МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ»

Кафедра ИиСП

Отчет
по лабораторной работе № 5

по дисциплине «Машинно-зависимые языки программирования»
Вариант 2

Выполнил: ст. гр. ПС-14
Сайфутдинов Э.Р.
Проверил: доцент, доцент кафедры ИиСП Баев А.А.

г. Йошкар-Ола
2024

Цель работы:

Научиться собирать схемы в протеусе и писать к ним код.

Задания на лабораторную работу:

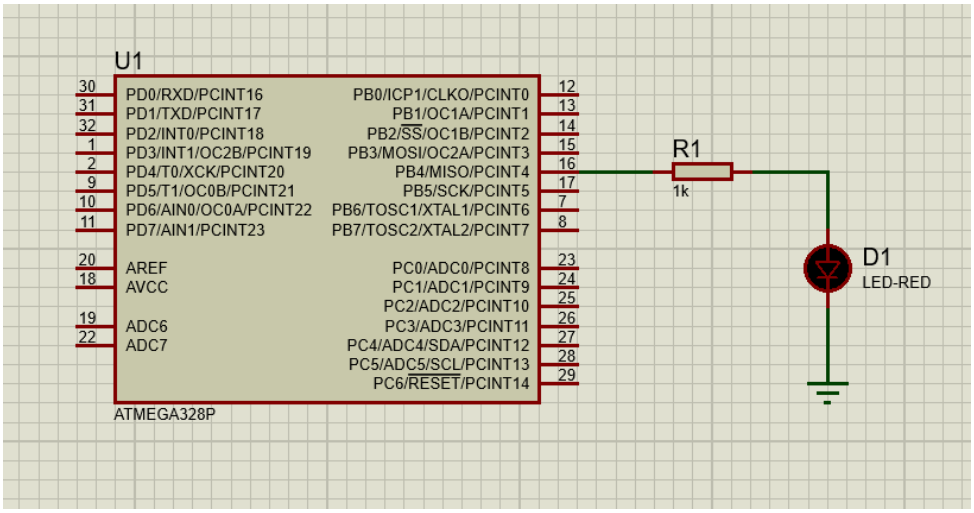
Собрать несколько схем в протеусе, включающих светодиод или 7-сегментный индикатор. Написать к собранным схемам код на С.

1. **Теоретические сведения**

   Учебное пособие “Применение микроконтроллеров в радиотехнических и биомедицинских системах”.

2. Практическая часть

1) Подключение светодиода



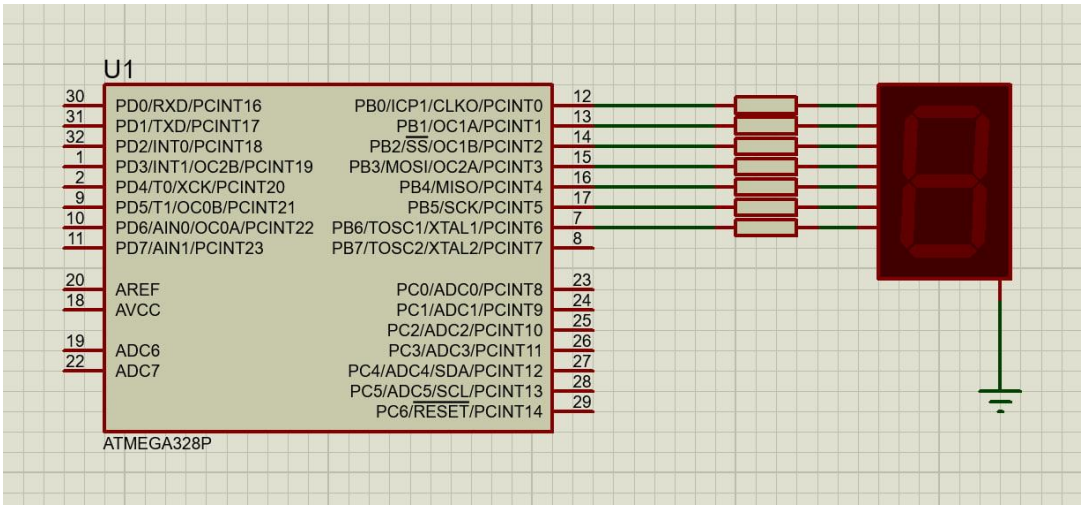
Код для собранной схемы:

```
#include <avr/io.h>
int main(void)
{
	DDRB = 0b00010000;
	PORTB = 0b00010000;
	while(1){}
}
```

Оптимизированный код:

```
#include <avr/io.h>
int main(void)
{
	DDRB |= (1 << PORTB4);
	PORTB |= (1 << PORTB4);
	while(1);
}
```

