ЧОУ ВО «Тольяттинская академия управления»

Кафедра прикладной информатики и высшей математики

Б1. В.18

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА)

Учебная дисциплина	*Программирование микроконтроллеров*
По направлению подготовки	*09.03.03*
	«Прикладная информатика»
Профиль (программа бакалавриата)	*«Прикладная информатика в цифровой экономике»*
Форма обучения	*Очная*

Оценочные материалы (средства) дисциплины рассмотрены (актуализированы) и утверждены на заседании кафедры прикладной информатики и высшей математики

Протокол заседания № 10 от «25» мая 2026 г.

Заведующий кафедрой Стрекалова Наталья Борисовна

2026

Оглавление

1. ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ И ЭТАПЫ ИХ ФОРМИРОВАНИЯ В ПРОЦЕССЕ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Оценочные материалы (средства) сформированы в соответствии с ФГОС ВО - бакалавриат по направлению подготовки 09.03.03 «Прикладная информатика и высшая математика», утвержденный приказом Министерства образования и науки Российской Федерации от 19.09.2017 № 922 (с изменениями и дополнениями от 26.11.2020, 08.02.2021). В соответствии с матрицей компетенций основной профессиональной образовательной программы «Прикладная информатика в цифровой экономике» (уровень бакалавриата) в процессе обучения по дисциплине «Программирование микроконтроллеров» происходит формирование закрепленных за дисциплиной компетенций обучающихся. Оценка сформированности компетенций на каждом этапе обучения происходит через оценку планируемых результатов обучения по дисциплине (знаний, умений, навыков).

Результаты освоения образовательной программы (компетенции)	Индикаторы достижения компетенций	Планируемые результаты обучения по дисциплине	Этапы формирования компетенции (семестры и темы)	Оценочное средство
ПК-3. Способен разрабатывать информационные системы	ПК-3.1. Осуществляет кодирование на современных языках программирования, в том числе используя возможности специализированных цифровых платформ для индивидуальной и совместной разработки информационных систем	**Знать:** - общую архитектуру микроконтроллеров; типы и виды устройств микроконтроллеров; методологию (правила, команды, способы) разработки программного обеспечения для микроконтроллера семейства Arduino**.** **Уметь:** - программировать режимы работы устройств микроконтроллера; программировать обмен данными с помощью стандартных интерфейсов; работу с цифровыми датчиками; **Владеть:** - языками процедурного и объектно-ориентированного программирования, навыками разработки и отладки программ;	Тема 1. Введение в Arduino Тема 2. Обзор контроллеров семейства Arduino Тема 3. Среда разработки и язык программирования контроллеров Arduino. Тема 4. Цифровые выводы, широтно - импульсная модуляция, память в Arduino. Тема 5. Программирование в Arduino.	- устный опрос по темам 1-5 - ответ на теоретический вопрос зачета

		методами программирования устройств микроконтроллера		
	ПК-3.2. Проводит тестирование компонентов информационных систем, в том числе используя возможности специализированных цифровых платформ	**Знать:** - организацию программ для микроконтроллеров; возможности и особенности работы микроконтроллеров семейства Arduino. **Уметь:** - выполнять проверку и отладку программного кода для микроконтроллера; проверять корректность работы микроконтроллера; **Владеть:** - навыками разработки и отладки программ.	Тема 6. Практическое применение Arduino. Лабораторная работа № 1 «Плавная регулировка вращением электродвигателя» Лабораторная работа № 2 «Написание скетча «Часы» регулировка установки времени.» Лабораторная работа № 3 «Внесение изменений в скетч «Будильник» перенос режима времени срабатывания будильника и добавление режима включения отключения будильника.» Лабораторная работа № 4 «Составление принципиальной схемы на базе собранного проекта.» Самостоятельный проект «Составление принципиальной схемы, программирование микроконтроллера на базе собственного проекта.»	- практические навыки по программированию микроконтроллеров - лабораторные работы 1-5

2. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА) ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ И ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ, ОБЕСПЕЧИВАЮЩИХ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ В ПРОЦЕССЕ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Устные опросы

Вопросы для проведения устного опроса
по теме «Введение в Arduino»

1. Общая характеристика микроконтроллера Arduino.
2. В каких областях могут быть использованы микроконтроллеры семейства Arduino.
3. Основные компоненты платы Arduino Nano.
4. Характеристики контроллера Arduino Nano.
5. Типы входов и выходов Arduino Nano их общее назначение.
6. Основные преимущества платформы Arduino.
7. Проблема исключающего «или».
8. Многослойный персептрон. Представление булевых функций.
9. Преодоление ограничения линейной разделимости и решение проблемы исключающего «или».

Вопросы для проведения устного опроса

по теме «Обзор контроллеров семейства Arduino»

1. Понятие микроконтроллера. Основные разновидности микроконтроллеров.
2. Обзор основных производителей контроллеров.
3. Внутреннее устройство микроконтроллера, основные внутренние модули микроконтроллера, особенности микроконтроллеров семейства Arduino
4. Понятие прерываний, разновидности прерываний. Обработка прерываний разных типов контроллера Arduino
5. Типы внутренней памяти микроконтроллеров.
6. Режимы доступа и адресации контроллера Arduino
7. Типовые схемы подключения периферийных модулей микроконтроллеров.
8. Подключение периферии нагрузки, внешние интерфейсы контроллера Arduino
9. Понятие АЦП, разновидности АЦП Arduino.
10. Режимы работы АЦП контроллера Arduino

Вопросы для проведения устного опроса
по теме «Среда разработки и язык программирования контроллеров Arduino»

1. Для чего предназначена программно-аппаратная платформа Arduino.
2. Что входит в состав программной части платформы Arduino.
3. Что входит в состав аппаратной части платформы Arduino.
4. Опишите интерфейс среды разработки Arduino.
5. Опишите панель Меню.
6. Поясните порядок запуска готовых скетчей.
7. Что такое компиляция программы.
8. От чего зависит процесс прошивки программы в Arduino.
9. Какие циклы применяются при программировании для Arduino.
10. По какому протоколу происходит обмен данными между компьютером и микропроцессором.

Вопросы для проведения устного опроса
по теме «Цифровые выводы, широтно - импульсная модуляция, память в Arduino»

1. Какие интерфейсы доступны на платах Arduino. Опишите особенности интерфейсов SPI, I2C, UART и их применение.
2. Что такое ШИМ (широтно-импульсная модуляция) и как она используется в Arduino.
3. Для чего изменяется ширина импульса.
4. С помощью какой команды можно регулировать уровень ШИМ-
5. Для чего может быть использован ШИМ-сигнал.
6. В чем суть функции Serial.read().
7. В чем суть функции Serial.readBytes().
8. В чем суть функции Serial.parseInt().
9. В чем суть функции Serial.parseFloat().
10. Какие типы памяти есть у микроконтроллеров Arduino.
11. Опишите Flash-память, SRAM, EEPROM и их назначение.?

Вопросы для проведения устного опроса
по теме «Программирование в Arduino.»

1. В чем разница между функциями setup() и loop()? Какова роль каждой функции в жизненном цикле программы?
2. Что такое pinMode() и какие аргументы она принимает?
3. Для чего нужны режимы INPUT, OUTPUT и INPUT_PULLUP?

4. Чем отличаются функции delay() и millis()?Почему использование delay() считается плохой практикой в сложных проектах?
5. Каковы особенности типов данных int, float и boolean в Arduino?
6. Сколько байт памяти занимает тип int на плате Arduino Nano?
7. В чем разница между digitalWrite() и analogWrite()? Как именно analogWrite() имитирует аналоговый сигнал?
8. Как считать показания с аналогового датчика? Каков диапазон значений, возвращаемых функцией analogRead(), и почему?
9. Зачем нужен интерфейс UART (Serial) и как им пользоваться?
10. Для чего используются функции Serial.begin() и Serial.print()?
11. Как работают протоколы связи I2C и SPI?

Критерии оценивая устного опроса:

- оценка «*отлично*» выставляется студенту, если он ответил развернуто на один из заданных вопросов, активно дополнял ответы других студентов или задавал им дополнительные вопросы;
- оценка «*хорошо*» выставляется студенту, если он ответил развернуто на один из заданных вопросов или ответил кратко на ряд вопросов;
- оценка «*удовлетворительно*» выставляется студенту, если он кратко ответил на один из задаваемых вопросов или просто дополнял ответы других студентов, задавал им дополнительные вопросы;
- оценка «*неудовлетворительно*» выставляется студенту, если он не смог ответить правильно на заданный вопрос, либо не отвечал на вопросы и не участвовал в их обсуждении.

Практические работы

Практическая работа № 1

Плавная регулировка вращением электродвигателя

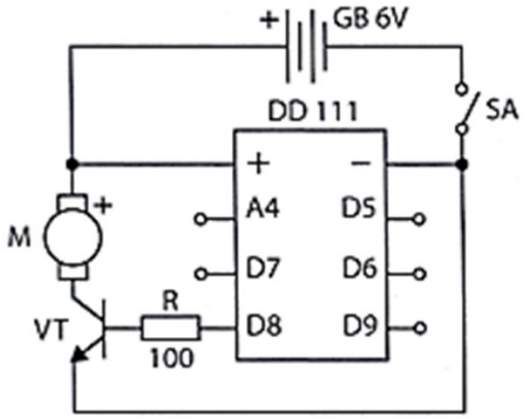
1.1 Цель роботы

Ознакомление с визуальной средой программирования IDE Arduino. Освоение методики создания скетчей и процедуры их настройки при помощи библиотек IDE Arduino. Приобретение навыков сборки практических схем.

1.2 Методические указания по самостоятельной работе студентов

План работы:

1. Сборка схемы рисунок 1. Проверка ее работоспособности. Максимальный ток, который может вырабатываться на выходах модуля 111, не превышает 40 мА, а для работы электродвигателя 38 требуется ток в несколько раз больше. Поэтому в качестве усилителя используется транзистор 52.



2. Применение среды программирования IDE Arduino, ее настройка под контроллер.
3. Повторение теоретического материала о программировании выводов ШИМ.
4. В разделе «Управление светодиодами – было изучено, как программными средствами организовать ШИМ на выводе D8. Самостоятельно написать скетч и научить данную схему плавно регулировать скорость вращения электродвигателя.
5. Проведение контрольных экспериментов. Каждый пункт должен быть отражён в отчёте.
6. В лабораторной работе отразить написанный скетч с комментариями.

