Министерство науки и высшего образования Российской Федерации

Томский Государственный Университет
систем управления и радиоэлектроники

Лобода Ю.О.
Черных М.

ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

Методические указания к лабораторным работам для студентов технических специальностей

Томск
2026

УДК 681.325.5
ББК 78.373.3
ЛБ.8

Рецензент:
Антипин М.Е., доцент кафедры управления инновациями ТУСУР, кан. физ. мат. наук

Лобода, Юлия Олеговна, Черных, Максим
ЛЧ.1 Лабораторный практикум: методические указания по организации по организации и проведению лабораторных работ для студентов технических направлений и специальностей / Ю.О. Лобода, М. Черных – Томск : гос. ун-т систем упр. и радиоэлектроники, 2026. – 24 с.

Настоящие методические указания для студентов составлены с учетом требований федеральных государственных образовательных стандартов высшего образования (ФГОС ВО). Методические указания содержат рекомендации к проведению лабораторных работ и предназначены для студентов технических специальностей.

Одобрено на заседании каф. управления инновациями,
протокол №11 от 05.07.2025

УДК 681.325.5

ББК.78.373.3

Оглавление

Введение

Дисциплина «Основы микропроцессорной техники» является одной из ключевых в подготовке студентов технических специальностей, так как закладывает фундаментальные знания и практические навыки работы с современными микроконтроллерами и микропроцессорными системами. В условиях стремительного развития электроники и автоматизации умение разрабатывать, программировать и отлаживать микропроцессорные устройства становится необходимым для инженеров в самых разных областях — от промышленной автоматики и робототехники до приборостроения и систем управления. Настоящие методические указания предназначены для проведения лабораторных работ. Они обеспечивают закрепление теоретических знаний и формирование практических навыков, связанных с основными компонентами и интерфейсами микропроцессорных систем. В ходе выполнения лабораторных работ студенты знакомятся с принципами работы портов ввода-вывода, обеспечивающих взаимодействие микроконтроллера с периферийными устройствами, использованием сериального интерфейса для обмена данными и отладки программ, принципами работы аналогово-цифрового преобразователя для измерения аналоговых сигналов, а также с основами формирования широтно-импульсной модуляции для управления исполнительными устройствами, такими как двигатели и светодиоды. Методические материалы ориентированы на практическое освоение указанных тем и формирование у студентов навыков проектирования простейших микропроцессорных систем.

1 Методические указания к проведению лабораторных работ

Данный раздел содержит ключевые рекомендации по организации и выполнению лабораторных работ, описанных в настоящем методическом указании. Соблюдение этих указаний поможет правильно подготовиться к выполнению заданий, организовать рабочее пространство и эффективно зафиксировать полученные результаты.

Для этого необходимо:

внимательно ознакомиться с теоретическим материалом;

выполнить лабораторную работу в присутствии преподавателя;

зафиксировать результаты наблюдений и расчетов в протоколе;

ответить на контрольные вопросы и защитить выполненную работу.

Отчет должен содержать:

титульный лист (с указанием ФИО, группы, темы, даты);

цель работы;

краткие теоретические сведения;

перечень оборудования;

ход выполнения работы;

результаты и выводы;

ответы на контрольные вопросы.

Для выполнения лабораторных работ, представленных в данном методическом указании, необходимо использовать специализированное оборудование, включая отладочную плату с микроконтроллером (например, Arduino Mega на базе ATmega2560 или Arduino Uno на базе ATmega328P), набор базовых электронных компонентов.

1.1 Лабораторная работа «Введение в Arduino. Порты ввода-вывода».

1. Цель работы

Лабораторная работа направлена на изучение работы с отладочными платами на базе микроконтроллеров «ATmega2560» или «ATmega328P». Также для изучения принципов работы портов ввода-вывода на примере использования светодиодов и кнопок.

2. Краткие теоретические сведения

Arduino — торговая марка аппаратно-программных средств построения и прототипирования простых систем, моделей и экспериментов в области электроники, автоматики, автоматизации процессов и робототехники.

Аппаратная часть представляет собой электронные платы с микроконтроллером, сопутствующими элементами (стабилизатор питания, кварцевый резонатор, блокировочные конденсаторы и т.п.), портом для связи с персональным компьютером, разъемами для сигналов ввода-вывода и т.п. В рамках данной дисциплины, лабораторные работы разработаны для отладочных плат Arduino UNO/MEGA, самых популярных из семейства плат Ардуино, возможности которых наглядно отражены на рисунке 1. 1.

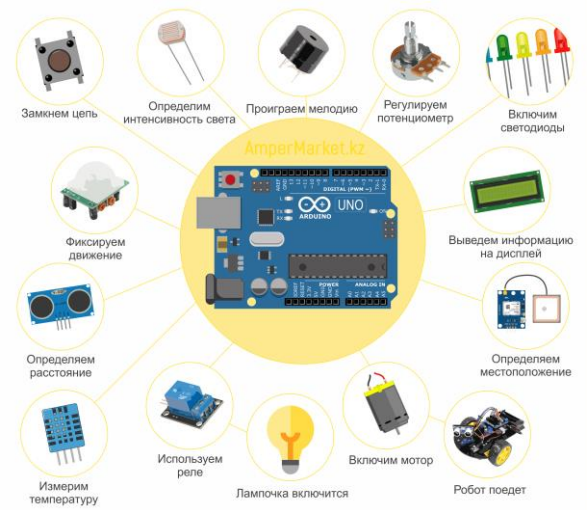

Рисунок 1.1 – Функционал Arduino

Arduino UNO, это отладочная плата выполненная на базе микроконтроллера ATmega328, а более новые версии на ATmega328P. Платформа имеет 14 цифровых вход/выходов (6 из которых могут использоваться как выходы ШИМ), 6 аналоговых входов, кварцевый генератор 16 МГц, разъем USB, силовой разъем, разъем ICSP и кнопку перезагрузки. Для работы необходимо подключить платформу к компьютеру посредством кабеля USB, либо подать питание при помощи адаптера AC/DC или батареи (рисунок 1.2).

