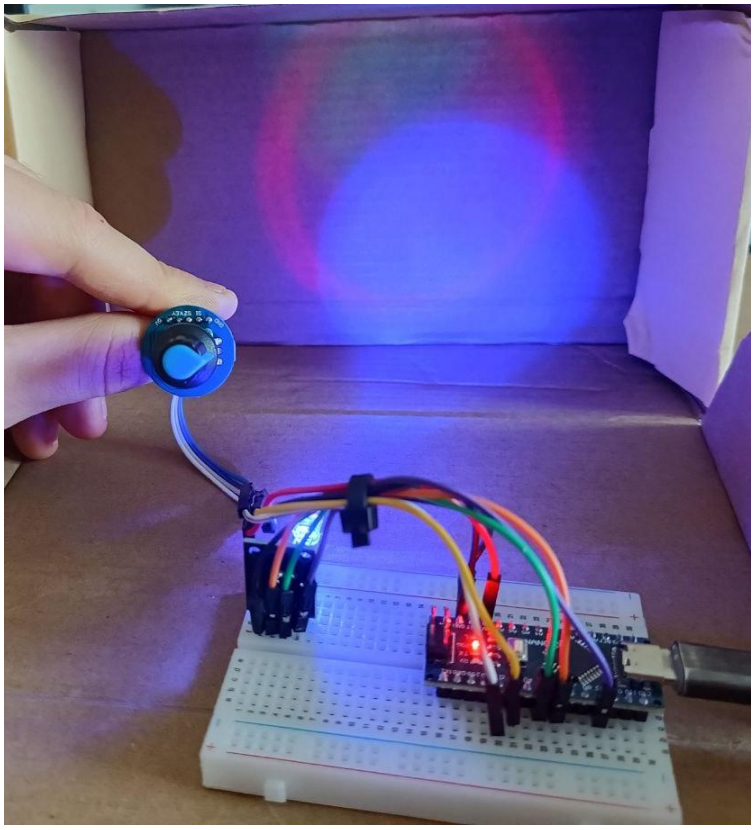

Рисунок 15 - реакция системы на положение энкодера №7.

Результаты

- Система корректно регирует на изменение положения энкодера. Прерывания обрабатываются стабильно и корректно. Программа не зависает.
- При изменении положения энкодера цвет RGB-светодиода меняется так, как и ожидалось.
- Микроконтроллер успешно обрабатывает сигналы при разных скоростях вращения ручки энкодера.
- Основной цикл программы (мигание встроенным светодиодом) работает в штатном режиме при изменении положения энкодера. Что является следствием корректной настройки прерываний.

9. Исходный код

Исходный код на Ассемблере

Полный листинг программы приведён в приложении А.

Список использованной литературы

1. Принцип работы инкрементального энкодера. — Текст : электронный // Иннодрайв : [сайт]. — URL: https://innodrive.ru/articles/enkodery/inkrementalnyj-ehnkoder/ (дата обращения: 21.09.2025).
2. Эффективное управление светодиодами: подробное изучение ШИМ-регулировки яркости. — Текст : электронный // MYLIKELED : [сайт]. — URL: https://mylikeled.com/ru/эффективное-управление-светодиодами--глубокое-погружение-в-ШИМ-регулирование-яркости/ (дата обращения: 21.09.2025).
3. Регулирование яркости светодиодов, принципы ШИМ-регулирования. — Текст : электронный // ELECTRIC INFO : [сайт]. — URL: https://electrik.info/main/praktika/824-regulirovanie-yarkosti-svetodiodov.html (дата обращения: 21.09.2025).
4. Программирование МК AVR на языке assembler в среде Linux. — Текст : электронный // HABR : [сайт]. — URL: https://habr.com/ru/articles/373677/ (дата обращения: 21.09.2025).
5. Ассемблер для микроконтроллера с нуля. Часть 1. Начало пути. — Текст : электронный // DATAGOR : [сайт]. — URL: https://datagor.ru/microcontrollers/3167-assembler-for-mcu-easy-part1.html (дата обращения: 21.09.2025).
6. Atmega48pa/88pa/168pa/328p. — Текст : электронный // files.amperka.ru : [сайт]. — URL: https://files.amperka.ru/datasheets/ATmega328.pdf (дата обращения: 21.09.2025).
7. ATMEGA328P Datasheet (PDF) - ATMEL Corporation. — Текст : электронный // Electronic Components Datasheet Search : [сайт]. — URL: https://www.alldatasheet.com/datasheet-pdf/pdf/241077/ATMEL/ATMEGA328P.html (дата обращения: 21.09.2025).