2) Подключение 7-сегментного индикатора



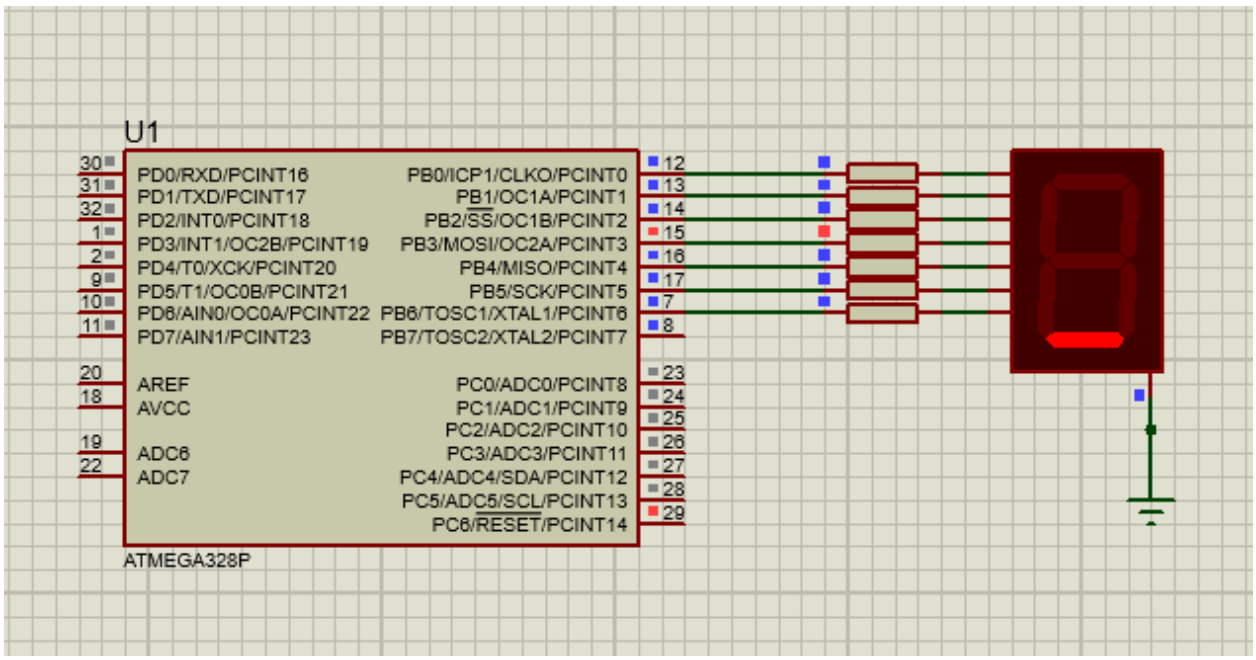
Код для собранной схемы:

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
int main(void)
{
	DDRB = 0xFF;
	while(1)
	{
		for( int i=0; i<6; i++){
			PORTB = (1<<i);
			_delay_ms(200);
		}
	}
}
```

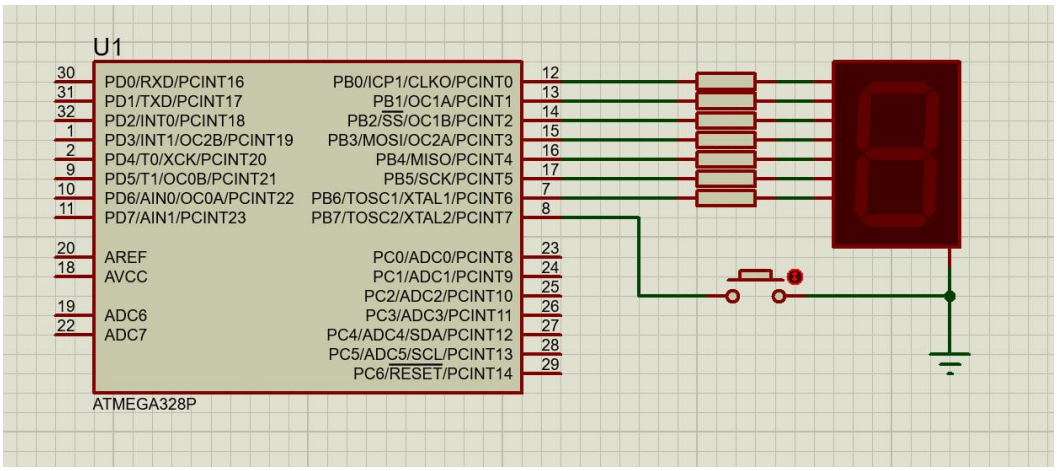
Оптимизированный код:

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
int main(void)
{
	uint8_t i = 0;
	DDRB = 0xFF; // Устанавливаем порт B как выход
	while (1)
	{
		PORTB = (1 << i); // Включаем светодиод на позиции i
		_delay_ms(200); // Задержка 200 мс
		i++;
		// Если i достигает 6, сбрасываем его до 0
		if (i == 6)
		i = 0;
	}
}
```

Данный код заставяет загораться элементы 7-сегментного индикатора по часовой стрелке



3) Подключение 7-сегментного индикатора с кнопкой



Код для данной схемы:

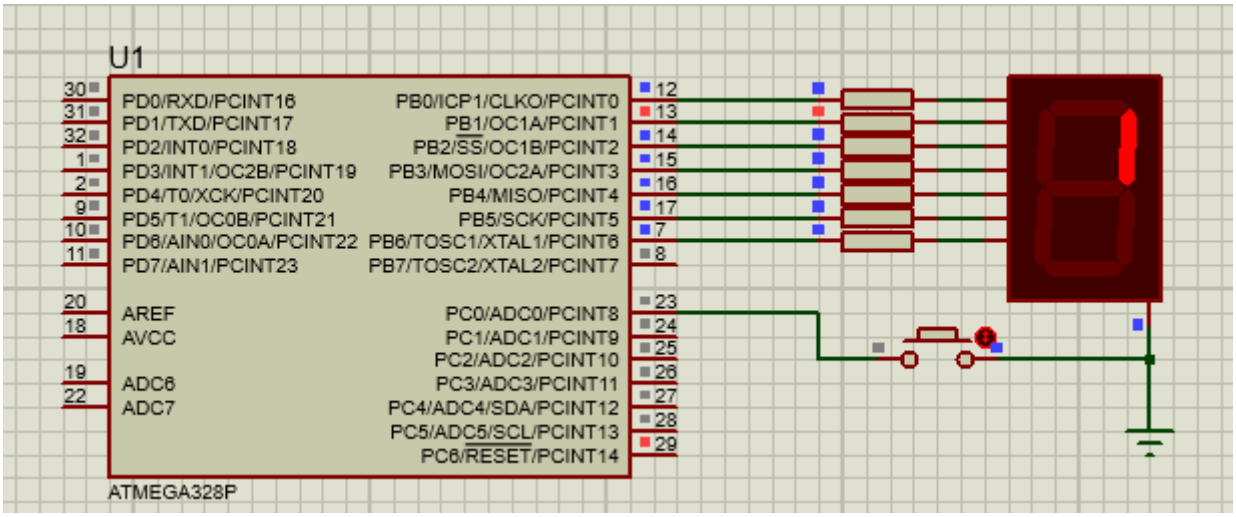
```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
int main(void)
{
	DDRB = 0xFF & ~(1<<PB7);// PB7 – вход, остальные - выходы
	PORTB |= (1<<PB7); // включение подтягивающего резистора
	int button = 0; // вспомогательная переменная
	while(1)
	{
		for(int i = 0; i < 6; i++){
			button = PINB & (1<<PB7); //чтение состояния PB7
			if(button != 0){
				PORTB = (1<<i);
				}else{
```

```
                        PORTB = (0x20>>i);
                    }
                _delay_ms(200);
            }
        }
}
```