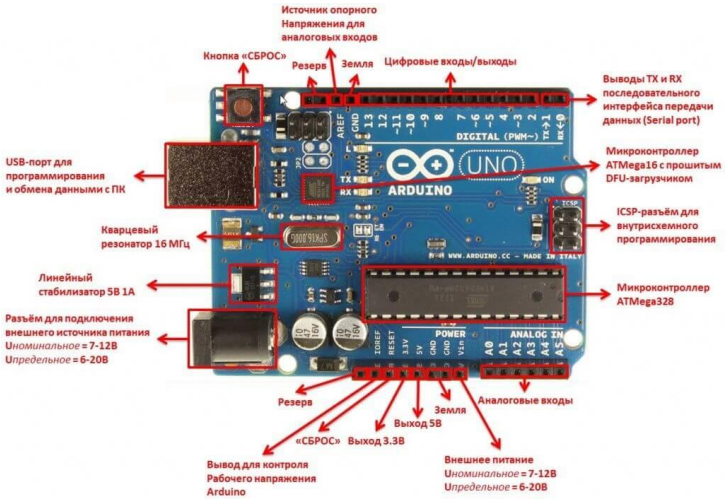

Рисунок 1.2 – Отладочная плата Arduino Uno

Arduino Mega, данная отладочная плата построена на микроконтроллере ATmega2560. Плата имеет 54 цифровых входа/выходов (14 из которых могут использоваться как выходы ШИМ), 16 аналоговых входов, 4 последовательных порта UART, кварцевый генератор 16 МГц, USB коннектор, разъем питания, разъем ICSP и кнопка перезагрузки (рисунок 1.3).

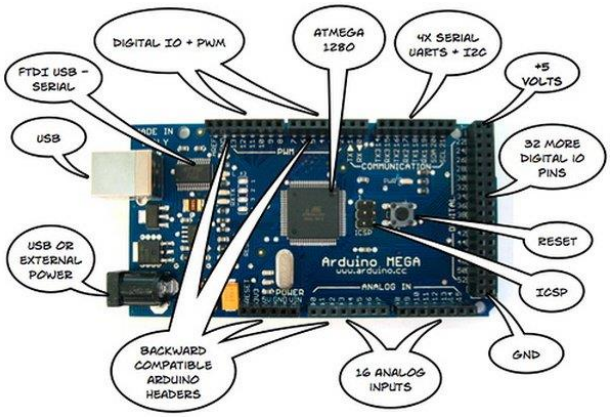

Рисунок 1.3 – Отладочная плата Arduino Mega

Светодиод — это полупроводниковый элемент, который, при прохождении через него электрического тока излучает свет. Светодиод пропускает ток только в одном направлении от анода к катоду. Это значит, что при подключении необходимо соблюдать полярность (рисунок 1.4).

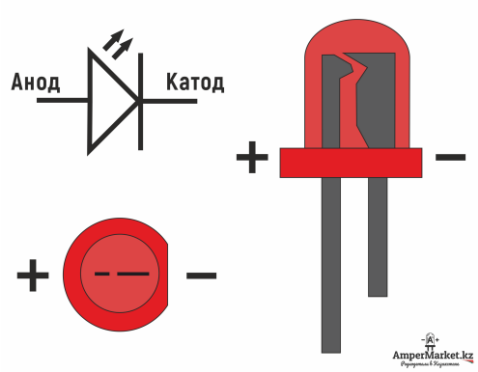

Рисунок 1.4 – Светодиод

Тактовая кнопка — механизм, замыкающий цепь при наличии давление на толкатель.

Для подключения кнопки, необходимо подвести плюс питания на кнопку и через резистор пустить на землю, для снятия цифрового сигнала (нажата кнопка или нет) необходимо подтянуть цифровой пин в цепь. Если цифровой сигнал снимать между плюсом питания и кнопкой, то при нажатии на кнопку на контроллер придет значение “0”, если наоборот - “1” (рисунок 1.5).

Рисунок 1.5 – Тактовая кнопка

Макетная плата — универсальная печатная плата для сборки и моделирования прототипов электронных устройств (рисунок 1.6).

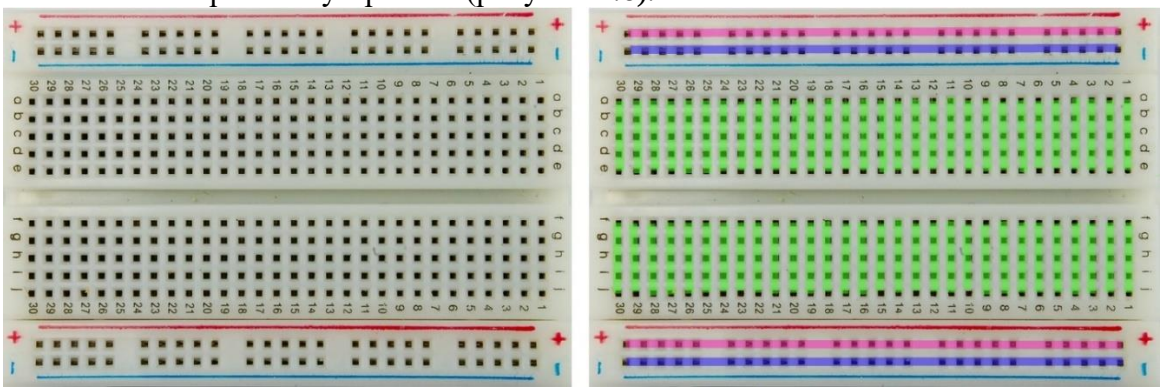

Рисунок 1.6 – Макетная плата

Центральные отверстия соединены параллельными рядами поперёк макетной платы, а не вдоль. В отличие от шин питания, которые размещены по краю макетной платы вдоль её краёв. Как видно, имеется две пары шин питания, что позволяет при необходимости подавать на плату два разных напряжения, например, 5 В и 3,3 В.

Arduino IDE — это C-подобная программная среда разработки, предназначенная для программирования всех плат ряда Ардуино (Arduino) (рисунок 1.7).

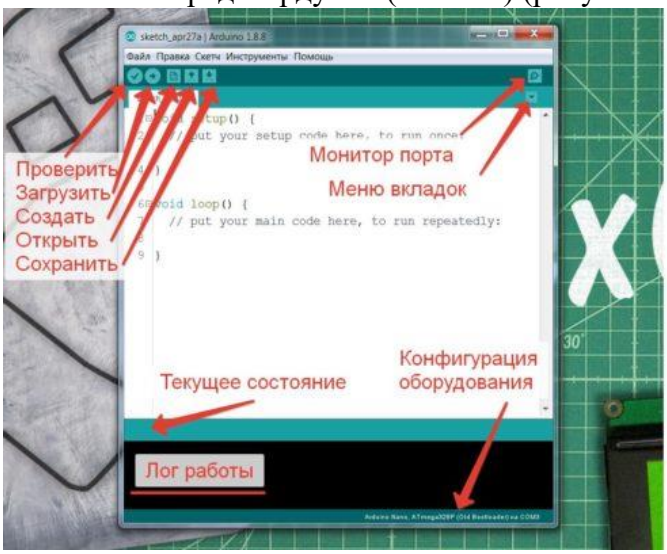

Рисунок 1.7 – Программное обеспечение Arduino IDE

Для выполнения работы по включению светодиода нужно знать несколько базовых команд Arduino:

pinMode(номер_пина, режим) — задаёт режим работы вывода: вход (INPUT) или выход (OUTPUT). Для светодиода пин нужно настроить как OUTPUT, чтобы можно было подать на него сигнал.

digitalWrite(номер_пина, состояние) — устанавливает на выводе логический уровень: HIGH (высокий уровень) что в рамках данной лабораторной работы позволяет включить светодиод или LOW (низкий уровень, выключить светодиод).

digitalRead(номер_пина) — считывает текущее состояние пина, настроенного как вход. Эта команда нужна, если нужно реагировать, например, на нажатие кнопки.

delay(миллисекунды) — делает паузу на заданное время в миллисекундах, позволяя создавать задержки между включением и выключением светодиода (например, мигание).