Порядок выполнения работы

1. Собрать схему в соответствии с приведенным рисунком.
2. Загрузить скетч через IDE Arduino в микроконтроллер.
3. Убедиться в работоспособности и функционировании схемы, внести изменения в скетч в соответствии с заданием.
4. Сделать выводы

Содержание отчёта

Содержание отчёта должно соответствовать плану работы.
Обязательные пункты:

1) Название работы
2) Цель работы
3) Краткое описание исследованной схемы и скетча
4) Краткое описание внесенных изменений
5) Листинг скетча с подробными комментариями
6) Расширенные выводы из лабораторной работы

Критерии оценивая:

- оценка «*отлично*» выставляется студенту, если задание выполнено в полном объеме, студент хорошо ориентируется в схемотехнике и написании скетча, скетч логично и корректно демонстрирует работу всех функций;
- оценка «*хорошо*» выставляется студенту, если он выполнил задание почти в полном объеме (может отсутствовать пункт), допускает небольшие «проектно-синтаксические» ошибки в написании скетча;
- оценка «*удовлетворительно*» выставляется студенту, если он выполнил написание скетча, но не составил отчет;
- оценка «*неудовлетворительно*» выставляется студенту, если он не выполнил написание скетча по заданию, и не составил отчет.

Практическая работа № 2

Написание скетча «Часы» регулировка установки времени.

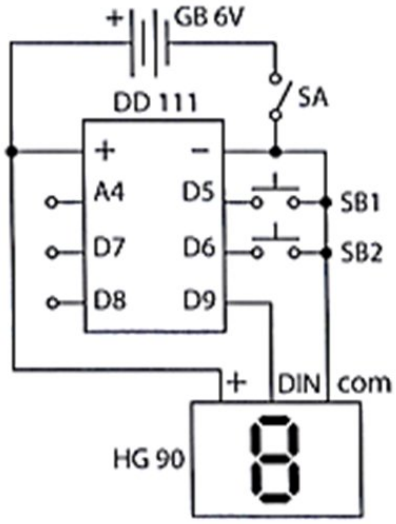
1.1 Цель роботы

Ознакомление с визуальной средой программирования IDE Arduino. Освоение методики создания скетчей и процедуры их настройки при помощи библиотек IDE Arduino. Приобретение навыков сборки практических схем.

1.2 Методические указания по самостоятельной работе студентов

План работы:

1. Сборка схемы рисунок 2. Проверка ее работоспособности.



2. В приведенном ниже скетче реализован алгоритм – короткое нажатие на левую кнопку позволяет настраивать часы (к отображаемому времени прибавляется один час), короткое нажатие на правую кнопку позволяет настроить минуты (к отображаемому времени прибавляется одна минута). После установки времени часы автоматически продолжают работу. В скетче используется 24 – часовой формат. Написать скетч на основе представленного, который осуществляет более сложный алгоритм: при длительном нажатии левой кнопки схема переходит в режим установки часов – левая кнопка уменьшает значение, правая кнопка увеличивает. Повторное длительное нажатие на левую кнопку переводит схему в рабочий режим. Аналогично должна работать и правая кнопка, которая должна переводить схему в режим установки минут, затем правой и левой кнопками увеличивать и уменьшать показания на дисплее.

```
#include "src/DigitalIO.h"

DigitalPin<9> pin9;

int buf = 0;
void setup() {
  Serial.begin(9600);
  pin9.config(OUTPUT, LOW);

  pinMode(5, INPUT_PULLUP);
  pinMode(6, INPUT_PULLUP);


  for (int i = 0; i < 64; i++)
  {
    bitZero();
    delayMicroseconds(6);
  }
}
```

```
void bitZero()
{
 noInterrupts();
 pin9.toggle();
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );


 pin9.toggle();
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 interrupts();
}

void bitOne()
{
 noInterrupts();
 pin9.toggle();
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );

 pin9.toggle();
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 asm volatile ( "nop":: );
 interrupts();
}


#define SEGMENT_COUNT       8
byte numberSegments[11] = {
 0b00111111,//0
 0b00000110,
 0b01011011,
 0b01001111,
 0b01100110,
 0b01101101,
 0b01111101,
 0b00000111,
 0b01111111,
 0b01101111,
 0b00000000
```

```
};
#define STR_ERROR_SYMBOL 0
#define STR_MA_SYMBOL 1
#define STR_KELVIN_SYMBOL 2
#define STR_OHM_SYMBOL 3
#define STR_VOLTAGE_SYMBOL 4
#define STR_LUX_SYMBOL 5
#define STR_DB_SYMBOL 6

#define TEMP_PIN 7

void printSymbolString(int k)
{

  for (int j = 0; j < 4; j++)
  {
    for (int i = 0; i < 8; i++)
    {
      if (i == k)
      {
        bitOne();
      }
      else
      {
        bitZero();
      }
      delayMicroseconds(6);
    }
  }
}

void printNum(int num)
{

  for (int i = 0; i < SEGMENT_COUNT; i++) {
    boolean enableSegment = bitRead(numberSegments[num], i);
    if (enableSegment == 1)
    {
      bitOne();
      delayMicroseconds(6);
    }
    else
    {
      bitZero();
      delayMicroseconds(6);
    }
  }
  delayMicroseconds(10);
}

void printNumW(int num)
{

  for (int i = 0; i < SEGMENT_COUNT; i++) {
    boolean enableSegment = bitRead(numberSegments[num], i);
    if (enableSegment == 1 || i == 7)
    {
```

```
      bitOne();
      delayMicroseconds(6);
    }
    else
    {
      bitZero();
      delayMicroseconds(6);
    }
  }
  delayMicroseconds(10);
}

void printZero()
{
  for (int i = 0; i < 6; i++)
  {
    bitOne();
    delayMicroseconds(6);
  }
  bitZero();
  delayMicroseconds(6);
  bitZero();
  delayMicroseconds(6);
}


void displayTimeSeconds(int num)
{
  int a, b, c, d;
  d = num % 10;
  num = num / 10;
  c = num % 10;
  num = num / 10;
  b = num % 10;
  num = num / 10;
  a = num;

  printNum(a);
  printNum(b);
  printNumW(c);
  printNum(d);
  //delay(100);
}

void displayClockMinutesSeconds(long int ms)
{
  Serial.print("ms = "); Serial.println(ms);
  int ms_in_seconds = ms / 1000;
  int clock_seconds = ms_in_seconds % 60;
  int c = clock_seconds / 10;
  int d = clock_seconds % 10;

  int clock_minutes = ms_in_seconds / 60;
  int a = clock_minutes / 10;
  int b = clock_minutes % 10;

  Serial.print(a);
```

```
  Serial.print(b);
  Serial.print(c);
  Serial.println(d);

  printNum(a);

  if (ms % 1000 > 500)
  {
    printNum(b);
  }
  else
  {
    printNumW(b);
  }
  printNum(c);
  printNum(d);

}

long set_minutes = 0;
long set_hours = 0;
void displayClockHoursMinutes(long int ms)
{
  int ms_in_seconds = ms / 1000;
  int clock_seconds = ms_in_seconds % 60;

  int clock_minutes = ms_in_seconds / 60;


  int clock_hours = clock_minutes / 60;

  int c = ((set_minutes +clock_minutes) % 60) / 10 ;
  int d = ((set_minutes +clock_minutes) % 60) % 10 ;

  int a = ((set_hours + clock_hours) % 60) / 10;
  int b = ((set_hours + clock_hours) % 60) % 10;


  printNum(a);

  if (ms % 1000 > 500)
  {
    printNum(b);
  }
  else
  {
    printNumW(b);
  }
  printNum(c);
  printNum(d);

}

void displayNumber(int num)
{

  int a, b, c, d;
```

```
  d = num % 10;
  num /= 10;
  c = num % 10;
  b = num / 10;
  a = 10;

  printNum(a);
  printNum(b);
  printNum(c);
  printNum(d);
  //printSymbolString(STR_VOLTAGE_SYMBOL);
}


//int cur_time = 0;
int counter = 0;
int prev_counter_value = 180;

boolean isCounting = false;
long cur_time = 0;
long prev_time = 0;
long stop_time = 0;
long start_time = 0;
long set_time = 0;

void loop()
{
  int k = 64;

  if (digitalRead(6) == LOW)
  {
    delay(50);
    if (digitalRead(6) == LOW)
    {
      set_hours += 1;
      set_hours %= 24;
    }
    delay(50);
  }

  if (digitalRead(5) == LOW)
  {
    delay(50);
    if (digitalRead(5) == LOW)
    {
       set_minutes += 1;
       set_hours += (set_minutes / 60);
       set_hours %= 24;
       set_minutes %= 60;
    }
  }

  displayClockHoursMinutes(millis() + set_time);
  delay(50);
}
```

3. Проведение контрольных экспериментов. Каждый пункт должен быть отражён в отчёте.
4. Повторение теоретического материала о программировании микроконтроллера Arduino Nano, составления практических схем, элементы схемотехники.
5. В лабораторной работе отразить итоговый скетч с подробными комментариями.
Каждый пункт должен быть отражён в отчёте.

Порядок выполнения работы

1. Собрать схему в соответствии с приведенным рисунком.
2. Загрузить скетч через IDE Arduino в микроконтроллер.
3. Убедиться в работоспособности и функционировании схемы, внести изменения в скетч в соответствии с заданием.
4. Сделать выводы

Содержание отчёта

Содержание отчёта должно соответствовать плану работы.
Обязательные пункты:
1) Название работы
2) Цель работы
3) Краткое описание исследованной схемы и скетча
4) Краткое описание внесенных изменений
5) Листинг скетча с подробными комментариями
6) Расширенные выводы из лабораторной работы

Критерии оценивая:

- оценка «*отлично*» выставляется студенту, если задание выполнено в полном объеме, студент хорошо ориентируется в схемотехнике и написании скетча, скетч логично и корректно демонстрирует работу всех функций;
- оценка «*хорошо*» выставляется студенту, если он выполнил задание почти в полном объеме (может отсутствовать пункт), допускает небольшие «проектно-синтаксические» ошибки в написании скетча;
- оценка «*удовлетворительно*» выставляется студенту, если он выполнил написание скетча, но не составил отчет;
- оценка «*неудовлетворительно*» выставляется студенту, если он не выполнил написание скетча по заданию, и не составил отчет.