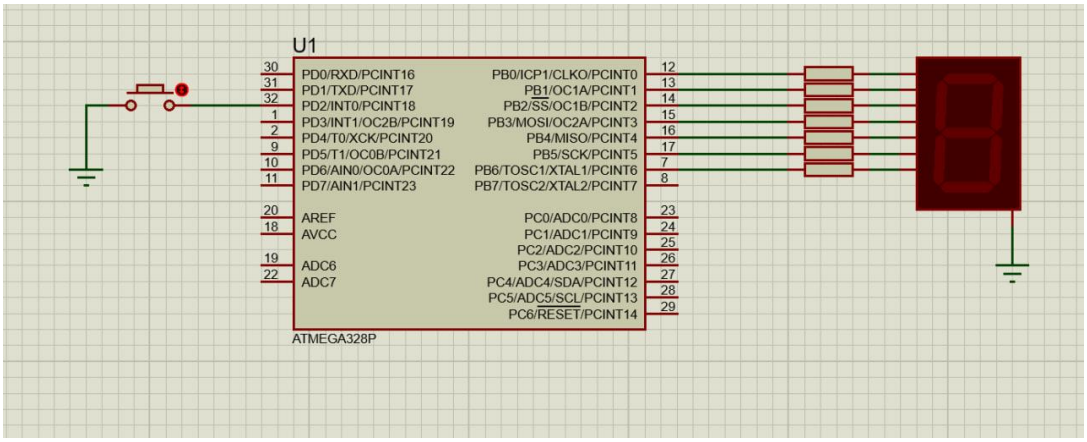
Оптимизированный код:

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
int main(void)
{
	DDRB = 0xFF & ~(1 << PB7);// PB7 – вход,остальные - выходы
	PORTB |= (1 << PB7); // включение подтягивающего резистора
	int button = 0; // вспомогательная переменная
	while(1)
	{
		for(int i = 0; i < 6; i++){
			button = PINB & (1 << PB7); //чтение состояния PB7
			if(button != 0){
				PORTB = (1 << i);
			}
			else{
				PORTB = (0x20 >> i);
			}
			_delay_ms(200);
		}
	}
}
```

Дополнение к предыдущему коду в том, что при удержании кнопки элементы 7-сегментного индикатора загораются против часовой стрелки



4) Секундомер



Секундомер 1(без использования прерывания):

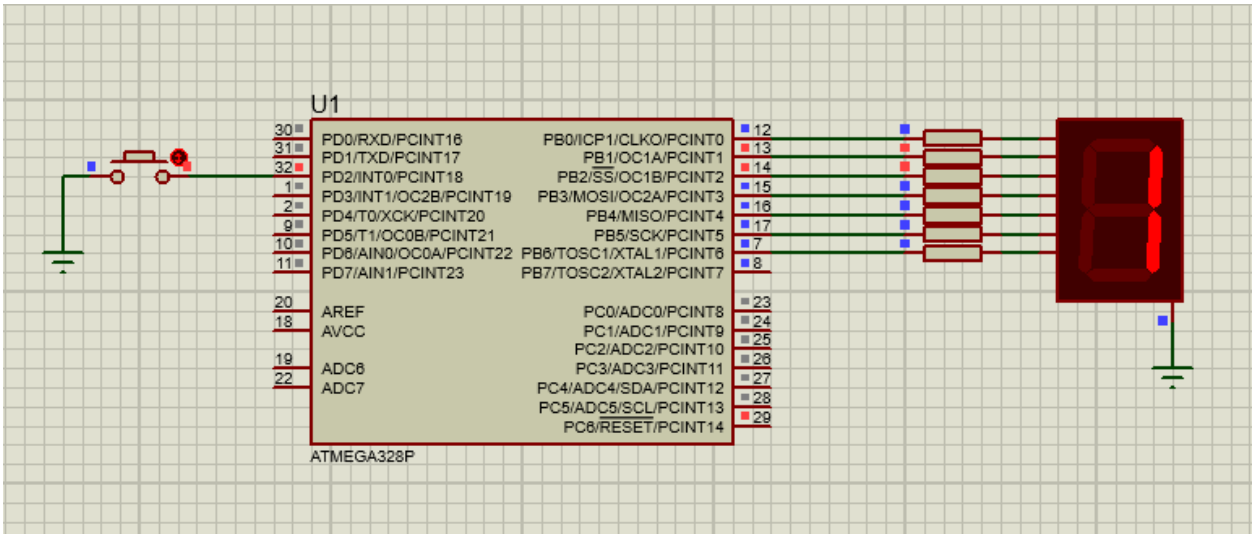
```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
uint8_t segments[]={
	// ___GFEDCBA
	0b00111111, // 0 - A, B, C, D, E, F
	0b00000110, // 1 - B, C
	0b01011011, // 2 - A, B, D, E, G
	0b01001111, // 3 - A, B, C, D, G
	0b01100110, // 4 - B, C, F, G
	0b01101101, // 5 - A, C, D, F, G
	0b01111101, // 6 - A, C, D, E, F, G
	0b00000111, // 7 - A, B, C
	0b01111111, // 8 - A, B, C, D, E, F, G
	0b01101111, // 9 - A, B, C, D, F, G
};