В Arduino IDE есть стандартная структура, которая включает подключение библиотек, определение констант, переменных и функций. Для подключения библиотек используется директива #include, с её помощью можно добавлять готовый код для работы с датчиками, дисплеями и другими устройствами. Например, #include <Wire.h> подключает библиотеку для работы по интерфейсу I2C. Директива #define используется для объявления констант и удобных имен для пинов или значений. Например, можно написать #define LED 13, чтобы вместо цифры 13 в коде использовать имя LED.

Функция void setup() выполняется один раз при включении питания или перезагрузке платы. В ней обычно задают режимы работы пинов с помощью команды pinMode, а также выполняют первоначальные настройки, например инициализацию связи по Serial.

После этого запускается функция void loop(), которая работает непрерывно в цикле. Именно в ней пишется основной код программы, который постоянно повторяется: например, включение и выключение светодиода, чтение состояния кнопок или датчиков, выполнение расчетов и управление другими устройствами. Кроме этих функций, студент может использовать иные или создавать собственные функции для упрощения кода и повторного использования одинаковых действий.

3. Пояснения к работе

Работа выполняется в программном обеспечении Arduino IDE. Для выполнения работы необходимо собрать схему по варианту опираясь на примеры представленные на рисунках 1.8-1.9.

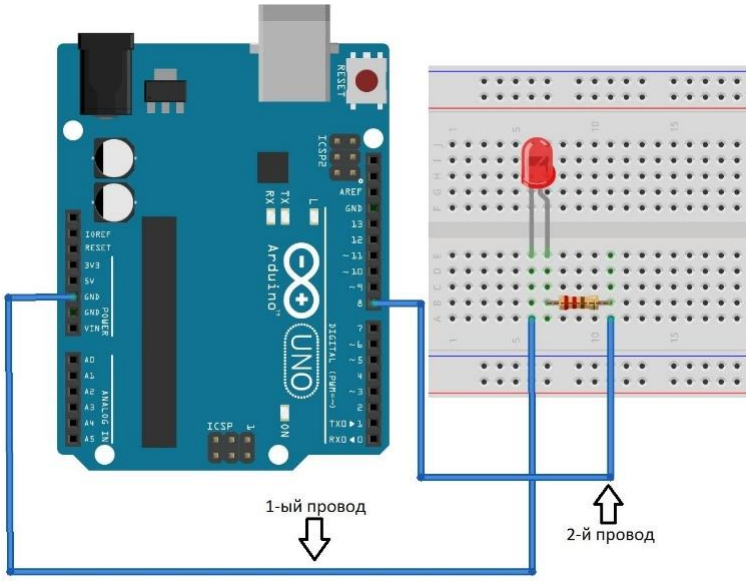

Рисунок 1.8 – Пример подключения светодиода к Arduino Uno через макетную плату с использованием резистора

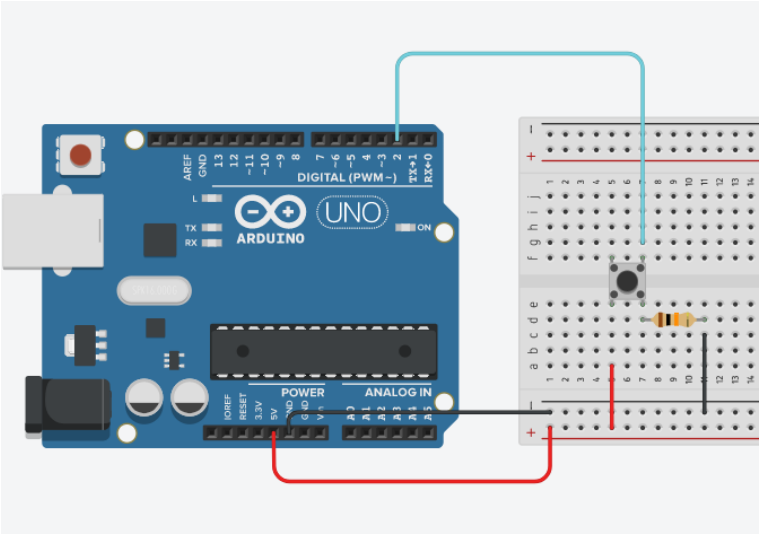

Рисунок 1.9 – Пример подключения тактовой кнопки к Arduino Uno

Для допуска к лабораторной работе предоставьте схему подключения для двух светодиодов к Arduino.

Варианты заданий для индивидуального выполнения представлены в таблице 1.

Таблица 1

1 Вариант Осуществить управление двумя светодиодами с помощью одной кнопки: при нажатии загорается 1 светодиод, при втором нажатии загорается 2 светодиод, при третьем нажатии светодиоды гаснут.
2 вариант Осуществить управление двумя светодиодами с помощью одной кнопки: при нажатии светодиоды начинают моргать 1 раз в секунду, при втором нажатии светодиоды должны начать моргать 2 раза в секунду при третьем нажатии светодиоды гаснут.
3 вариант Осуществить управление двумя светодиодами с помощью одной кнопки: при нажатии загорается 2 светодиода, при втором нажатии гаснет 1-ый светодиод.
4 вариант Осуществить управление двумя светодиодами с помощью одной кнопки: при нажатии загораются оба светодиода, при втором нажатии светодиоды начинают мигать 1 раз в секунду.
5 вариант Осуществить управление двумя светодиодами с помощью одной кнопки: при первом нажатии загораются оба светодиода, при втором нажатии оба светодиода гаснут.
6 вариант Осуществить управление двумя светодиодами с помощью одной кнопки: при первом нажатии загораются оба светодиода, при втором нажатии сначала гаснет 1 светодиод, через 3 секунды гаснет 2 светодиод.
7 вариант Осуществить управление двумя светодиодами с помощью одной кнопки: при запуске системы оба светодиода изначально горят, при первом нажатии светодиоды мигают 1 раз в секунду, при втором гаснут.