ПРИЛОЖЕНИЕ А

```
; Определяем микроконтроллер
.include "m328Pdef.inc"

; Настройка настроек
.def temp = r16
.def counter1 = r17
.def counter2 = r18
.def counter3 = r19

.def red_led_state = r20    ; Яркость красного (0-255)
.def green_led_state = r21  ; Яркость зелёного (0-255)
.def blue_led_state = r22   ; Яркость синего (0-255)
.def pos_reg = r23          ; Позиция энкодера

; Определение пинов
.equ PIN_A = 2    ; PD2
.equ PIN_B = 3    ; PD3

; Начало программы
.cseg
; Векторы прерываний
.org 0x0000
    jmp reset
.org INT0addr
    jmp ISR_INT0
.org INT1addr
    jmp ISR_INT1


.org 0x100
```

```
reset:
    ; Настройка стека
    ldi temp, low(RAMEND)
    out SPL, temp
    ldi temp, high(RAMEND)
    out SPH, temp

    ; Настройка пинов энкодера как входов с подтяжкой
    cbi DDRD, PIN_A          ; D2 как вход (канал A энкодера)
    cbi PORTD, PIN_A         ; Выключить подтяжку к питанию
    cbi DDRD, PIN_B          ; D3 как вход (канал B энкодера)
    cbi PORTD, PIN_B         ; Выключить подтяжку к питанию

    ; Настройка прерываний для энкодера (по нисходящему фронту)
    ldi temp, (1<<ISC01)|(0<<ISC00)|(1<<ISC11)|(0<<ISC10)
    sts EICRA, temp
    ldi temp, (1<<INT0)|(1<<INT1) ; Разрешить INT0 и INT1
    out EIMSK, temp

    ; Настройка пинов D5 и D6 как выходов
    ldi temp, 0b01100000   ; D6 и D5 как выходы
    out DDRD, temp

    ; Настройка пина D9 как выхода
    ldi temp, 0b00000010   ; D9 как выход
    out DDRB, temp

    ; Настройка ТАЙМЕРА0
    ldi temp, (1<<COM0A1)|(1<<COM0B1)|(1<<WGM01)|(1<<WGM00)
    out TCCR0A, temp
    ldi temp, (1<<CS01)        ; Предделитель = 8
    out TCCR0B, temp
```

```
    ; Настройка ТАЙМЕРА1
    ldi temp, (1<<COM1A1)|(1<<WGM10)   ; Non-inverting mode на канале A,
Fast PWM 8-bit
    sts TCCR1A, temp
    ldi temp, (1<<WGM12)|(1<<CS11)      ; Fast PWM mode, предделитель =
8
    sts TCCR1B, temp

    ; Задание начальных значений переменных
    ldi red_led_state, 0
    ldi green_led_state, 255
    ldi blue_led_state, 50
    ldi pos_reg, 0       ; Начальная позиция энкодера

    ; Применить начальные значения
    rcall update_leds

    ; Разрешить глобальные прерывания
    sei

    rjmp main_loop



; Главный бесконечный цикл
main_loop:
    ; Обновляем яркость на основе позиции энкодера
    mov red_led_state, pos_reg
    mov blue_led_state, pos_reg

    ldi temp, 255
    sub temp, pos_reg
    mov green_led_state, temp
```

```
    rcall update_leds

    ; Мигаем светодиодом на D13 для индикации работы
    sbi PORTB, 5
    rcall delay_100ms
    cbi PORTB, 5
    rcall delay_100ms

    rjmp main_loop


; Обновление светодиодов
update_leds:
    out OCR0B, green_led_state  ; D5 = Зеленый (0-255)
    out OCR0A, red_led_state    ; D6 = Красный (0-255)
    sts OCR1AL, blue_led_state  ; D9 = Синий (0-255)
    ret


; Подпрограмма задержки ~100ms
delay_100ms:
    push counter1
    push counter2
    push counter3

    ldi counter1, 13
    ldi counter2, 45
    ldi counter3, 215

delay_loop:
    dec counter3
```

```
    brne delay_loop
    dec counter2
    brne delay_loop
    dec counter1
    brne delay_loop

    pop counter3
    pop counter2
    pop counter1
    ret


; Обработчик прерывания для канала A (PIN_A)
ISR_INT0:
    push temp        ; Сохраняем используемые регистры
    push r17
    push r18
    in r18, SREG    ; Сохраняем регистр статуса
    push r18

    ; Чтение состояния пинов
    in r17, PIND
    andi r17, (1<<PIN_A)|(1<<PIN_B)

    ; Проверка состояния канала A
    sbrs r17, PIN_A
    rjmp case_A_0

case_A_1:
    ; Если PIN_A = 1
    sbrc r17, PIN_B
    rjmp dec_pos_A    ; Если PIN_B = 1
```

```
    rjmp inc_pos_A    ; Если PIN_B = 0

case_A_0:
    ; Если PIN_A = 0
    sbrc r17, PIN_B
    rjmp inc_pos_A    ; Если PIN_B = 1
    rjmp dec_pos_A    ; Если PIN_B = 0

inc_pos_A:
    ; Увеличение позиции
    ldi temp, 8
    add pos_reg, temp
    rjmp end_int0

dec_pos_A:
    ; Уменьшение позиции
    ldi temp, 8
    sub pos_reg, temp

end_int0:
    pop r18         ; Восстанавливаем регистр статуса
    out SREG, r18
    pop r18
    pop r17
    pop temp
    reti

; Обработчик прерывания для канала B (PIN_B)
ISR_INT1:
    push temp       ; Сохраняем используемые регистры
    push r17
    push r18
```

```
    in r18, SREG    ; Сохраняем регистр статуса
    push r18

    ; Чтение состояния пинов
    in r17, PIND
    andi r17, (1<<PIN_A)|(1<<PIN_B)

    ; Проверка состояния канала B
    sbrs r17, PIN_B
    rjmp case_B_0

case_B_1:
    ; Если PIN_B = 1
    sbrc r17, PIN_A
    rjmp inc_pos_B  ; Если PIN_A = 1
    rjmp dec_pos_B  ; Если PIN_A = 0

case_B_0:
    ; Если PIN_B = 0
    sbrc r17, PIN_A
    rjmp dec_pos_B  ; Если PIN_A = 1
    rjmp inc_pos_B  ; Если PIN_A = 0

inc_pos_B:
    ; Увеличение позиции
    ldi temp, 8
    add pos_reg, temp
    rjmp end_int1

dec_pos_B:
    ; Уменьшение позиции
    ldi temp, 8
```

```
    sub pos_reg, temp

end_int1:
    pop r18         ; Восстанавливаем регистр статуса
    out SREG, r18
    pop r18
    pop r17
    pop temp
    reti
```