Практическая работа № 3

Внесение изменений в скетч «Будильник» перенос режима времени срабатывания будильника и добавление режима включения отключения будильника.

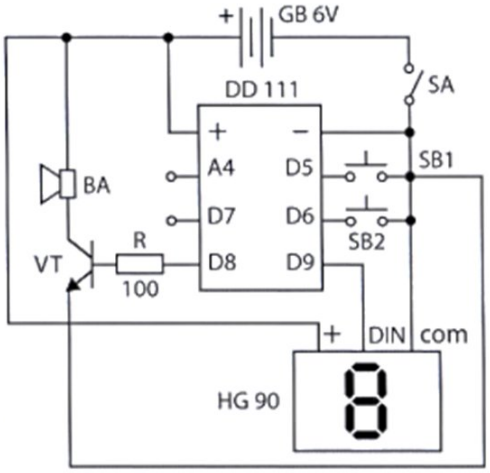
1.1 Цель роботы

Ознакомление с визуальной средой программирования IDE Arduino. Освоение методики создания скетчей и процедуры их настройки при помощи библиотек IDE Arduino. Приобретение навыков сборки практических схем.

1.2 Методические указания по самостоятельной работе студентов

План работы:

1 Сборка схемы рисунок 3. Проверка ее работоспособности



2 Описание работы контрольного скетча

Установка текущего времени (левая кнопка)

- Нажмите и удерживайте (не мене 2 сек.) левую кнопку – схема перейдет в режим установки часов текущего времени – начнет моргать соответ-ствующая секция на индикаторе 90.
- Короткими нажатиями на эту же кнопку устанавливается текущий час.
- Нажмите и удерживайте (не менее 2 сек.) левую кнопку – схема перейдет в режим установки минут - начнет моргать соответствующая секция на индикаторе 90.
- Короткими нажатиями на эту же кнопку устанавливается необходимое значение.
- Нажмите и удерживайте (не менее 2 сек.) левую кнопку – схема перейдет в режим текущего времени.

Установка будильника (правая кнопка)

- Нажмите и удерживайте (не мене 2 сек.) правую кнопку – схема перейдет в режим установки часов будильника – начнет моргать соответствующая секция на индикаторе 90.
- Короткими нажатиями на эту же кнопку устанавливается час пробужде-ния.
- Нажмите и удерживайте (не мене 2 сек.) правую кнопку – схема перейдет в режим установки минут будильника – начнет моргать соответствующая секция на индикаторе 90.
- Короткими нажатиями на эту же кнопку устанавливается необходимое значение.
- Нажмите и удерживайте (не менее 2 сек.) левую кнопку – схема перейдет в режим текущего времени.

По достижении заданных значений будильник сработает и раздастся звуковой сигнал. Что бы выключить будильник необходимо нажать на правую кнопку.

```
#include "src/DigitalIO.h"
DigitalPin<9> pin9;


int buf = 0;
void setup()
{

  pin9.config(OUTPUT, LOW);

  pinMode(5, INPUT_PULLUP);
  pinMode(6, INPUT_PULLUP);
  for (int i = 0; i < 64; i++)
  {
    bitZero();
    delayMicroseconds(6);
  }
}
```

```
void bitZero()
{
  noInterrupts();
  pin9.toggle();
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );


  pin9.toggle();
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  interrupts();
}

void bitOne()
{
  noInterrupts();
  pin9.toggle();
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );

  pin9.toggle();
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  asm volatile ( "nop":: );
  interrupts();
}


#define SEGMENT_COUNT       8
byte numberSegments[11] = {
  0b00111111,//0
  0b00000110,
  0b01011011,
  0b01001111,
  0b01100110,
  0b01101101,
  0b01111101,
  0b00000111,
  0b01111111,
  0b01101111,
```

```
  0b00000000
};
#define STR_ERROR_SYMBOL 0
#define STR_MA_SYMBOL 1
#define STR_KELVIN_SYMBOL 2
#define STR_OHM_SYMBOL 3
#define STR_VOLTAGE_SYMBOL 4
#define STR_LUX_SYMBOL 5
#define STR_DB_SYMBOL 6

#define TEMP_PIN 7

void printSymbolString(int k)
{

  for (int j = 0; j < 4; j++)
  {
    for (int i = 0; i < 8; i++)
    {
      if (i == k)
      {
        bitOne();
      }
      else
      {
        bitZero();
      }
      delayMicroseconds(6);
    }
  }
}

void printNum(int num)
{

  for (int i = 0; i < SEGMENT_COUNT; i++) {
    boolean enableSegment = bitRead(numberSegments[num], i);
    if (enableSegment == 1)
    {
      bitOne();
      delayMicroseconds(6);
    }
    else
    {
      bitZero();
      delayMicroseconds(6);
    }
  }
  delayMicroseconds(10);
}

void printNumW(int num)
{

  for (int i = 0; i < SEGMENT_COUNT; i++) {
    boolean enableSegment = bitRead(numberSegments[num], i);
    if (enableSegment == 1 || i == 7)
```

```
    {
      bitOne();
      delayMicroseconds(6);
    }
    else
    {
      bitZero();
      delayMicroseconds(6);
    }
  }
  delayMicroseconds(10);
}

void printZero()
{
  for (int i = 0; i < 6; i++)
  {
    bitOne();
    delayMicroseconds(6);
  }
  bitZero();
  delayMicroseconds(6);
  bitZero();
  delayMicroseconds(6);
}


void displayTimeSeconds(int num)
{
  int a, b, c, d;
  Serial.println(num);
  d = num % 10;
  num = num / 10;
  c = num % 10;
  num = num / 10;
  b = num % 10;
  num = num / 10;
  a = num;
  Serial.print(a);
  Serial.print(b);
  Serial.print(c);
  Serial.println(d);
  printNum(a);
  printNum(b);
  printNumW(c);
  printNum(d);
  //delay(100);
}

#define debounce 20
#define holdTime 1000

int l_buttonPin = 6;
int l_buttonVal = 0;
int l_buttonLast = 0;
long l_btnDnTime;
long l_btnUpTime;
```

```
boolean l_ignoreUp = false;

int r_buttonPin = 5;
int r_buttonVal = 0;
int r_buttonLast = 0;
long r_btnDnTime;
long r_btnUpTime;
boolean r_ignoreUp = false;

boolean workTimeMode = true;
boolean isAlarmRinging = false;
boolean setupTimeMode = false;
boolean setupAlarmMode = false;

boolean hoursMode = false;

boolean hoursIsBlinking = false;
boolean dotsIsBlinking = true;
boolean minutesIsBlinking = false;

int cur_time_hours = 12;
int cur_time_minutes = 34;

int cur_alarm_hours = 12;
int cur_alarm_minutes = 36;

long int prev_hit_time = 0;

long int diff = 0;

void displayClockHoursMinutes(long int hours, long int minutes)
{
 hours = hours % 24;
 minutes = minutes % 60;

 int c = minutes / 10 ;
 int d = minutes % 10 ;

 int a = hours / 10;
 int b = hours % 10;


 if (hoursIsBlinking && millis() % 1000 > 500)
 {
  printNum(10);
  printNumW(10);
 }
 else
 {
  printNum(a);
 if (dotsIsBlinking && millis() % 1000 > 500)
 {
  printNum(b);
 }
 else
 {
  printNumW(b);
```

```
  }
  }


  //
  if (minutesIsBlinking && millis() % 1000 > 500)
  {
    printNum(10);
    printNum(10);
  }
  else
  {
    printNum(c);
    printNum(d);
  }
}

void incCurTimeHour()
{
  cur_time_hours++;
  if (cur_time_hours >= 24)
  {
    cur_time_hours = 0;
  }
}

void incCurAlarmHour()
{
  cur_alarm_hours++;
  if (cur_alarm_hours >= 24)
  {
    cur_alarm_hours = 0;
  }
}

void incCurTimeMinute()
{
  cur_time_minutes++;
  if (cur_time_minutes >= 60)
  {
    cur_time_minutes = 0;
    incCurTimeHour();
  }
}

void incCurAlarmMinute()
{
  cur_alarm_minutes++;
  if (cur_alarm_minutes >= 60)
  {
    cur_alarm_minutes = 0;
    incCurAlarmHour();
  }
}

void loop()
{
```

```
  diff = millis() - prev_hit_time;
  if (diff > 60000)
  {
    prev_hit_time = millis();
    for (int i = 0; i < diff / 60000; i++)
      incCurTimeMinute();
  }


  l_buttonVal = digitalRead(l_buttonPin);
  r_buttonVal = digitalRead(r_buttonPin);


  if (l_buttonVal == LOW && l_buttonLast == HIGH && (millis() - l_btnUpTime) >
long(debounce))
  {
    l_btnDnTime = millis();
  }
  if (r_buttonVal == LOW && r_buttonLast == HIGH && (millis() - r_btnUpTime) >
long(debounce))
  {
    r_btnDnTime = millis();
  }

  if (l_buttonVal == HIGH && l_buttonLast == LOW && (millis() - l_btnDnTime) >
long(debounce))
  {
    if (l_ignoreUp == false)
    {
      //short
      l_event1();
    }
    else
    {
      l_ignoreUp = false;
    }
    l_btnUpTime = millis();
  }
  if (r_buttonVal == HIGH && r_buttonLast == LOW && (millis() - r_btnDnTime) >
long(debounce))
  {
    if (r_ignoreUp == false)
    {
      //short
      r_event1();
    }
    else
    {
      r_ignoreUp = false;
    }
    r_btnUpTime = millis();
  }


  if (l_buttonVal == LOW && (millis() - l_btnDnTime) > long(holdTime))
  {
    //long
```

```
    l_event2();
    l_ignoreUp = true;
    l_btnDnTime = millis();
   }
   if (r_buttonVal == LOW && (millis() - r_btnDnTime) > long(holdTime))
   {
    //long
    r_event2();
    r_ignoreUp = true;
    r_btnDnTime = millis();
   }

   l_buttonLast = l_buttonVal;
   r_buttonLast = r_buttonVal;

   if (cur_time_hours == cur_alarm_hours && cur_time_minutes == cur_alarm_minutes &&
workTimeMode && diff < 2000)
   {
    isAlarmRinging = true;

    hoursIsBlinking = true;
    dotsIsBlinking = true;
    minutesIsBlinking = true;
   }

   if (isAlarmRinging)
   {
    alarm();
   }


   if (!setupAlarmMode)
    displayClockHoursMinutes(cur_time_hours, cur_time_minutes);
   else
    displayClockHoursMinutes(cur_alarm_hours, cur_alarm_minutes);

   delay(100);
}

void l_event1()
{
   if (setupTimeMode == true)
   {
    if (hoursMode)
      incCurTimeHour();
    else
      incCurTimeMinute();
   }
}

void l_event2()
{
   if (workTimeMode == true)
   {
    workTimeMode = false;
    setupTimeMode = true;
    hoursMode = true;
```

```
    hoursIsBlinking = true;
    dotsIsBlinking = false;
    minutesIsBlinking = false;

  } else if (setupTimeMode)
  {
    if (hoursMode)
    {
      hoursMode = false;

      hoursIsBlinking = false;
      dotsIsBlinking = false;
      minutesIsBlinking = true;
    } else
    {
      setupTimeMode = false;
      hoursMode = false;
      workTimeMode = true;

      hoursIsBlinking = false;
      dotsIsBlinking = true;
      minutesIsBlinking = false;
    }
  }
}


void r_event1()
{
  if (setupAlarmMode == true)
  {
    if (hoursMode)
      incCurAlarmHour();
    else
      incCurAlarmMinute();
  } else if (isAlarmRinging)
  {
    isAlarmRinging = false;
    hoursIsBlinking = false;
    dotsIsBlinking = true;
    minutesIsBlinking = false;
  }
}

void r_event2()
{
  if (workTimeMode == true)
  {
    workTimeMode = false;
    setupAlarmMode = true;
    hoursMode = true;

    hoursIsBlinking = true;
    dotsIsBlinking = false;
    minutesIsBlinking = false;
  } else if (setupAlarmMode)
```

```
  {
   if (hoursMode)
   {
    hoursMode = false;

    hoursIsBlinking = false;
    dotsIsBlinking = false;
    minutesIsBlinking = true;
   } else
   {
    setupAlarmMode = false;
    hoursMode = false;
    workTimeMode = true;

    hoursIsBlinking = false;
    dotsIsBlinking = true;
    minutesIsBlinking = false;
   }
  }

}


void alarm()
{
 pinMode(8, OUTPUT);

 tone(8, 440, 150);
 delay(150);
 tone(8, 660, 150);
 delay(150);
}
```