После сборки макетной платы, необходимо написать программный код в соответствии с вариантом.

4. Программа работы

После подготовки оборудования работа выполняется по следующему алгоритму:

1. Взять отладочную и макетную плату с необходимыми компонентами для выполнения лабораторной работы.
2. Подключить к макетной плате светодиод и кнопку (рисунки 1.7-1.8).
3. Создать блок-схемы алгоритмов.
4. В зависимости от варианта написать программный код.
5. Зафиксировать выполнение работы в виде фотографий макетной платы с пошаговым выполнением работы программы, которые необходимо внести в итоговый отчет.

5. Содержание отчета

В отчете представить результаты выполнения программы работы, анализ работы и выводы.

6. Контрольные вопросы

1. Что такое Arduino, опишите основные особенности и виды плат?
2. Как «прошить» Arduino, особенности написания программного кода?
3. Что такое макетная плата?
4. Что такое тактовая кнопка?
5. Что такое светодиод?
6. Как подключить светодиод? (необходимо нарисовать схему)
7. Как подключить кнопку? (необходимо нарисовать схему)
8. Возможности использования резистора для подключения светодиода.
9. Что такое ArduinoIDE?
10. Чем отличается Arduino UNO от Arduino MEGA?

1.2 Лабораторная работа «Монитор порта».

1. Цель работы

Изучить принципы обмена данными между микроконтроллером Arduino и компьютером через последовательный порт, освоить работу с последовательным монитором (Serial Monitor) в Arduino IDE, а также научиться передавать и принимать данные для диагностики и отладки работы устройств.

2. Краткие теоретические сведения

Монитор порта — инструмент в Arduino IDE, позволяющий обмениваться текстовыми и числовыми данными между микроконтроллером и компьютером через последовательный интерфейс (UART). Чтобы открыть монитор порта необходимо в программном обеспечении Arduino IDE нажать на «Монитор порта», как показано на рисунке 1.10.

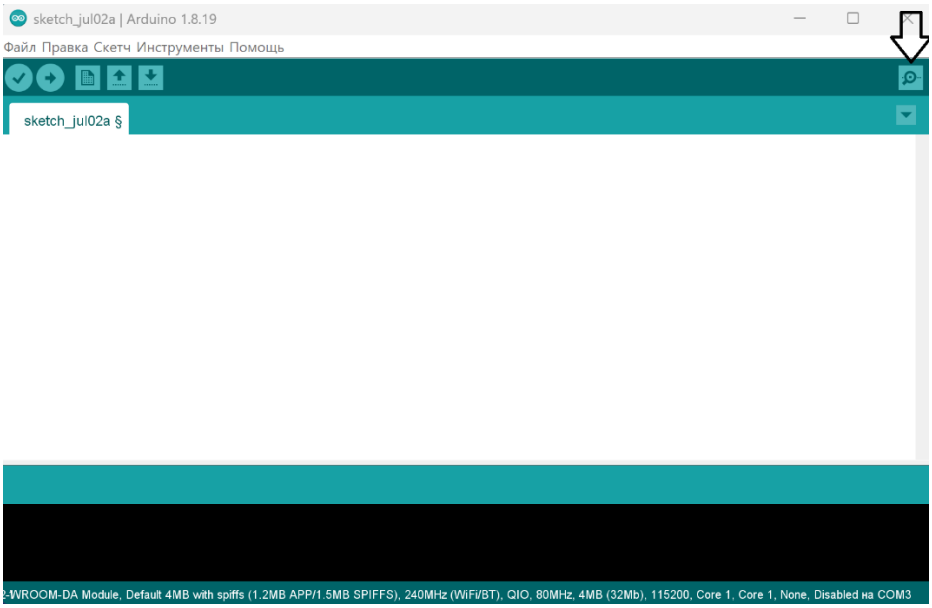

Рисунок 1.10 – Монитор порта

UART - это универсальный приёмник-передатчик, почти все микроконтроллеры имеют такой интерфейс для связи с внешними устройствами. При помощи преобразователя UART-USB данные с Arduino могут передаваться в монитор порта (рисунок 1.11).

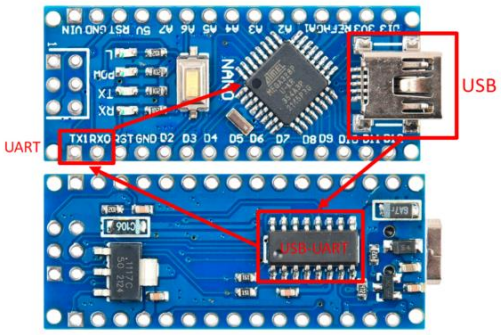

Рисунок 1.11 – Преобразователь UART-USB

COM-порт

При подключении Arduino на стороне компьютера создаётся виртуальный COM порт (последовательный порт).

COM-порт, или последовательный порт, это тип интерфейса для передачи данных по одному биту за раз, в отличие от параллельного порта, где биты передаются одновременно. Он используется для подключения периферийных устройств к компьютеру, таких как модемы, принтеры и некоторые промышленные устройства.

Методы Serial

Для работы с портом на стороне Arduino используется системный объект Serial, он позволяет читать и отправлять данные (рисунок 1.12).

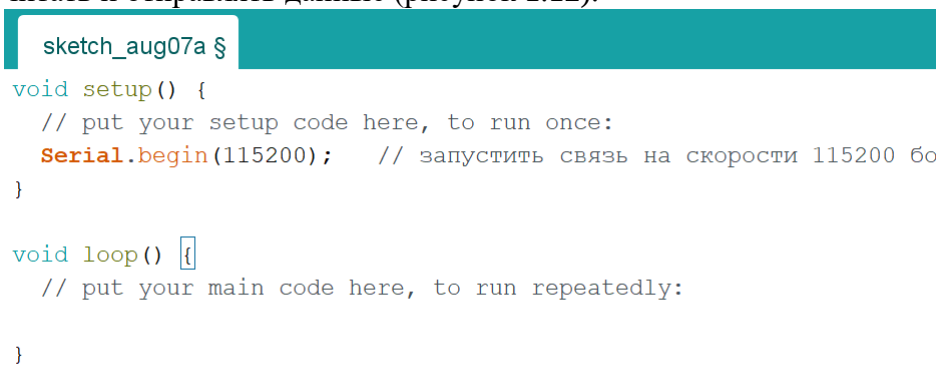

Рисунок 1.12 – Запуск связи на скорости 115200

Число 115200 — это скорость передачи (baud rate). Она должна совпадать со скоростью, установленной в мониторе порта.