int main(void)
{
	DDRB = 0xFF; // порт B на выход
	DDRD &= ~(1<<PD2); // вывод PD2 на вход
	PORTD |= (1<<PD2); // подтяжка PD2 включена
	int button = 0;
	int switch_state = 0;
	int counter = 0;
	while(1)
	{
		button = PIND & (1<<PD2); //опрос кнопки
		if(button == 0){
			while((PIND & (1<<PD2)) == 0);//ожидание отпускания
			if(switch_state == 0){
				switch_state = 1;
				} else {
				switch_state = 0;
				counter = 0;
			}
		}
		if(switch_state == 0){
			if(counter < 10){
```

```
                        PORTB = segments[counter++];
                        _delay_ms(1000);
                        } else {
                        counter = 0;
                        PORTB = segments[counter++];
                        _delay_ms(1000);
                  }
            }
      }
}
```

Оптимизированный код:

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
uint8_t segments[]={
      // ___GFEDCBA
      0b00111111, // 0 - A, B, C, D, E, F
      0b00000110, // 1 - B, C
      0b01011011, // 2 - A, B, D, E, G
      0b01001111, // 3 - A, B, C, D, G
      0b01100110, // 4 - B, C, F, G
      0b01101101, // 5 - A, C, D, F, G
      0b01111101, // 6 - A, C, D, E, F, G
      0b00000111, // 7 - A, B, C
      0b01111111, // 8 - A, B, C, D, E, F, G
      0b01101111, // 9 - A, B, C, D, F, G
};
int main(void)
{
      DDRB = 0xFF; // порт B на выход
      DDRD &= ~(1 << PD2); // вывод PD2 на вход
      PORTD |= (1 << PD2); // подтяжка PD2 включена
      int button = 0;
      int switch_state = 0;
      int counter = 0;
      while(1)
      {
            button = PIND & (1 << PD2); //опрос кнопки
            if(button == 0){
                  while((PIND & (1 << PD2)) == 0);//ожидание отпускания
                  if(switch_state == 0){
                        switch_state = 1;
                  }
                  else {
                        switch_state = 0;
                        counter = 0;
                  }
            }
            if(switch_state == 0){
                  if(counter < 10){
                        PORTB = segments[counter++];
                  }
                  else {
                        counter = 0;
```

```
                        PORTB = segments[counter++];
                    }
                    _delay_ms(1000);
                }
        }
}
```



Секундомер 2:

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
uint8_t segments[]={ //0b01111111
        // ___GFEDCBA
        0b00111111, // 0 - A, B, C, D, E, F
        0b00000110, // 1 - B, C
        0b01011011, // 2 - A, B, D, E, G
        0b01001111, // 3 - A, B, C, D, G
        0b01100110, // 4 - B, C, F, G
        0b01101101, // 5 - A, C, D, F, G
        0b01111101, // 6 - A, C, D, E, F, G
        0b00000111, // 7 - A, B, C
        0b01111111, // 8 - A, B, C, D, E, F, G
        0b01101111, // 9 - A, B, C, D, F, G
};
volatile int button = 0;
volatile int switch_state = 0;
volatile int counter = 0;
ISR(INT0_vect){
        // Обработчик прерывания
        if(switch_state == 0){
                switch_state = 1;
                } else {
                switch_state = 0;
                counter = 0;
        }
}
int main(void)
```

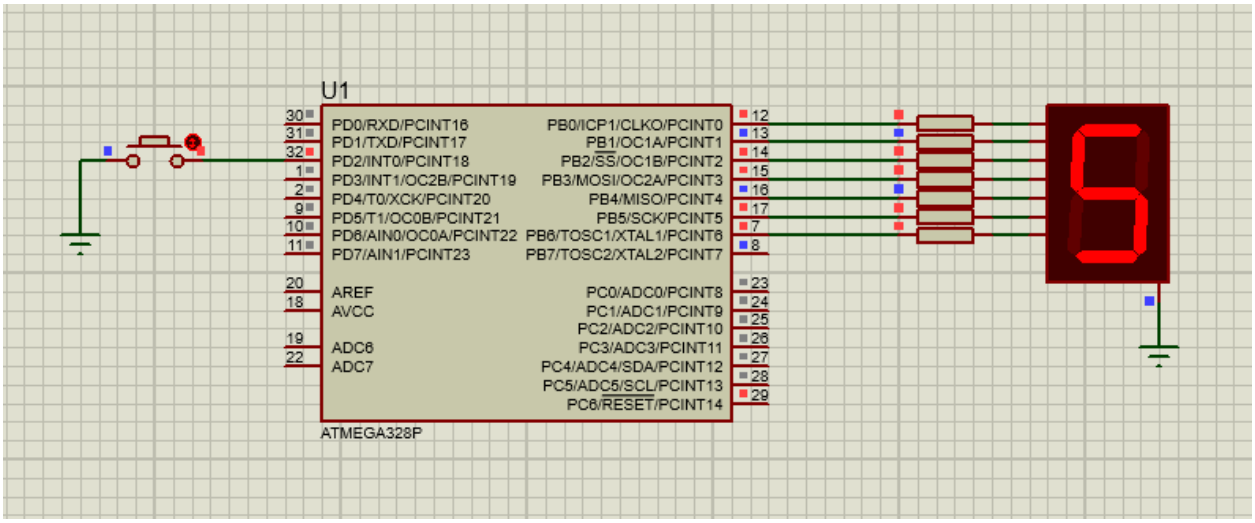
```
{
	DDRB = 0xFF;
	PORTD |= (1<<PD2);
	EIMSK |= (1<<INT0); //Включаем INT0
	EICRA |= (1<<ISC01); //Прерывание по спадающему фронту INT0
	sei(); //Глобальное разрешение прерываний
	while(1){
		if(switch_state == 0){
			if(counter < 10){
				PORTB = segments[counter];
				counter += 1;
				_delay_ms(500);
				} else {
				counter = 0;
				PORTB = segments[counter];
				counter += 1;
				_delay_ms(500);
			}
		}
	}
}
```

Оптимизированный код:

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
uint8_t segments[]={ //0b01111111
	// ___GFEDCBA
	0b00111111, // 0 - A, B, C, D, E, F
	0b00000110, // 1 - B, C
	0b01011011, // 2 - A, B, D, E, G
	0b01001111, // 3 - A, B, C, D, G
	0b01100110, // 4 - B, C, F, G
	0b01101101, // 5 - A, C, D, F, G
	0b01111101, // 6 - A, C, D, E, F, G
	0b00000111, // 7 - A, B, C
	0b01111111, // 8 - A, B, C, D, E, F, G
	0b01101111, // 9 - A, B, C, D, F, G
};
volatile int button = 0;
volatile int switch_state = 0;
volatile int counter = 0;
ISR(INT0_vect){
	// Обработчик прерывания
	if(switch_state == 0){
		switch_state = 1;
	}
	else {
		switch_state = 0;
		counter = 0;
	}
}
int main(void)
{
	DDRB = 0xFF;
	PORTD |= (1 << PD2);
	EIMSK |= (1 << INT0); //Включаем INT0
```