3 Добавьте режим переноса времени срабатывания будильника (функция SNOOZE) – во время срабатывания будильника при нажатии на левую кнопку будильник выключается, но через минуту снова срабатывает. Затем добавьте возможность включения/отключения будильника, не выходя из режима текущего времени, путем краткосрочного нажатия на правую кнопку, о чем должен сигнализировать символ «Перегрузка» на дисплее.

4 Проведение контрольных экспериментов. Каждый пункт должен быть отражён в отчёте.

5 Повторение теоретического материала о программировании микроконтроллера Arduino Nano, составления практических схем, элементы схемотехники.

6 В лабораторной работе отразить итоговый скетч с подробными комментариями.

Каждый пункт должен быть отражён в отчёте.

Порядок выполнения работы

1. Собрать схему в соответствии с приведенным рисунком.
2. Загрузить скетч через IDE Arduino в микроконтроллер.
3. Убедиться в работоспособности и функционировании схемы, внести изменения в скетч в соответствии с заданием.
4. Сделать выводы

Содержание отчёта

Содержание отчёта должно соответствовать плану работы.
Обязательные пункты:

1) Название работы
2) Цель работы
3) Краткое описание исследованной схемы и скетча
4) Краткое описание внесенных изменений
5) Листинг скетча с подробными комментариями
6) Расширенные выводы из лабораторной работы

Критерии оценивая:

- оценка «*отлично*» выставляется студенту, если задание выполнено в полном объеме, студент хорошо ориентируется в схемотехнике и написании скетча, скетч логично и корректно демонстрирует работу всех функций;
- оценка «*хорошо*» выставляется студенту, если он выполнил задание почти в полном объеме (может отсутствовать пункт), допускает небольшие «проектно-синтаксические» ошибки в написании скетча;
- оценка «*удовлетворительно*» выставляется студенту, если он выполнил написание скетча, но не составил отчет;
- оценка «*неудовлетворительно*» выставляется студенту, если он не выполнил написание скетча по заданию, и не составил отчет.

Практическая работа № 4

Составление принципиальной схемы на базе собранного проекта

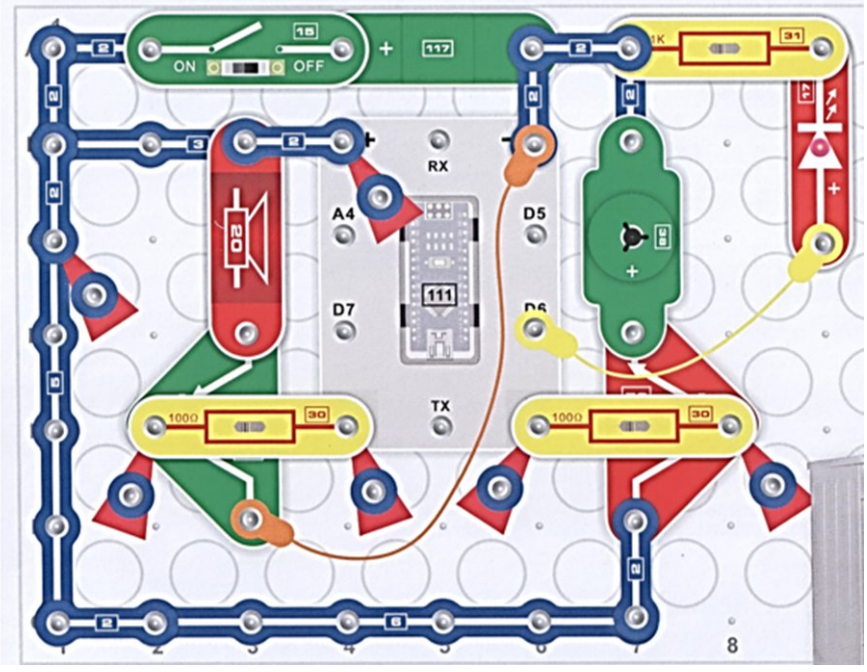
1.1 Цель работы

Ознакомление с составлением и правилами схемотизирования принципиальных схем. Приобретение практических навыков составления принципиальных схем.

1.2 Методические указания по самостоятельной работе студентов

План работы:

1. На основании схемы рисунок 4 составить и нарисовать принципиальную схему.



2. Краткие правила рисования принципиальных схем:

- Логика слева направо: Располагайте входные интерфейсы слева, а выходные — справа.
- Поток питания: Линии питания (VCC) всегда направляйте вверх, а «землю» (GND) — вниз.
- Стандартные УГО: Используйте только общепринятые графические обозначения компонентов (ГОСТ или ANSI/IEEE).
- Позиционные обозначения: Каждому компоненту присвойте уникальный десигнатор (например, R1, C10) и укажите его номинал.
- Читаемость связей: Минимизируйте количество пересечений линий; вместо длинных запутанных линий используйте именованные метки сетей (Net Labels).
- Точки соединений: Обязательно ставьте точку (junction) в местах электрического контакта пересекающихся линий.
- Группировка: Размещайте функционально связанные компоненты (например, обвязку микросхемы) рядом друг с другом.
- Текстовая ориентация: Располагайте все надписи горизонтально или вертикально (с поворотом на 90°) для удобства чтения.
- Соблюдение шага сетки: Всегда рисуйте по сетке, чтобы выводы компонентов точно совпадали с линиями связи.

3 Проведение контрольных экспериментов. Каждый пункт должен быть отражён в отчёте.

4 Повторение теоретического материала о программировании микроконтроллера Arduino Nano, составления практических схем, элементы схемотехники.

5 В лабораторной работе отразить итоговый схему с подробными комментариями.

Каждый пункт должен быть отражён в отчёте.

Порядок выполнения работы

1. Собрать схему и нарисовать в соответствии с приведенным рисунком.
2. Загрузить пробный скетч через IDE Arduino в микроконтроллер.
3. Убедиться в работоспособности и функционировании схемы, внести изменения в скетч в соответствии с заданием.
4. Сделать выводы

Содержание отчёта

Содержание отчёта должно соответствовать плану работы.
Обязательные пункты:

1) Название работы
2) Цель работы
3) Краткое описание исследованной схемы и ее функционирования
4) Краткое описание внесенных изменений
5) Листинг скетча с подробными комментариями
6) Расширенные выводы из лабораторной работы

Критерии оценивая:

- оценка «*отлично*» выставляется студенту, если задание выполнено в полном объеме, студент хорошо ориентируется в схемотехнике составлении схем и написании скетча, скетч логично и корректно демонстрирует работу всех функций;
- оценка «*хорошо*» выставляется студенту, если он выполнил задание почти в полном объеме (может отсутствовать пункт), допускает небольшие «проектно-синтаксические» ошибки в написании скетча и в алгоритме составления схемы;

- оценка «*удовлетворительно*» выставляется студенту, если он выполнил написание скетча и корректно начертил схему, но не составил отчет;
- оценка «*неудовлетворительно*» выставляется студенту, если он не выполнил написание скетча, не начертил схему по заданию, и не составил отчет.

Самостоятельный проект (является альтернативой выполнению лабораторных работ)

Разработка и составление принципиальной схемы, программирование микроконтроллера на базе собственного разработанного проекта.

1.1 Цель роботы

Самостоятельно создать устройство и запрограммировать его на основе выбранного контроллера. Нарисовать принципиальную схему с описанием всех компонентов. Составить программу и ее описание, привести листинг с подробными комментариями.
Самостоятельный проект, является альтернативой выполнению лабораторных работ и учитывается в полном объеме.

1.2 Методические указания по самостоятельной проекту студентов

2. Краткие правила рисования принципиальных схем:
- Логика слева направо: Располагайте входные интерфейсы слева, а выходные — справа.
- Поток питания: Линии питания (VCC) всегда направляйте вверх, а «землю» (GND) — вниз.
- Стандартные УГО: Используйте только общепринятые графические обозначения компонентов (ГОСТ или ANSI/IEEE).
- Позиционные обозначения: Каждому компоненту присвойте уникальный десигнатор (например, R1, C10) и укажите его номинал.
- Читаемость связей: Минимизируйте количество пересечений линий; вместо длинных запутанных линий используйте именованные метки сетей (Net Labels).
- Точки соединений: Обязательно ставьте точку (junction) в местах электрического контакта пересекающихся линий.
- Группировка: Размещайте функционально связанные компоненты (например, обвязку микросхемы) рядом друг с другом.
- Текстовая ориентация: Располагайте все надписи горизонтально или вертикально (с поворотом на 90°) для удобства чтения.
- Соблюдение шага сетки: Всегда рисуйте по сетке, чтобы выводы компонентов точно совпадали с линиями связи.