Скорость передачи данных (baud rate) в основе своей обозначает скорость передачи данных по каналу связи. Хотя ее часто приравнивают к битам в секунду (bps), более точное определение учитывает количество сигнальных единиц, передаваемых в секунду, необходимых для представления этих битов.

Также можно остановить общение по Serial (рисунок 1.13).

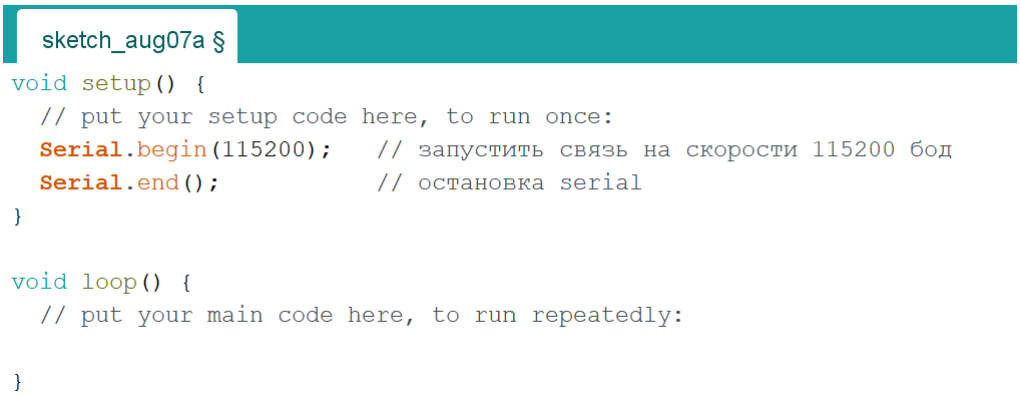

Рисунок 1.13 – Остановка передачи данных по Serial

Для отправки используются методы Serial.print(data) и Serial.println(data) – где «data» данные в виде текста, которые необходимо передать. Метод Serial.println отличается от метода Serial.print, переносом строки на новую (рисунок 1.14).

sketch_aug07a

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);   // запустить связь на скорости 115200 бод

}

void loop() {
  // put your main code here, to run repeatedly:
      Serial.print("data");
}
```

Рисунок 1.14 – Пример использования Serial.print

Чтение

В монитор порта также можно отправлять данные и читать их из программы. Для проверки наличия входящих данных используется метод available() - он вернёт количество входящих байт. В таком условии можно проверить, что есть хотя бы один байт для чтения (рисунок 1.15).

sketch_aug07a

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);   // запустить связь на скорости 115200 бод

}

void loop() {
  // put your main code here, to run repeatedly:
      Serial.print("data");
      if (Serial.available()) {
      // есть входящие данные
      }

}
```

Рисунок 1.15 – Проверка байтов на чтение

Запись

За запись отвечает Serial.write. Данная команда отправляет байты данных напрямую (без форматирования).

Используется для передачи необработанных данных (например, символов ASCII или бинарных значений) (рисунок 1.16).

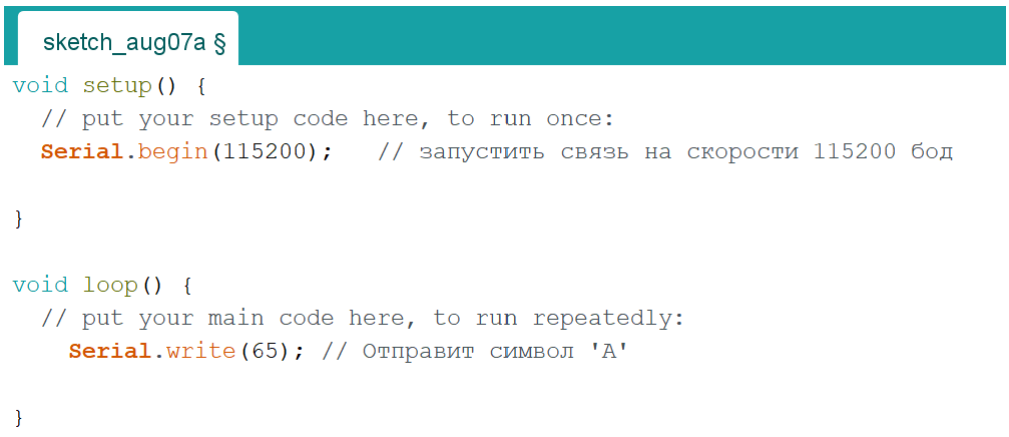

Рисунок 1.16 – Пример использования записи

3. Пояснения к работе

Работа выполняется в программном обеспечении Arduino IDE. Для выполнения работы необходимо выполнить задание, соответствующее варианту пользуясь информацией из раздела кратких теоретических сведений к лабораторной работе.

Варианты заданий для индивидуального выполнения представлены в таблице 2.

Таблица 2

1 вариант
Реализовать управление одним светодиодом и одной кнопкой через монитор порта. Вводится символ 'A' — светодиод загорается. При нажатии на кнопку светодиод гаснет. При каждом изменении состояния в монитор порта выводятся сообщения «Светодиод включен» и «Светодиод выключен».
2 вариант
Реализовать управление одним светодиодом и одной кнопкой через монитор порта. При вводе символа 'T' светодиод меняет состояние (если горит — гаснет, если выключен — загорается). Нажатие кнопки возвращает светодиод в выключенное состояние. В монитор порта выводится текущее состояние светодиода.
3 вариант
Реализовать управление одной кнопкой через монитор порта. При каждом нажатии кнопки увеличивается счётчик. Каждую секунду в монитор порта выводится количество нажатий. При вводе через монитор порта символа 'R' счётчик сбрасывается на ноль.
4 вариант
Реализовать управление одним светодиодом и одной кнопкой через монитор порта. Пользователь вводит символы через монитор порта. Если введена правильная последовательность (например, «123»), светодиод загорается. Нажатие кнопки сбрасывает состояние (гасит светодиод). Все сообщения о вводе и проверке отображаются в мониторе порта.
5 вариант
Реализовать управление одним светодиодом и одной кнопкой через монитор порта. При вводе символа 'A' светодиод включается на 5 секунд. Каждую секунду в монитор порта выводится оставшееся время до выключения: «Выключение через: 5», «Выключение через: 4» и так далее. Кнопкой можно досрочно выключить светодиод.