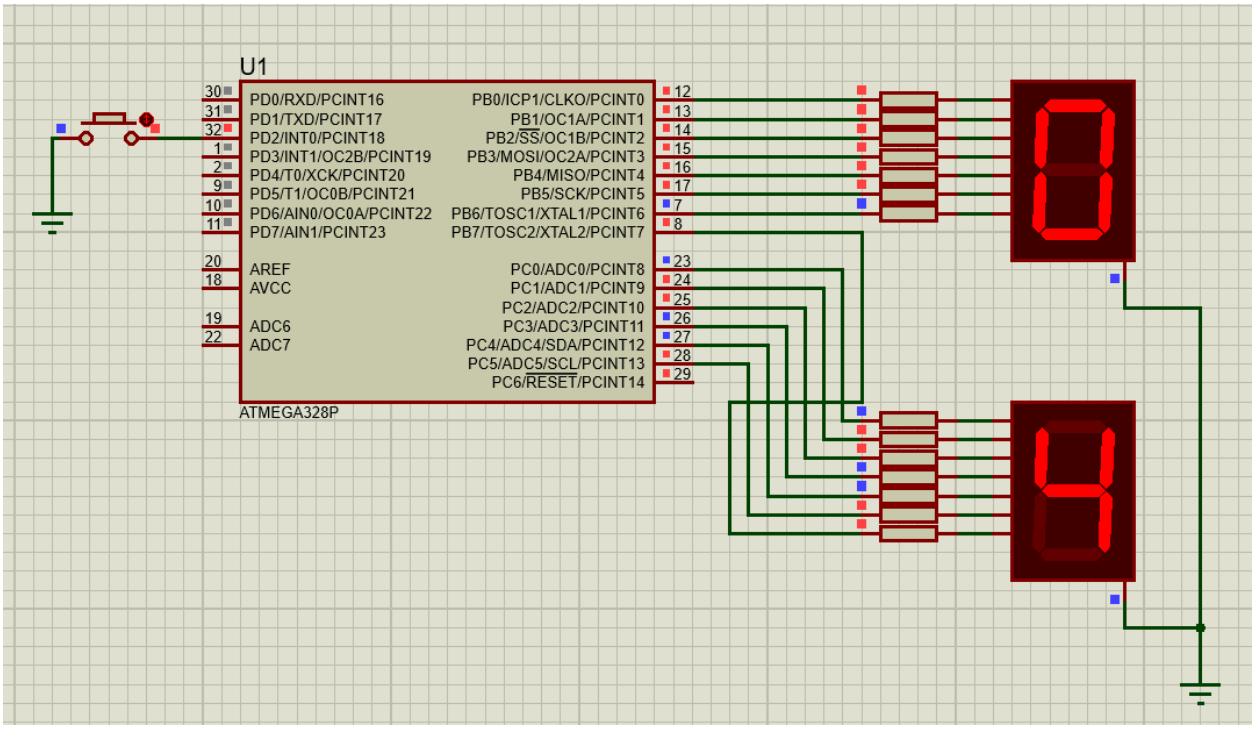
```
        EICRA |= (1 << ISC01); //Прерывание по спадающему фронту INT0
        sei(); //Глобальное разрешение прерываний
        while(1){
              if(switch_state == 0){
                    if(counter < 10){
                          PORTB = segments[counter];
                    }
                    else {
                          counter = 0;
                          PORTB = segments[counter];
                    }
                    counter += 1;
                    _delay_ms(500);
              }
        }
}
```