3 Проведение контрольных экспериментов. Каждый пункт должен быть отражён в отчёте.
4 Повторение теоретического материала о программировании микроконтроллера Arduino Nano, составления практических схем, элементы схемотехники.
5 В лабораторной работе отразить итоговый схему с подробными комментариями.
Каждый пункт должен быть отражён в отчёте.

Порядок выполнения работы

1. Собрать схему и начертить в соответствии с выбранным проектом.
2. Загрузить скетч через IDE Arduino в микроконтроллер.

3. Убедиться в работоспособности и функционировании схемы, внести изменения в скетч при обнаружении сбоев.
4. Сделать выводы

Содержание отчёта

Содержание отчёта должно соответствовать плану работы.
Обязательные пункты:

1) Название работы
2) Цель работы
3) Краткое описание разработанной схемы и ее функционирования
4) Краткое описание внесенных изменений
5) Листинг скетча с подробными комментариями, схема проекта
6) Расширенные выводы из лабораторной работы

Критерии оценивая:

- оценка «*отлично*» выставляется студенту, если задание выполнено в полном объеме, студент хорошо ориентируется в схемотехнике составлении схем и написании скетча, скетч логично и корректно демонстрирует работу всех функций;
- оценка «*хорошо*» выставляется студенту, если он выполнил задание почти в полном объеме (может отсутствовать пункт), допускает небольшие «проектно-синтаксические» ошибки в написании скетча и в алгоритме составления схемы;
- оценка «*удовлетворительно*» выставляется студенту, если он выполнил написание скетча и корректно начертил схему, но не составил отчет;
- оценка «*неудовлетворительно*» выставляется студенту, если он не выполнил написание скетча, не начертил схему по заданию, и не составил отчет.

3. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА) ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ

ОЦЕНКА ЗНАНИЙ ОБУЧАЮЩИХСЯ НА ЗАЧЕТЕ

ВОПРОСЫ ДЛЯ ПОДГОТОВКИ К ЗАЧЕТУ

Компетенция ПК-3. Способен разрабатывать информационные системы.

Индикатор ПК-3.1. Осуществляет кодирование на современных языках программирования, в том числе используя возможности специализированных цифровых платформ для индивидуальной и совместной разработки информационных систем

Обучающийся знает: общую архитектуру микроконтроллеров; типы и виды устройств микроконтроллеров; методологию (правила, команды, способы) разработки программного обеспечения для микроконтроллера семейства Arduino

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам его участия в устных опросах, а также по результатам ответа на зачете:

1. **Гарвардская и фон-Неймановская архитектуры**
   В чем их главное различие при работе с памятью? Какая из них используется в AVR (Arduino)?
2. **Структурные блоки микроконтроллера**
   Назовите основные узлы типичного МК. За что отвечают АЛУ, Flash, SRAM и EEPROM?
3. **Регистры общего назначения и ввода-вывода**

Какова роль регистров в управлении периферией? Чем отличаются регистры направления порта от регистров данных?

4. **Классификация по разрядности**
   В чем разница между 8-битными, 16-битными и 32-битными МК? Как разрядность влияет на точность вычислений и объем памяти?
5. **Микроконтроллер vs Микропроцессор**
   Сформулируйте главное отличие МК от центрального процессора (ЦПУ). В каких задачах целесообразно применять каждое из устройств?
7. **Жизненный цикл программы в Arduino**
   Опишите назначение и порядок выполнения функций setup() и loop(). Что происходит с МК после завершения кода в setup()?
8. **Управление портами ввода-вывода**
   Каковы правила конфигурирования пинов с помощью pinMode()? Объясните разницу между режимами INPUT и INPUT_PULLUP.
9. **Аналоговый ввод и вывод**
   Как работает функция analogRead() и что показывает возвращаемое значение? Каким образом функция analogWrite() имитирует аналоговый сигнал (ШИМ)?
10. **Работа со временем и задержками**
    Чем опасен блокирующий подход с использованием функции delay()? Опишите методологию построения неблокирующего кода с помощью millis().
11. **Прерывания (Interrupts)**
    Что такое внешнее прерывание и для каких задач оно необходимо? Какие ограничения накладываются на код внутри функции обработки прерывания (ISR)?
12. **Интерфейсы передачи данных**
    Перечислите базовые интерфейсы, доступные на платах Arduino (UART, SPI, I2C). Какой из них требует наименьшее количество проводов для подключения нескольких датчиков?

Компетенция ПК-3. Способен разрабатывать информационные системы.
Индикатор ПК-3.2. Проводит тестирование компонентов информационных систем, в том числе используя возможности специализированных цифровых платформ
Обучающийся знает: организацию программ для микроконтроллеров; возможности и особенности работы микроконтроллеров семейства Arduino.
Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам его участия в устных опросах, а также по результатам ответа на зачете.

1. **Архитектура программного обеспечения МК**
   В каких случаях классической «суперпетли» (super-loop) становится недостаточно?
2. **Проектирование на основе конечных автоматов (FSM)**
   Как концепция конечных автоматов помогает избавиться от сложных вложенных условий if-else? С помощью каких языковых конструкций C/C++ (например, switch-case, enum) реализуется автомат?
3. **Многозадачность в одноядерном МК**
   Как организовать псевдопараллельное выполнение нескольких задач на плате Arduino? Опишите диспетчеризацию задач по таймеру без использования операционной системы.
4. **Оптимизация использования оперативной памяти (SRAM)**
   К каким последствиям приводит фрагментация памяти при частом использовании динамических строк String? Зачем нужен макрос F() при выводе текстовых строк в Serial-порт?
5. **Глобальные, локальные и статические переменные**

Как область видимости и время жизни переменных влияют на стабильность работы прошивки? В чем специфика применения модификатора static внутри функций микроконтроллера?

6. **Аппаратные таймеры и их использование**
   Какие стандартные функции Arduino перестанут работать, если программист переконфигурирует Таймер 0?
7. **Прямое управление регистрами (Direct Port Manipulation)**
   В чем главные недостатки стандартных функций digitalWrite() и digitalRead()? Как и зачем осуществлять запись напрямую в регистры портов (например, PORTB, DDRB)?
8. **Энергосбережение и режимы сна (Sleep Modes)**
   Какие встроенные режимы пониженного энергопотребления поддерживает микроконтроллер Arduino? Каким образом и какими событиями можно «разбудить» МК из глубокого сна?
9. **Особенности работы со встроенной энергонезависимой памятью (EEPROM)**
   Для каких задач используется EEPROM в Arduino и каковы главные физические ограничения этой памяти? Чем функция EEPROM.update() принципиально лучше, чем EEPROM.write()?
10. **Аппаратный сторожевой таймер (Watchdog Timer)**
    Какова роль Watchdog-таймера в обеспечении отказоустойчивости автономных систем?Что произойдет с микроконтроллером, если программа «зависнет» и не успеет сбросить сторожевой таймер?
11. **Специфика работы с волатильными данными в прерываниях**
    Зачем переменные, изменяемые внутри обработчика прерываний (ISR), должны объявляться с ключевым словом volatile? Что такое атомарный доступ к данным и почему он важен для 8-битных МК при работе с двухбайтовыми переменными?

ОЦЕНКА УМЕНИЙ И НАВЫКОВ ОБУЧАЮЩИХСЯ НА ЗАЧЕТЕ

Компетенция ПК-3 Способен разрабатывать информационные системы
Индикатор ПК-3.1. Осуществляет кодирование на современных языках программирования, в том числе используя возможности специализированных цифровых платформ для индивидуальной и совместной разработки информационных систем
Обучающийся умеет: программировать режимы работы устройств микроконтроллера; программировать обмен данными с помощью стандартных интерфейсов; работу с цифровыми датчиками;
Обучающийся владеет: языками процедурного и объектно- ориентированного программирования, навыками разработки и отладки программ; методами программирования устройств микроконтроллера
Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам выполнения практических работ в ходе обучения и лабораторных работ.

Компетенция ПК-3 Способен разрабатывать информационные системы
Индикатор ПК-3.2. Проводит тестирование компонентов информационных систем, в том числе используя возможности специализированных цифровых платформ
Обучающийся умеет: выполнять проверку и отладку программного кода для микроконтроллера; проверять корректность работы микроконтроллера;
Обучающийся владеет: - навыками разработки и отладки программ
Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам выполнения практических работ и лабораторных работ, что свидетельствует

о накоплении опыта работы с программными средствами, программным окружением, программирования микроконтроллеров и опыта разработки схемотехнических решений..

4. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ОПРЕДЕЛЯЮЩИЕ ПРОЦЕДУРЫ ОЦЕНОВАНИЯ ЗНАНИЙ, УМЕНИЙ И НАВЫКОВ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ

КРИТЕРИИ ОЦЕНИВАНИЯ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
Компетенция ПК-3 Способен разрабатывать информационные системы **Индикатор ПК-3.1**. Осуществляет кодирование на современных языках программирования, в том числе используя возможности специализированных цифровых платформ для индивидуальной и совместной разработки информационных систем					
Знать: - общую архитектуру микроконтроллеров; типы и виды устройств микроконтроллеров; методологию (правила, команды, способы) разработки программного обеспечения для микро контроллера семейства Arduino .	Не знает	Фрагментарные знания об архитектуре микроконтроллеров; методологии (правила, команды, способы) разработки программного обеспечения для микро контроллера семейства Arduino .	Общие, но не структурированные знания об архитектуре микроконтроллеров; методологии (правила, команды, способы) разработки программного обеспечения для микро контроллера семейства Arduino .	Сформированные, но содержащие отдельные пробелы знания об архитектуре микроконтроллеров; методологии (правила, команды, способы) разработки программного обеспечения для микро контроллера семейства Arduino .	Сформированные систематизированные прочные знания об архитектуре микроконтроллеров; методологии (правила, команды, способы) разработки программного обеспечения для микро контроллера семейства Arduino .
Уметь: - программировать режимы работы устройств микроконтроллера; программировать обмен данными с помощью стандартных интерфейсов; работу с цифровыми датчиками;	Не умеет	Частично освоенные умения программировать режимы работы устройств микроконтроллера; программировать обмен данными с помощью стандартных интерфейсов; работу с цифровыми датчиками;	В целом освоенные, но не подкрепляемые знаниями и самостоятельностью выполнения умения программировать режимы работы устройств микроконтроллера; программировать обмен	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения программировать режимы работы устройств микроконтроллера; программировать обмен данными с	Успешно сформированные, самостоятельно и осознанно выполняемые умения программировать режимы работы устройств микроконтроллера; программировать обмен данными с помощью стандартных интерфейсов; работу с

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
			данными с помощью стандартных интерфейсов; работу с цифровыми датчиками;	помощью стандартных интерфейсов; работу с цифровыми датчиками;	цифровыми датчиками;
Владеть: - языками процедурного и объектно-ориентированного программирования, навыками разработки и отладки программ; - методами программирования устройств микроконтроллера	Не владеет	Фрагментарно сформированные навыки решения задач: на языках процедурного и объектно-ориентированного программирования, с применением навыков разработки и отладки программ; методами программирования устройств микроконтроллера	В целом сформированные, но выполняемые на элементарном уровне навыки решения задач: на языках процедурного и объектно-ориентированного программирования, с применением навыков разработки и отладки программ; методами программирования устройств микроконтроллера	Сформированные, но не устойчивые в сложных ситуациях навыки решения задач: на языках процедурного и объектно-ориентированного программирования, с применением навыков разработки и отладки программ; методами программирования устройств микроконтроллера	Успешно сформированные, устойчивые, технологически грамотно выполняемые навыки решения задач на языках процедурного и объектно-ориентированного программирования, с применением навыков разработки и отладки программ; методами программирования устройств микроконтроллера
Компетенция ПК-3 Способен разрабатывать информационные системы **Индикатор ПК-3.2**. Проводит тестирование компонентов информационных систем, в том числе используя возможности специализированных цифровых платформ					
Знать: - организацию программ для микроконтроллеров; возможности и особенности работы микроконтроллеров семейства Arduino.	Не знает	Фрагментарные знания об организации программ для микроконтроллеров; возможности и особенности работы микроконтроллеров семейства Arduino.	Общие, но не структурированные знания об организации программ для микроконтроллеров; возможности и особенности работы микроконтроллеров семейства Arduino.	Сформированные, но содержащие отдельные пробелы знания об организации программ для микроконтроллеров; возможности и особенности работы микроконтроллеров семейства Arduino.	Сформированные систематизированные прочные знания об организации программ для микроконтроллеров; возможности и особенности работы микроконтроллеров семейства Arduino.

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
Уметь: - выполнять проверку и отладку программного кода для микроконтролера; проверять корректность работы микроконтролера;	Не умеет	Частично освоенные умения по выполнению проверки и отладки программного кода для микроконтроллера; умения проверять корректность работы микроконтроллера;	В целом освоенные, но не подкрепляемые знаниями и самостоятельностью по выполнению проверки и отладки программного кода для микроконтроллера; умения проверять корректность работы микроконтроллера;	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения по выполнению проверки и отладки программного кода для микроконтроллера; умения проверять корректность работы микроконтроллера;	Успешно сформированные, самостоятельно и осознанно выполняемые умения по выполнению проверки и отладки программного кода для микроконтроллера; умения проверять корректность работы микроконтроллера;
Владеть: - навыками разработки и отладки программ	Не владеет	Фрагментарно сформированные навыки работы с программными средствами	В целом сформированные, но выполняемые на элементарном уровне навыки работы с программными средствами	Сформированные, но не устойчивые в сложных ситуациях навыки работы с программными средствами	Успешно сформированные, устойчивые, технологически грамотно выполняемые навыки работы с программными средствами

ПРОЦЕДУРА ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ И КРИТЕРИИ ОЦЕНКИ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Для контроля усвоения обучающимися данной дисциплины учебным планом предусмотрен зачет или по результатам текущего контроля. Зачет в третьем семестре проводится в форме устного ответа на теоретический вопрос, сдачи отчета по практическим работам, беседы с преподавателем по материалам отчета. В ходе промежуточной аттестации перевод результатов работы обучающихся (результатов текущего контроля, рейтинговых баллов) в систему оценки знаний осуществляется с ориентацией на критерии оценивания сформированности компетенций следующим образом:

Отметка «**зачтено**» выставляется обучающемуся, который в итоговом отчете выполнил все практические задания на отметку не ниже «удовлетворительно», ответил на вопросы по некоторым заданиям, ответил на устный вопрос преподавателя из списка вопросов к зачету, показал сформированные систематизированные прочные знания об общей архитектуре микроконтроллеров; типах и видах устройств микроконтроллеров; методологии (правила, команды, способы) разработки программного обеспечения для микро контроллера

семейства Arduino; по организации программ для микроконтроллеров; возможности и особенности работы микроконтроллеров семейства Arduino

Отметка «**не зачтено**» выставляется обучающемуся, который не выполнил в полном объеме все практические задания, соответственно не предоставил на контрольное мероприятие итоговый отчет, имеет фрагментарные знания об общей архитектуре микроконтроллеров; типах и видах устройств микроконтроллеров; методологии (правила, команды, способы) разработки программного обеспечения для микро контроллера семейства Arduino; по организации программ для микроконтроллеров; возможности и особенности работы микроконтроллеров семейства Arduino

5. ДИАГНОСТИЧЕСКИЕ ЗАДАНИЯ ДЛЯ ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ, ХАРАКТЕРИЗУЮЩИХ УРОВЕНЬ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ

Компетенция ПК-3 Способен разрабатывать информационные системы.
Обучающийся знает: общую архитектуру микроконтроллеров; типы и виды устройств микроконтроллеров; методологию (правила, команды, способы) разработки программного обеспечения для микро контроллера семейства Arduino; организацию программ для микроконтроллеров; возможности и особенности работы микроконтроллеров семейства Arduino.
Обучающийся умеет: программировать режимы работы устройств микроконтроллера; программировать обмен данными с помощью стандартных интерфейсов; работу с цифровыми датчиками; выполнять проверку и отладку программного кода для микроконтроллера; проверять корректность работы микроконтроллера.
Обучающийся владеет: языками процедурного и объектно- ориентированного программирования, навыками разработки и отладки программ; методами программирования устройств микроконтроллера; навыками разработки и отладки программ.

ЗАДАНИЯ ЗАКРЫТОГО ТИПА

Задание 1 Какое максимальное значение возвращает функция analogRead() на плате Arduino Nano?
а) 255
б) 1023
в) 4095
Ответ б) 1023. АЦП платы Arduino Nano имеет разрядность 10 бит, поэтому диапазон значений составляет от 0 до \(2^{10} - 1\).

Задание 2 Какая функция позволяет выполнять код параллельно, не блокируя работу процессора, в отличие от delay()?
а) millis()
б) pulseIn()
в) microtime()
Ответ а) millis(). Данная функция возвращает количество миллисекунд с момента старта платы и позволяет строить неблокирующие таймеры.

Задание 3 Сколько аналоговых входов (A0–A7) доступно на оригинальной плате Arduino Nano?
а) 6
б) 8
в) 10
Ответ б) 8. В отличие от Arduino Uno (где их 6), Nano в том же микроконтроллере ATmega328P задействует две дополнительные лапки под аналоговые входы A6 и A7.

Задание 4 Какой тип данных занимает в памяти Arduino Nano ровно 1 байт?
а) int
б) float
в) byte
Ответ в) byte. Тип byte хранит беззнаковое число от 0 до 255 и занимает ровно 8 бит (1 байт). int на AVR занимает 2 байта.

Задание 5 Какое ключевое слово используется для объявления переменной, значение которой нельзя изменить в процессе работы программы?

а) static
б) const
в) void

Ответ б) const. Модификатор const защищает переменную от случайного изменения, делая её константой.

Задание 6 Для чего используется функция pinMode(7, INPUT_PULLUP)?
а) Для включения пина 7 на выход с высоким уровнем сигнала
б) Для настройки пина 7 на вход с подключением встроенного подтягивающего резистора к VCC
в) Для настройки пина 7 на вход с подключением стягивающего резистора к GND

Ответ б) Для настройки пина 7 на вход с подключением встроенного подтягивающего резистора к VCC. Это позволяет подключать кнопку напрямую между пином и землей (GND) без внешних резисторов.

Задание 7 Какой интерфейс связи использует встроенный на плате Arduino Nano чип-конвертер (например, CH340 или FT232RL) для общения с компьютером?
а) SPI
б) I2C
в) UART

Ответ в) UART. Чип-конвертер преобразует сигналы USB от компьютера в аппаратный последовательный интерфейс UART (пины RX/TX).

Задание 8 К каким пинам Arduino Nano необходимо подключать линии SDA и SCL для работы с интерфейсом I2C?
а) A4 (SDA) и A5 (SCL)
б) D11 (SDA) и D12 (SCL)
в) D2 (SDA) и D3 (SCL)

Ответ а) A4 (SDA) и A5 (SCL). У микроконтроллера ATmega328P аппаратные пины интерфейса TWI (I2C) совмещены с аналоговыми входами A4 и A5.

Задание 9 Что произойдет, если в функции digitalWrite(pin, value) передать значение 5?
а) Пин выдаст напряжение 5 Вольт
б) Произойдет ошибка компиляции
в) Пин воспримет это как высокий уровень (HIGH)

Ответ в) Пин воспримет это как высокий уровень (HIGH). В среде Arduino любое значение переменной, отличное от нуля (логического нуля), интерпретируется как логическая единица (HIGH).