6 вариант
Реализовать управление одним светодиодом и одной кнопкой через монитор порта. Пользователь вводит через монитор порта число от 1 до 9. Светодиод мигает столько раз, сколько введено. Нажатие на кнопку, сбрасывает значение и светодиод должен потухнуть.
7 вариант
Реализовать управление одним светодиодом и одной кнопкой через монитор порта. Arduino ждёт символ через монитор порта. Если пришёл 'P', отправляет обратно сообщение «PING получен» и включает светодиод на 1 секунду. Нажатием на кнопку, светодиод включается.

4. Программа работы

Порядок выполнения работы

1. Подключите Arduino к компьютеру и откройте Arduino IDE.
2. Создайте новый скетч.
3. В функци setup() инициализируйте последовательный порт:
4. В функции loop() реализуйте ввод/вывод данных с использованием Serial.print(), Serial.println() и Serial.write().
5. Загрузите скетч и откройте монитор порта (Tools → Serial Monitor).
6. Изучите вывод программы, добавьте в отчет.

5. Содержание отчета

В отчете представить результаты выполнения программы работы, анализ работы и выводов.

6. Контрольные вопросы

1. Для чего используется последовательный порт в Arduino?
2. В чем разница между Serial.print() и Serial.write()?
3. Что делает Serial.println() дополнительно по сравнению с Serial.print()?
4. Как изменить скорость передачи данных?
5. Можно ли передавать числа и строки через Serial.write()?

1.3 Лабораторная работа «АЦП в Arduino».

1. Цель работы

Изучить принципы работы аналогово-цифрового преобразователя (АЦП) в микроконтроллере Arduino, освоить чтение аналоговых сигналов с помощью функции analogRead(), а также научиться обрабатывать и интерпретировать полученные данные для работы с аналоговыми датчиками.

2. Краткие теоретические сведения

Аналого-цифровой преобразователь (АЦП) — это устройство, которое преобразует аналоговое напряжение в цифровое значение. В Arduino Uno используется 10-битный АЦП, который возвращает значения от 0 до 1023, соответствующие напряжению от 0 до 5 В (по умолчанию).

Опорное напряжение - максимальное напряжение, которое можно подавать на АЦП. Минимальное - ноль, GND.

Разрешение (разрядность) - точность, количество значений, с которой АЦП измеряет напряжение от 0 до опорного, измеряется в битах.

Само аналоговое напряжение может иметь бесконечно большое разрешение, то есть меняться с шагом например 0.00000001 В и меньше. АЦП оцифровывает напряжение с конечной точностью, то есть часть информации теряется.

Порты Arduino UNO/MEGA, которые поддерживают обработку АЦП отличаются от других портов и имеют вид A0, A1 … A9.

Для измерения напряжения используется функция analogRead(порт), где порт - номер порта (A0, A1..). В каком виде передаётся номер пина лучше уточнить в описании к конкретной плате (рисунок 1.17).

sketch_aug07a §

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);   // запустить связь на скорости 115200 бод

}

void loop() {
  // put your main code here, to run repeatedly:
    analogRead(A0) // измерение напряжения на пине A0