С использованием прерывания в секундомере 2, улучшилась отзвчивость кнопки на нажатия, а также добавилась функция сброса секундомера при повторном нажатии



Задание для 2 варианта:

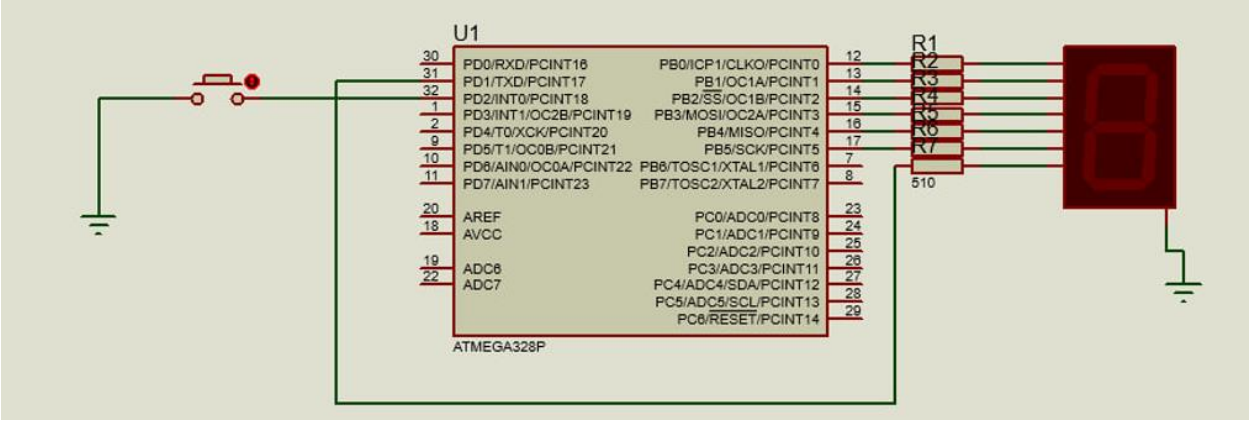
Напишите программу, работающую с кнопкой и двумя 7-сегментными индикаторами, которая будет считать число нажатий на кнопку до 15 раз.

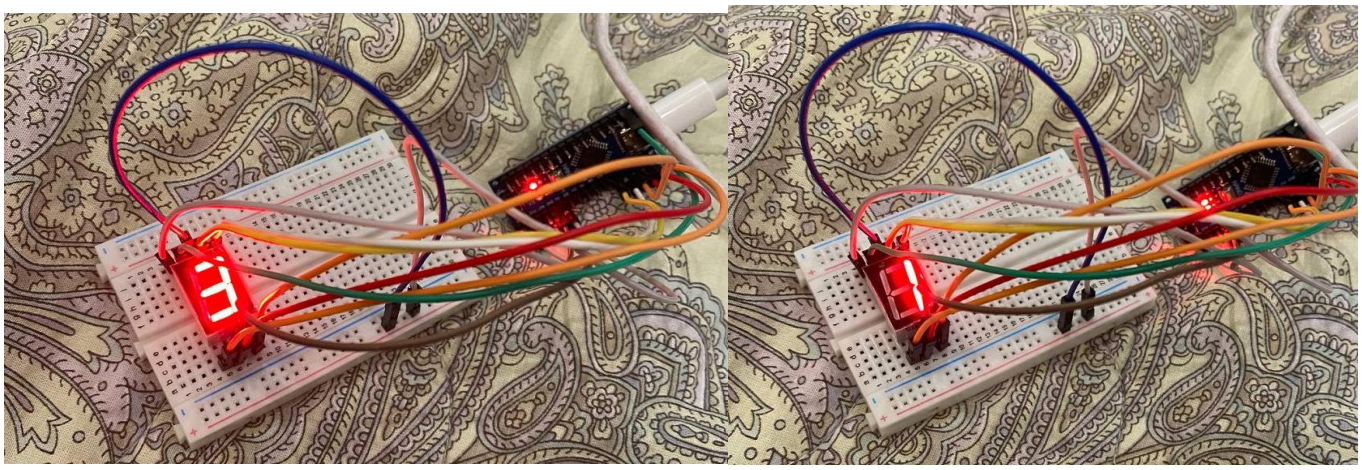



```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

uint8_t segments[]={
	//  ___GFEDCBA
	0b00111111, // 0 - A, B, C, D, E, F
	0b00000110, // 1 - B, C
	0b01011011, // 2 - A, B, D, E, G
	0b01001111, // 3 - A, B, C, D, G
	0b01100110, // 4 - B, C, F, G
	0b01101101, // 5 - A, C, D, F, G
	0b01111101, // 6 - A, C, D, E, F, G
	0b00000111, // 7 - A, B, C
	0b01111111, // 8 - A, B, C, D, E, F, G
	0b01101111, // 9 - A, B, C, D, F, G
	0b01111100, // 10 - A, B, C, D, E
	0b01110001, // 11 - C, D, E, F
	0b01101001, // 12 - A, E, F, G
	0b00111001, // 13 - B, C, D
	0b01110101, // 14 - A, C, E, F
	0b01111101  // 15 - A, B, C, E, G
};

volatile uint8_t press_count = 0; // Количество нажатий кнопки

ISR(INT0_vect)
{
	// Обработчик прерывания для кнопки
	if (press_count < 15) {
```

```
            press_count++;
            } else {
            press_count = 0; // Сброс счетчика после 15 нажатий
      }
}

int main(void)
{
      DDRB = 0xFF;  // Порт B - выход (для первого индикатора)
      DDRC = 0xFF;  // Порт C - выход (для второго индикатора)
      PORTD |= (1<<PIND2); // Включаем внутренний подтягивающий резистор для INT0
(кнопка)

      // Настройка прерывания INT0
      EIMSK |= (1<<INT0);  // Включаем INT0
      EICRA |= (1<<ISC01); // Прерывание по спадающему фронту INT0
      sei();  // Глобальное разрешение прерываний

      while(1)
      {
            // Отображаем количество нажатий на индикаторах
            PORTB = segments[press_count / 10];  // Десятки на первом индикаторе (порт
B)

            // Управление сегментом "G" на втором индикаторе (PINB7)
            if (segments[press_count % 10] & 0b01000000) {
                  PORTB |= (1 << PINB7);  // Включаем сегмент "G"
                  } else {
                  PORTB &= ~(1 << PINB7); // Выключаем сегмент "G"
            }

            PORTC = (segments[press_count % 10] & 0b00111111); // Выводим остальные
сегменты на порт C

            _delay_ms(200); // Задержка для обновления индикаторов
      }
}
```