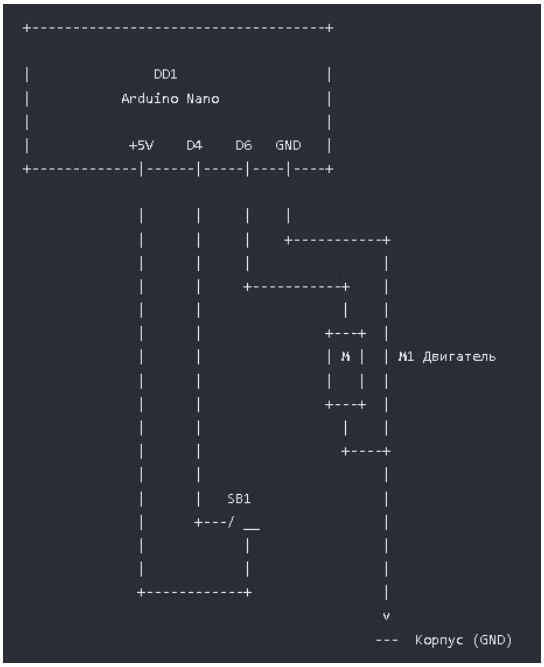
Задание 10 Какое прерывание (interrupt) соответствует цифровому пину D2 на плате Arduino Nano при использовании функции attachInterrupt()?
а) 0
б) 1
в) 2

Ответ а) 0. На плате Arduino Nano внешние прерывания доступны на пинах D2 (прерывание 0) и D3 (прерывание 1).

ЗАДАНИЯ ОТКРЫТОГО ТИПА (15 заданий, из них 5 – с развернутым ответом)
----------Уметь----------

Задание 1. Кнопочный включатель вентилятора
Идея: Мотор выполняет роль вентилятора. При нажатии на кнопку мотор включается, при отпускании — выключается. Нарисовать принципиальную схему, составить скетч.
Карта соединений:
- Кнопка: Клемма 1 \(\rightarrow \) **D4**, Клемма 2 \(\rightarrow \) **GND** (используем встроенную подтяжку INPUT_PULLUP).
- Мотор: Вход управления \(\rightarrow \) **D6**, Второй контакт \(\rightarrow \) **GND**.

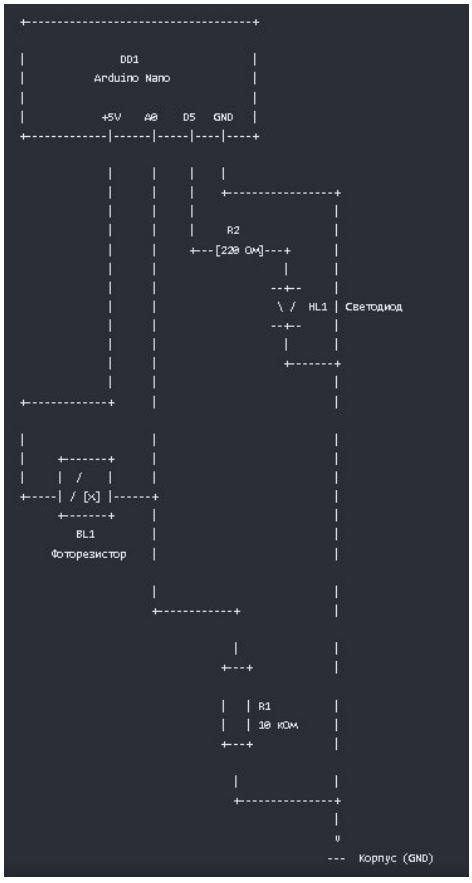
Ответ: Принципиальная схема



Задание 2. Автоматический ночной светильник
Идея: Фоторезистор следит за освещенностью в комнате. Как только становится темно, автоматически загорается светодиод. Нарисовать принципиальную схему, составить скетч.
Карта соединений:

- Фоторезистор: Сигнальный контакт \(\rightarrow \) **A0** (питание модуля берется с шины 5V/GND набора).
- Светодиод: Плюс (Анод) \(\rightarrow \) **D5**, Минус (Катод) \(\rightarrow \) **GND**.

Ответ: Принципиальная схема

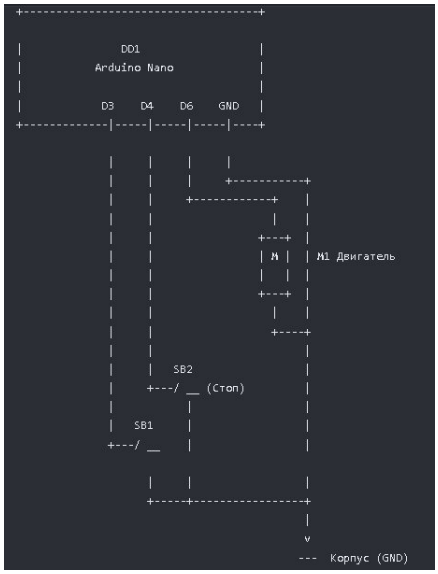


Задание 3. Двухкнопочный переключатель (Старт/Стоп)
Идея: Первая кнопка запускает постоянное вращение мотора. Мотор крутится, даже если кнопку отпустили. Вторая кнопка полностью останавливает мотор. Нарисовать принципиальную схему, составить скетч.

Карта соединений:

- Кнопка 1 (Старт): Клеммы \(\rightarrow \) **D3** и **GND**.
- Кнопка 2 (Стоп): Клеммы \(\rightarrow \) **D4** и **GND**.
- Мотор: Контакты \(\rightarrow \) **D6** и **GND**.

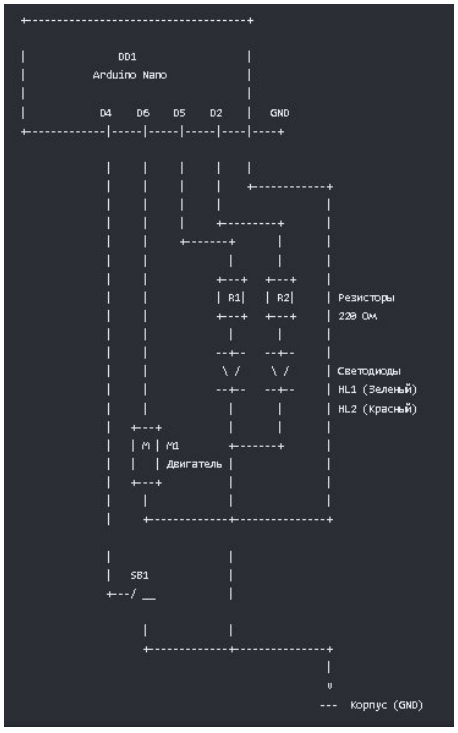
Ответ. Принципиальная схема



Задание 4. **Световой сигнализатор работы мотора**
Идея: Управление мотором осуществляется кнопкой. Когда мотор выключен — горит зеленый светодиод (безопасно). Когда мотор работает — зеленый тухнет, а красный светодиод начинает быстро мигать (опасность, вращение!). Нарисовать принципиальную схему, составить скетч.
Карта соединений:

- Кнопка: Вход \(\rightarrow \) **D4** и **GND**.
- Мотор: Вход \(\rightarrow \) **D6** и **GND**.
- Светодиод 1 (Зеленый): Вход \(\rightarrow \) **D5** и **GND**.
- Светодиод 2 (Красный): Вход \(\rightarrow \) **D2** и **GND**.

Ответ. Принципиальная схема

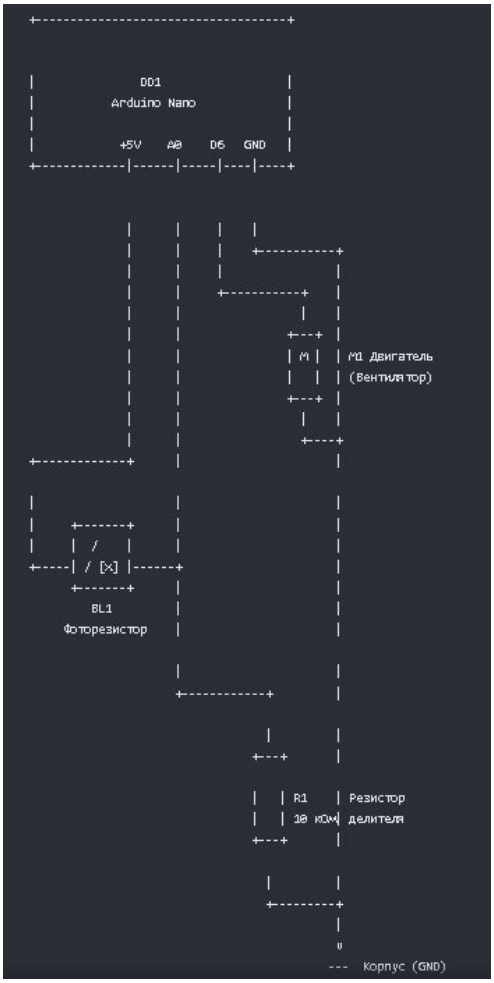


Задание 5. Сумеречный вентилятор (Регулятор скорости)

Идея: Чем темнее становится на улице (закрываем фоторезистор рукой), тем медленнее вращается мотор-вентилятор. Днем на солнце он работает на полную мощность. Нарисовать принципиальную схему, составить скетч.
Карта соединений:

- Фоторезистор: Сигнал \(\rightarrow \) **A0**.
- Мотор: Управление \(\rightarrow \) **D6** (Важно: пин D6 на Arduino Nano поддерживает ШИМ/PWM для плавной регулировки скорости).

Ответ. Принципиальная схема



Задание 6. Кнопочный включатель вентилятора
Идея: Мотор выполняет роль вентилятора. При нажатии на кнопку мотор включается, при отпускании — выключается. Составить скетч.
Карта соединений:

- Кнопка: Клемма 1 \(\rightarrow \) **D4**, Клемма 2 \(\rightarrow \) **GND** (используем встроенную подтяжку INPUT_PULLUP).
- Мотор: Вход управления \(\rightarrow \) **D6**, Второй контакт \(\rightarrow \) **GND**.

Ответ:

```
void setup() {
  pinMode(4, INPUT_PULLUP); // Кнопка
  pinMode(6, OUTPUT);       // Мотор
}
void loop() {
  // Нажатая кнопка выдает LOW (ноль)
  if (digitalRead(4) == LOW) {
    digitalWrite(6, HIGH); // Включить мотор
  } else {
    digitalWrite(6, LOW);  // Выключить мотор
  }
}
```

Задание 7. Автоматический ночной светильник
Идея: Фоторезистор следит за освещенностью в комнате. Как только становится темно, автоматически загорается светодиод. Составить скетч.
Карта соединений:

- Фоторезистор: Сигнальный контакт \(\rightarrow \) **A0** (питание модуля берется с шины 5V/GND набора).
- Светодиод: Плюс (Анод) \(\rightarrow \) **D5**, Минус (Катод) \(\rightarrow \) **GND**.