}
```

Рисунок 1.17 – Пример использования analogRead()

Если к аналоговому входу Arduino подключить потенциометр, запитанный от источника напряжения 5 В, то для перевода цифрового значения, считанного с помощью функции analogRead(), в реальное напряжение необходимо использовать следующую формулу:

$$U = \frac{\text{значение АЦП}}{1023} \times 5.0\text{В}$$

Где, значение АЦП – целочисленное число от 0 до 1023, полученное с помощью analogRead().

5.0 В – напряжение питания потенциометра (опорное напряжение АЦП Arduino).

Эта формула позволяет получить значение напряжения на входе аналогового пина в вольтах, учитывая, что максимальное цифровое значение 1023 соответствует напряжению 5 В.

В данной лабораторной работе будет использоваться потенциометр и фоторезистор, формирования переменного аналогового сигнала. Потенциометр позволяет вручную изменять уровень напряжения, а фоторезистор — изменять его в зависимости от уровня освещённости. Считывание этих сигналов выполняется встроенным аналого-цифровым преобразователем (АЦП) микроконтроллера Arduino, что позволяет продемонстрировать процесс преобразования аналогового напряжения (0–5 В) в цифровое значение (0–1023) и последующую обработку полученных данных в программе.

Потенциометр — это трехконтактный резистор с регулируемым сопротивлением, используемый для деления напряжения. Изменение положения движка потенциометра изменяет соотношение сопротивлений между выводами, что приводит к изменению выходного напряжения на одном из контактов.

Фоторезистор — это полупроводниковый резистор, сопротивление которого изменяется в зависимости от уровня освещённости: при увеличении освещения сопротивление уменьшается, при снижении — увеличивается.

3. Пояснения к работе

Лабораторная работа выполняется в программном обеспечение Arduino IDE. Для выполнения работы необходимо подключить компоненты (фоторезистор или потенциометр) (рисунки 1.18,1.19). Далее при помощи изменения напряжения управлять светодиодом или выводить значения в монитор порта в соответствии с вариантом.

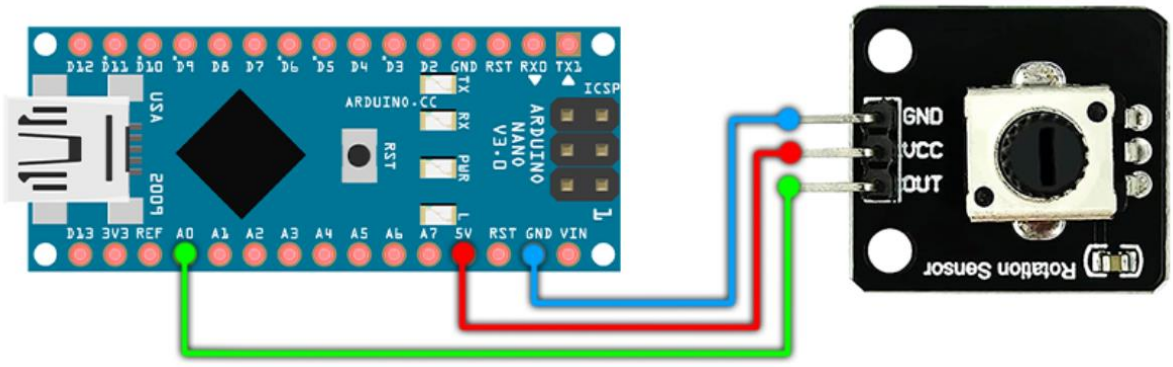

Рисунок 1.18 – Подключение потенциометра к Arduino

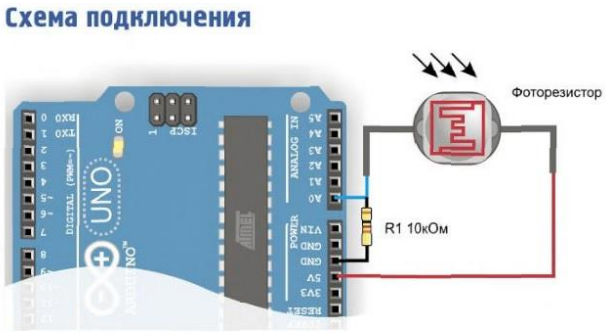

Рисунок 1.19 – Подключение фоторезистора к Arduino

Варианты заданий для индивидуального выполнения представлены в таблице 3.

Таблица 3.

1 вариант
Реализовать управление светодиодом при помощи АЦП. Считать данные с A0 и каждую секунду выводить в монитор порта «сырое» значение и напряжение в вольтах, при достижении 3В включать светодиод.
2 вариант
Реализовать управление светодиодом при помощи АЦП. Если напряжение выше 2,5 В — зажечь светодиод, если ниже — погасить. Выводить в монитор порта сообщение о состоянии светодиода.
3 вариант
Реализовать управление светодиодом при помощи АЦП. Порог включения светодиода задаётся изначально 2,5 В, но при нажатии кнопки он увеличивается на 0,5 В. Все изменения порога выводить в монитор порта.
4 вариант
Реализовать управление светодиодом при помощи АЦП. Если напряжение превышает 4 В, светодиод начинает мигать с частотой 2 Гц, при нормальном напряжении горит постоянно. В монитор порта выводить состояние системы.
5 вариант
Реализовать управление светодиодом при помощи АЦП. Считать значение с фоторезистора и выводить его в монитор порта каждые 500 мс. Если значение выше определённого порога (например, > 700), зажечь светодиод; если ниже — погасить.
6 вариант
Реализовать управление светодиодом при помощи АЦП. Считать данные с фоторезистора. Если освещённость падает ниже порога (например, < 300), светодиод начинает мигать с частотой 1 Гц. При восстановлении освещённости светодиод должен погаснуть. В монитор порта выводить текущее состояние («Светло», «Темно»).
7 вариант
Реализовать управление светодиодом при помощи АЦП. Каждые 2 секунды считывать значение освещённости и отправлять его в монитор порта вместе с номером измерения. После 10 измерений включить светодиод на 10 секунд, далее выключить.

4. Программа работы

1. Подключите потенциометр или фоторезистор к Arduino (средний вывод к A0, крайние — к +5 В и GND).

2. В функции setup() инициализируйте монитор порта.

3. Считайте значение с аналогового входа с помощью analogRead().

4. Выполните работу в соответствии с вариантом.

5. Содержание отчета

В отчете представить результаты выполнения программы работы, анализ работы и выводов.

6. Контрольные вопросы

1. Какой диапазон значений возвращает analogRead() и почему?

2. Как пересчитать значение АЦП в напряжение?

3. Можно ли изменить опорное напряжение АЦП на Arduino Uno?

4. Какие пины Arduino поддерживают аналоговый ввод?

1.4 Использование «ШИМ в Arduino».

1. Цель работы

Изучить принципы работы широтно-импульсной модуляции. Изучить аппаратный ШИМ, который используется в Arduino.

2. Краткие теоретические сведения

Широтно-импульсная модуляция (ШИМ) — метод управления мощностью на выходе за счёт изменения длительности импульсов при фиксированной частоте. На Arduino ШИМ реализован на цифровых пинах, поддерживающих функцию analogWrite().

Частота ШИМ

Частота измеряется в Герцах, Гц (Hertz, Hz), 1 Гц равен одному периоду колебаний в секунду (T на рисунке 1.20). Период и частота - взаимно обратные величины, период измеряется в секундах: частота = 1/T, T-период. Периодом считается время между одинаковыми точками сигнала.

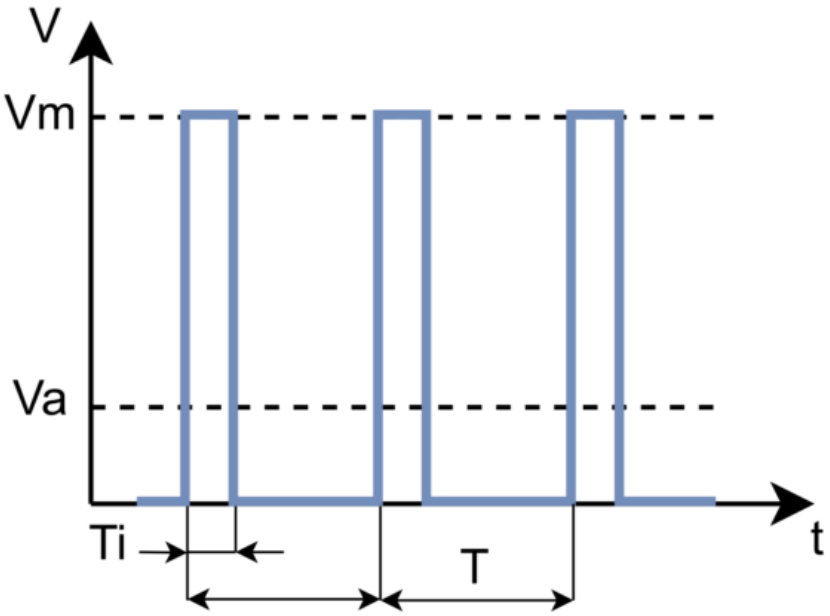

Рисунок 1.20 – График ШИМ

Где, T - период, Ti - ширина импульса, Vm - напряжение сигнала, Va - среднее напряжение

Чем выше частота, тем более незаметны импульсы для человека, например со светодиодом при частоте 10 Гц мы увидим мерцание, а 100 Гц - уже нет. Светодиод загорается практически мгновенно, а вот глаз не способен уловить короткие вспышки, он их "интегрирует" из-за инертности зрения. Если подавать ШИМ на нагреватель, то можно регулировать мощность нагрева.

Ширина импульса

Ширина или длительность импульса (Ti на рисунке 1.20) измеряется в секундах в диапазоне от 0 до периода колебаний T. Чем длиннее импульс, тем выше среднее напряжение сигнала (Va на графике) - оно пропорционально отношению длины импульса к периоду. Например период 1с, импульс 0.25с, среднее напряжение сигнала при питании 5 В (Vm на рисунке 1.20) будет равно:

$$V_a = V_m \times \frac{Ti}{T} = 1.25 \text{ В}$$

Отношение длины импульса к периоду (Ti / T) также называется коэффициентом заполнения ШИМ - это более удобная безразмерная величина, которая уже не зависит от частоты и периода - имеет значение от 0.0 до 1.0 (или заполнение ШИМ 0.. 100%). Чтобы получить результирующее напряжение, нужно просто умножить коэффициент заполнения на напряжение питания ШИМ. Величина, обратная заполнению, называется скважностью ШИМ - на практике используется редко.

Аппаратный ШИМ

У многих МК есть возможность генерировать ШИМ аппаратно - при помощи периферии, вообще не используя ресурсы процессора. Обычно это делается при помощи аппаратного таймера - универсального блока, который может засекать время и самостоятельно управлять пином.

Диапазон значений: 0–255, где 0 — 0% заполнения (выход LOW), а 255 — 100% заполнения (выход HIGH).

Функция analogWrite(pin, value), где

pin — номер цифрового пина с поддержкой ШИМ (обозначается ~ на плате).

value — значение от 0 до 255, задающее коэффициент заполнения

3. Пояснения к работе

Работа выполняется в программном обеспечении Arduino IDE. Для выполнения работы необходимо выполнить задание, соответствующее варианту пользуясь информацией из раздела кратких теоретических сведений к лабораторной работе.

Варианты заданий для индивидуального выполнения представлены в таблице 4.

Таблица 4

1 вариант
Необходимо реализовать управление светодиодом при помощи кнопки и ШИМ. Первая кнопка должна поэтапно увеличивать яркость светодиода на фиксированное значение, например на 50 единиц, пока не будет достигнута максимальная яркость. Вторая кнопка должна поэтапно уменьшать яркость на те же 50 единиц, пока светодиод полностью не погаснет.
2 вариант
Необходимо реализовать управление светодиодом при помощи кнопки и ШИМ. При каждом нажатии кнопки яркость светодиода должна переключаться по циклу: сначала светодиод выключен, затем горит на половину яркости, потом на полную яркость, после чего снова гаснет.

3 вариант
Необходимо реализовать управление двумя светодиодами при помощи кнопки и ШИМ. При нажатии кнопки один светодиод должен постепенно увеличивать яркость, а второй — одновременно уменьшать её, создавая эффект «противоположного» свечения.
4 вариант
Нужно подключить три светодиода и одну кнопку, чтобы при каждом нажатии запускался эффект «бегущего огня». При этом светодиоды должны поочерёдно плавно загораться и гаснуть при помощи ШИМ, создавая эффект движения света слева направо и обратно.
5 вариант
Создать с использованием ШИМ режим плавного мигания одного светодиода, который постепенно загорается и гаснет. При нажатии кнопки этот процесс должен приостанавливаться, и светодиод фиксируется на текущем уровне яркости до следующего нажатия.
6 вариант
Первая кнопка запускает плавное мигание первого светодиода при помощи ШИМ, вторая кнопка — второго. Если нажать обе кнопки одновременно, светодиоды должны мигать в противофазе, то есть когда один загорается, второй гаснет.
7 вариант
Необходимо реализовать управление тремя светодиодами при помощи двух кнопок и ШИМ. При нажатии первой кнопки уровень увеличивается: загорается сначала один светодиод, затем два, затем три. При нажатии второй кнопки уровень уменьшается в обратном порядке.

4. Программа работы

1. Подключите 1–3 светодиода к пинам Arduino, поддерживающим ШИМ (например, D9, D10, D11).
2. Подключите одну или две кнопки к цифровым входам с подтягивающим резистором.
3. В функции setup() настройте пины светодиодов как выходы, а кнопки — как входы.
4. Реализуйте управление яркостью светодиодов через analogWrite(), изменяя значение ШИМ по нажатию кнопок.

5. Содержание отчета

5. В отчете представить результаты выполнения программы работы, анализ работы и выводов.

6. Контрольные вопросы

1. Как работает широтно-импульсная модуляция?
2. В каком диапазоне значений работает функция analogWrite()?
3. Как преобразовать результат analogRead() (0–1023) в диапазон ШИМ (0–255)?
4. Почему ШИМ позволяет управлять яркостью светодиода?
5. На каких пинах Arduino Uno реализован ШИМ?

Основная и дополнительная литература

Основная литература:

1. Блум, Дж. Изучаем Arduino: инструменты и методы технического волшебства / Дж. Блум; пер. с англ. – Санкт-Петербург: БХВ-Петербург, 2015. – 336 с.

2. Монк, С. Программируем Arduino: профессиональная работа со скетчами / С. Монк. – Санкт-Петербург: Питер, 2017. – 272 с.

3. Петин, В. А. Электроника. Проекты с использованием контроллера Arduino / В. А. Петин. – Санкт-Петербург: БХВ-Петербург, 2014. – 400 с.

4. 1. Сажнев, А. М. Цифровые устройства и микропроцессоры : учебное пособие для вузов / А. М. Сажнев. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2022. — 139 с. — (Высшее образование). — ISBN 978-5-534-10883-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: [Электронный ресурс]: — Режим доступа: https://urait.ru/bcode/492264. (дата обращения 10.08.2025)

5. Харрис, Д. М. Цифровая схемотехника и архитектура компьютера RISC-V / Д. М. Харрис, С. Л. Харрис ; под редакцией А. Ю. Романова ; перевод с английского В. С. Яценкова. — Москва : ДМК Пресс, 2022. — 810 с. — ISBN 978-5-97060-961-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: [Электронный ресурс]: — Режим доступа: https://e.lanbook.com/book/241166. (дата обращения 11.08.2025)

Дополнительная литература:

1. Ревич, Ю. В. Занимательная электроника / Ю. В. Ревич. – Санкт-Петербург: БХВ-Петербург, 2018. – 672 с.

2. Wiki.Amperka: распиновка Arduino Uno: сайт / Amperka. – Москва: Amperka, 2010–2025. – URL:
https://wiki.amperka.ru/products%3Aarduino-uno (дата обращения: 18.08.2025). – Режим доступа: свободный.

3. Фоторезистор. Проект 13: сайт / Arduino-kit.ru. – Санкт-Петербург: Arduino-Kit, 2010–2025. – URL:
https://arduino-kit.ru/blogs/blog/project_13 (дата обращения: 18.08.2025). – Режим доступа: свободный.

4. Потенциометр и Arduino: сайт / Edurobots.org. – Москва: Edurobots, 2014–2025. – URL:
https://edurobots.org/2014/04/arduino-potenciometr (дата обращения: 18.08.2025). – Режим доступа: свободный.

5. Резистор: сайт / Wikipedia. – Сан-Франциско: Wikimedia Foundation, 2001–2025. – URL:
https://ru.wikipedia.org/wiki/Резистор (дата обращения: 18.08.2025). – Режим доступа: свободный.