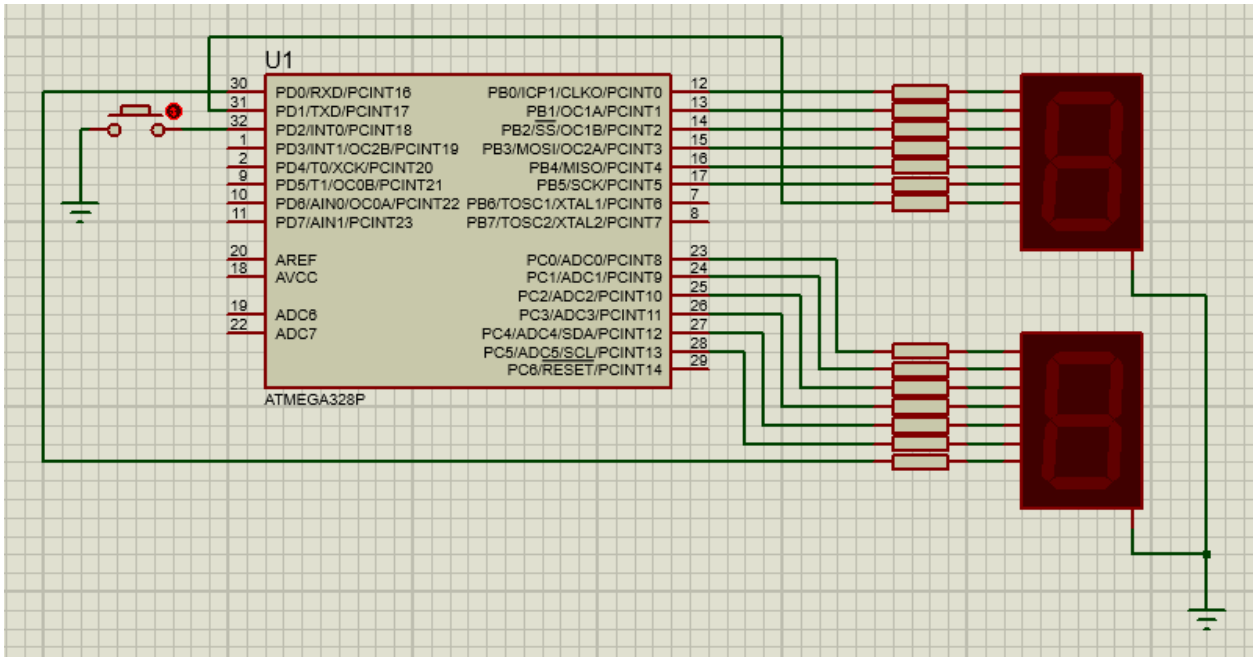
Работа с железом

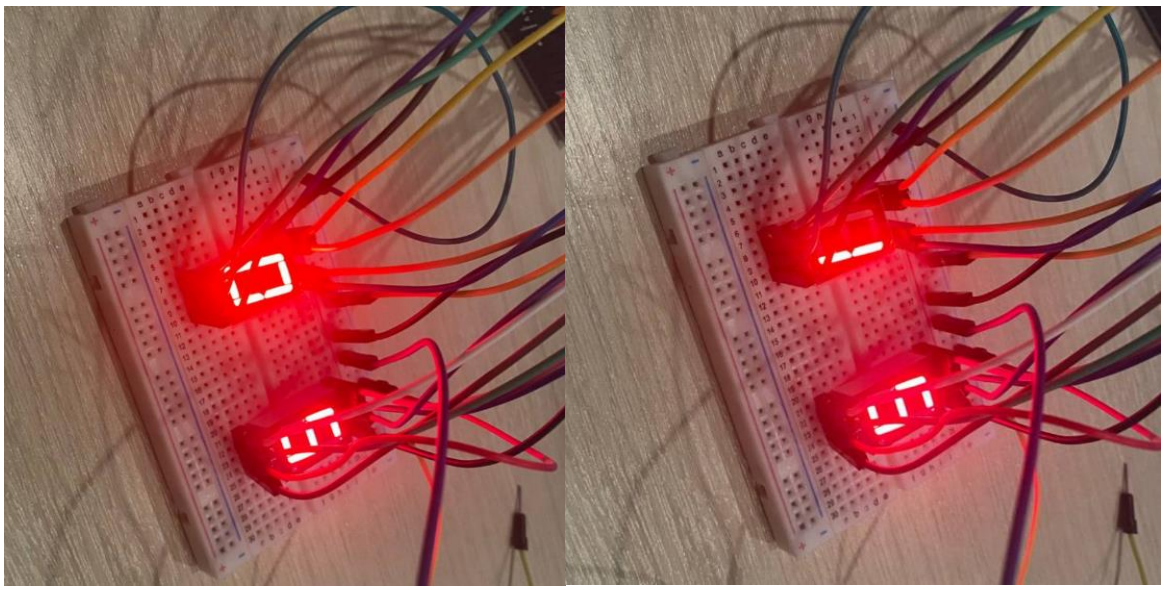




```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

uint8_t segments[] = {
	// ___GFEDCBA
	0b00111111, // 0 - A, B, C, D, E, F
	0b00000110, // 1 - B, C
	0b01011011, // 2 - A, B, D, E, G
	0b01001111, // 3 - A, B, C, D, G
	0b01100110, // 4 - B, C, F, G
	0b01101101, // 5 - A, C, D, F, G
	0b01111101, // 6 - A, C, D, E, F, G
	0b00000111, // 7 - A, B, C
	0b01111111, // 8 - A, B, C, D, E, F, G
	0b01101111  // 9 - A, B, C, D, F, G
};

volatile int button = 0;
volatile int switch_state = 0;
volatile int counter = 0;

ISR(INT0_vect) { // Обработчик прерывания
	if(switch_state == 0) {
```

```
                switch_state = 1;
                } else {
                switch_state = 0;
                counter = 0;
        }
}

int main(void) {
        DDRB = 0xFF; // PB0-PB7 как выходы (Сегменты A-F)
        DDRD |= (1 << PD3); // PD3 как выход (Сегмент G)

        PORTD |= (1 << PD2); // Подтяжка для BUTTON
        EIMSK |= (1 << INT0); // Включаем INT0
        EICRA |= (1 << ISC01); // Прерывание по спадающему фронту INT0
        sei(); // Глобальное разрешение прерываний

        while(1) {
                if(switch_state == 0) {
                        if(counter < 10) {
                                PORTB = segments[counter] & 0b01111101; // Сегменты A-F без G
                                if (segments[counter] & 0b10000000) {
                                        PORTD |= (1 << PD3); // Включаем сегмент G
                                        } else {
                                        PORTD &= ~(1 << PD3); // Выключаем сегмент G
                                }
                                counter += 1;
                                } else {
                                counter = 0;
                                PORTB = segments[counter] & 0b01111101; // Сегменты A-F без G
                                if (segments[counter] & 0b10000000) {
                                        PORTD |= (1 << PD3); // Включаем сегмент G
                                        } else {
                                        PORTD &= ~(1 << PD3); // Выключаем сегмент G
                                }
                                counter += 1;
                        }
                        _delay_ms(500);