Ответ:

```
void setup() {
  pinMode(5, OUTPUT); // Светодиод
}
void loop() {
  int light = analogRead(A0); // Читаем датчик (0 - темно, 1023 - светло)
  if (light < 300) {
    digitalWrite(5, HIGH);   // Темно -> зажечь свет
  } else {
    digitalWrite(5, LOW);    // Светло -> выключить
  }
}
```

Задание 8. Двухкнопочный переключатель (Старт/Стоп)
Идея: Первая кнопка запускает постоянное вращение мотора. Мотор крутится, даже если кнопку отпустили. Вторая кнопка полностью останавливает мотор. Составить скетч.
Карта соединений:

- Кнопка 1 (Старт): Клеммы \(\rightarrow \) **D3** и **GND**.
- Кнопка 2 (Стоп): Клеммы \(\rightarrow \) **D4** и **GND**.
- Мотор: Контакты \(\rightarrow \) **D6** и **GND**.

Ответ:

```
bool isRunning = false;
void setup() {
  pinMode(3, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
  pinMode(6, OUTPUT);
}
void loop() {
  if (digitalRead(3) == LOW) isRunning = true;  // Старт
  if (digitalRead(4) == LOW) isRunning = false; // Стоп

  digitalWrite(6, isRunning ? HIGH : LOW);
}
```

Задание 9. Световой сигнализатор работы мотора
Идея: Управление мотором осуществляется кнопкой. Когда мотор выключен — горит зеленый светодиод (безопасно). Когда мотор работает — зеленый тухнет, а красный светодиод начинает быстро мигать (опасность, вращение!). Составить скетч.
Карта соединений:

- Кнопка: Вход \(\rightarrow \) **D4** и **GND**.
- Мотор: Вход \(\rightarrow \) **D6** и **GND**.
- Светодиод 1 (Зеленый): Вход \(\rightarrow \) **D5** и **GND**.
- Светодиод 2 (Красный): Вход \(\rightarrow \) **D2** и **GND**.

Ответ:

```
unsigned long prevMillis = 0;
bool ledState = LOW;

void setup() {
  pinMode(4, INPUT_PULLUP);
  pinMode(6, OUTPUT);
  pinMode(5, OUTPUT); // Зеленый
  pinMode(2, OUTPUT); // Красный
}
void loop() {
  if (digitalRead(4) == LOW) { // Мотор включен
    digitalWrite(6, HIGH);
    digitalWrite(5, LOW); // Гасим зеленый

    // Мигаем красным без задержки delay
    if (millis() - prevMillis >= 200) {
      prevMillis = millis();
      ledState = !ledState;
      digitalWrite(2, ledState);
    }
  } else { // Мотор выключен
    digitalWrite(6, LOW);
    digitalWrite(2, LOW);  // Гасим красный
    digitalWrite(5, HIGH); // Зажигаем зеленый
  }
}
```

Задание 10. Сумеречный вентилятор (Регулятор скорости)
Идея: Чем темнее становится на улице (закрываем фоторезистор рукой), тем медленнее вращается мотор-вентилятор. Днем на солнце он работает на полную мощность. Составить скетч.
Карта соединений:

- Фоторезистор: Сигнал \(\rightarrow \) **A0**.

- Мотор: Управление \(\rightarrow \) **D6** (Важно: пин D6 на Arduino Nano поддерживает ШИМ/PWM для плавной регулировки скорости).

Ответ:

```
void setup() {
  pinMode(6, OUTPUT);
}
void loop() {
  int lightVal = analogRead(A0);
  // Переводим значения датчика (0-1023) в скорость мотора (0-255)
  int motorSpeed = map(lightVal, 0, 1023, 0, 255);

  analogWrite(6, motorSpeed); // Плавное управление скоростью
  delay(30);
}
```

----------Владеть----------

Задание 11. Опишите назначение и логику работы базовых функций **setup()** и **loop()**. В чем заключается главный недостаток использования метода **delay()** для создания временных задержек в микроконтроллерных системах?

Ответ.
Логика setup() и loop(): При запуске микроконтроллера ATmega328P на плате Arduino Nano сначала однократно выполняется функция setup(). В ней инициализируются ресурсы: настраиваются режимы работы пинов (pinMode), запускается последовательный порт (Serial.begin). После этого управление передается функции loop(), которая выполняется в бесконечном цикле, пока на плату подается питание. Это обеспечивает непрерывный опрос датчиков и управление исполнительными механизмами.
Недостаток delay(): Функция delay() реализует метод *блокирующей задержки*. Микроконтроллер останавливает выполнение основного кода и просто «считает такты» внутри функции. В этот момент он абсолютно «слеп»: если в процессе delay(5000) пользователь нажмет на кнопку аварийного стопа или датчик зафиксирует критическое изменение, контроллер этого не заметит. Это делает устройство неотзывчивым и опасным в реальных задачах автоматики.

Задание 12. Объясните разницу между режимами работы цифрового пина INPUT и INPUT_PULLUP. Какую проблему в схемотехнике решает режим INPUT_PULLUP при подключении обычной тактовой кнопки, и как этот режим влияет на логику написания кода (состояние пина при нажатой и отпущенной кнопке)?.

Ответ.
Разница режимов: В режиме INPUT пин находится в состоянии высокого импеданса (входного сопротивления) и просто замеряет напряжение. В режиме INPUT_PULLUP внутри микроконтроллера параллельно входу подключается встроенный подтягивающий резистор номиналом от 20 до 50 кОм, соединенный с линией питания +5В.
Решаемая проблема: Если подключить кнопку к пину в режиме INPUT без внешних компонентов, то при отпущенной кнопке пин окажется «подвешен в воздухе» (не подключен ни к 5В, ни к GND). Он начнет собирать электромагнитные наводки из воздуха, и функция digitalRead() будет хаотично возвращать то HIGH, то LOW. Чтобы этого избежать, в классической схемотехнике приходится ставить внешний подтягивающий резистор. Режим INPUT_PULLUP позволяет отказаться от лишних деталей и подключить кнопку напрямую между пином Nano и землей (GND). Встроенный резистор всегда жестко удерживает на пине стабильные 5В, пока кнопка отпущена.

Влияние на код: Этот режим инвертирует привычную логику. Когда кнопка **отпущена**, ток течет через внутренний резистор на пин, и digitalRead() возвращает HIGH (1). Когда кнопка **нажата**, пин напрямую соединяется с GND, весь ток уходит в землю, и digitalRead() возвращает LOW (0). Поэтому условие нажатия кнопки в коде всегда проверяется как if (digitalRead(pin) == LOW).

Задание 13. Что такое Монитор последовательного порта (Serial Monitor) в среде Arduino IDE? Опишите пошаговый метод отладки программы, если подключенный к аналоговому входу A0 фоторезистор выдает некорректные (или статичные) значения. Какие команды языка для этого необходимы?

Ответ.
Что такое Serial Monitor: Это программный инструмент в Arduino IDE, который позволяет обмениваться текстовыми данными между компьютером и платой Arduino Nano через USB-кабель (по интерфейсу UART). Это главный инструмент разработчика для «заглядывания внутрь» работающего микроконтроллера при отсутствии дорогого аппаратного отладчика.
Метод отладки фоторезистора: Если данные с датчика кажутся неверными, метод отладки с помощью вывода промежуточных переменных выглядит так:

- В функции setup() необходимо инициализировать последовательный порт на стандартной скорости: Serial.begin(9600);.
- В функции loop() объявляется переменная для хранения сырых данных с АЦП и считывается значение: int rawLight = analogRead(A0);.
- Сразу после считывания значение отправляется в компьютер с переносом строки: Serial.println(rawLight);.
- Для предотвращения переполнения буфера порта добавляется небольшая задержка, например, delay(200);.
- Разработчик открывает Монитор порта в IDE и анализирует поведение чисел. Если при закрытии датчика рукой числа не меняются (стоят на 0 или 1023), то проблема в схемотехнике (перепутаны провода, отсутствует резистор делителя). Если числа меняются, но в узком диапазоне (например, от 500 до 600), программист понимает, что нужно скорректировать пороговые значения в условиях if или применить функцию масштабирования map().

Задание 14. Физический цифровой пин микроконтроллера может выдавать только два состояния: 0В или 5В. Каким методом программирования на Arduino Nano достигается эффект плавного изменения скорости вращения мотора или яркости светодиода?

Ответ.
Метод ШИМ: Плавная регулировка мощности реализуется методом Широтно-Импульсной Модуляции (ШИМ, или англ. *PWM — Pulse Width Modulation*). Вместо изменения именно *величины* напряжения, микроконтроллер начинает с очень высокой частотой (около 490 или 980 Гц) переключать пин между состояниями 0В и 5В. Эффект изменения мощности достигается за счет изменения *скважности* — отношения времени, когда пин находится в состоянии HIGH, к общему периоду импульса (это называется коэффициентом заполнения). Инерционные устройства (такие как обмотка мотора или человеческий глаз, воспринимающий свет от светодиода) усредняют эти импульсы. Если пин 50% времени включен и 50% выключен, мотор будет вращаться так, будто на него подали постоянные 2.5В.

Задание 15. Каковы ограничения встроенной функции analogWrite() (диапазон значений, на каких пинах Nano она работает)?

Ответ.
Функция analogWrite(): Для управления ШИМ используется встроенная функция analogWrite(pin, value). Вопреки слову *analog* в названии, она выдает именно цифровой ШИМ-сигнал.

- **Диапазон:** Таймеры микроконтроллера, отвечающие за генерацию базового ШИМ, являются 8-битными. Поэтому аргумент value принимает значения строго от 0 (0% мощности, постоянный LOW) до 255 (100% мощности, постоянный HIGH).
- **Ограничения по пинам:** На плате Arduino Nano аппаратный ШИМ реализован не на всех пинах, а только на тех, рядом с которыми на текстолите платы нарисован знак тильды (~). Это цифровые пины: **D3, D5, D6, D9, D10 и D11**. Попытка вызвать analogWrite() на других пинах приведет к тому, что при значениях меньше 128 пин выдаст обычный LOW, а при 128 и больше — обычный HIGH.

6. ЛИСТ СОГЛАСОВАНИЯ

Составил:

Н.Б.Стрекалова, д.п.н., доцент 

(подпись)

Заведующий кафедрой

Н.Б.Стрекалова, д.п.н., доцент 

(подпись)

Заведующий выпускающей кафедрой

Н.Б.Стрекалова, д.п.н., доцент 

(подпись)