                }
        }
}
```





```c
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

uint8_t segments[]={
	//  ___GFEDCBA
	0b00111111, // 0 - A, B, C, D, E, F
	0b00000110, // 1 - B, C
	0b01011011, // 2 - A, B, D, E, G
	0b01001111, // 3 - A, B, C, D, G
	0b01100110, // 4 - B, C, F, G
	0b01101101, // 5 - A, C, D, F, G
	0b01111101, // 6 - A, C, D, E, F, G
	0b00000111, // 7 - A, B, C
	0b01111111, // 8 - A, B, C, D, E, F, G
	0b01101111, // 9 - A, B, C, D, F, G
	0b01111100, // 10 - A, B, C, D, E
	0b01110001, // 11 - C, D, E, F
	0b01101001, // 12 - A, E, F, G
	0b00111001, // 13 - B, C, D
	0b01110101, // 14 - A, C, E, F
	0b01111101  // 15 - A, B, C, E, G
};

volatile uint8_t press_count = 0; // Количество нажатий кнопки

ISR(INT0_vect)
{
	// Обработчик прерывания для кнопки
	if (press_count < 15) {
		press_count++;
		} else {
		press_count = 0; // Сброс счетчика после 15 нажатий
	}
}

int main(void)
{
	DDRB = 0xFF;  // Порт B - выход (для первого индикатора)
	DDRC = 0xFF;  // Порт C - выход (для второго индикатора)
	DDRD |= (1 << PD0) | (1 << PD1); // Настройка PD0 и PD1 как выходы для сегментов
"G"
```

```
	PORTD |= (1<<PIND2); // Включаем внутренний подтягивающий резистор для INT0 (кнопка)

	// Настройка прерывания INT0
	EIMSK |= (1<<INT0);  // Включаем INT0
	EICRA |= (1<<ISC01); // Прерывание по спадающему фронту INT0
	sei();  // Глобальное разрешение прерываний

	while(1)
	{
		// Отображаем количество нажатий на индикаторах
		PORTB = segments[press_count / 10];  // Десятки на первом индикаторе (порт B)

		// Управление сегментом "G" на первом индикаторе (PD1)
		if (segments[press_count / 10] & 0b01000000) {
			PORTD |= (1 << PD1);  // Включаем сегмент "G" первого индикатора
			} else {
			PORTD &= ~(1 << PD1); // Выключаем сегмент "G" первого индикатора
		}

		PORTC = (segments[press_count % 10] & 0b00111111); // Выводим остальные сегменты на порт C

		// Управление сегментом "G" на втором индикаторе (PD0)
		if (segments[press_count % 10] & 0b01000000) {
			PORTD |= (1 << PD0);  // Включаем сегмент "G" второго индикатора
			} else {
			PORTD &= ~(1 << PD0); // Выключаем сегмент "G" второго индикатора
		}

		_delay_ms(200); // Задержка для обновления индикаторов
	}
}
```

Выводы

Изучил основные приемы работы с контроллерами AVR. Важными моментами здесь являются работа с портами, сдвиги и битовые операции, прерывания.