Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

К.Н. Афонин, В.И. Туев

ПРОГРАММИРОВАНИЕ ЭЛЕКТРОННЫХ УСТРОЙСТВ НА ПЛАТФОРМЕ «ARDUINO»

Методические указания к практическим (лабораторным) занятиям
для студентов всех направлений и уровней подготовки

Томск
2022

УДК 004.432.2
ББК 32.973.2
А 94

Рецензент:
Несмелова Н.Н., доцент кафедры радиоэлектронных технологий и экологического мониторинга, канд. биол. наук

Афонин, Кирилл Нильевич, Туев, Василий Иванович

А 94 Программирование электронных устройств на платформе «Arduino»: методические указания к практическим (лабораторным) занятиям для студентов всех направлений и уровней подготовки / К.Н. Афонин, В.И. Туев. – Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2022. – 19 с.

В методическом пособии рассмотрены основы схемотехнического проектирования и программирования электронных устройств на платформе «Arduino». На примере трёх проектов студенты познакомятся с функциями, чаще всего применяемыми в языке программирования платформы.

Учебно-методическое пособие предназначено для студентов всех направлений и уровней подготовки.

Одобрено на заседании кафедры РЭТЭМ протокол № 81 от 19.12.2022.

УДК 004.432.2
ББК 32.973.2

Оглавление

1 АППАРАТНАЯ ЧАСТЬ ПЛАТФОРМЫ ARDUINO

Arduino – это «*open-source*» платформа для моделирования программируемых электронных устройств. В понятие ардуино входят торговая марка, аппаратная платформа, среда разработки и язык программирования. Все это позволяет желающим без лишних усилий создавать свои проекты в области автоматики, робототехники и интернета вещей (*IoT).*

Аппаратная часть (рисунок 1) представляет собой плату с микроконтроллером и минимальной необходимой обвязкой (обычно это кварцевый генератор, стабилизатор напряжения, и обвязка с выводами *GPIO*). В разных моделях используются разные микроконтроллеры, разные форм-факторы плат и разная обвязка.

Главной особенностью *Arduino* является использование прошитого в микроконтроллер загрузчика (*bootloader*), который позволяет программировать микроконтроллер без программатора (что еще больше удешевляет проектирование). Загрузка осуществляется через *USB*-провод. Платформа представляет особый интерес из-за возможности использовать платы расширения (*shield*) – это платы для подключения без пайки к ардуино отдельных устройств (*Ethernet, Bluetooth* контроллеров). Они спроектированы по модульному принципу и устанавливаются в разъемы выводов микроконтроллера.

Рисунок 1 – *Arduino Uno*

Дальнейшее знакомство с ардуино будет проходить на примере платы Arduino Uno, которая содержит:

- микроконтроллер ATmega328 с обвязкой;
- 14 контактов цифрового ввода-вывода (0-13) которые могут выдавать два состояния сигнала: 0В и 5 В или принимать два состояния: есть напряжение (1), нет напряжения (0) (между выбранным выводом и выводом «земли» – GND);
- выводы могут быть как входами, так и выходами, что определяется программой;
- контакты аналогового выхода (3, 5, 6, 9, 10 и 11) – могут передавать сигнал в диапазоне от 0 до 5 В при помощи широтно-импульсной модуляции;

- 6 контактов аналогового входа (0–5). Эти отдельные контакты для аналогового входа получают аналоговые значения (величину напряжения на датчике в диапазоне от 0 до 5 В) и преобразовывают их в цифры от 0 до 1023;
- *USB*-порт служит для загрузки программы в память микроконтроллера и передачи данных между микроконтроллером и ПК (если такая передача предполагается вашей программой). Кроме того, по *USB*-порту передается питание на плату.

Таблица 1 – Технические характеристики платы

Рабочее напряжение	5 В
Входное напряжение (рекомендуемое)	7-12 В
Входное напряжение (предельное)	6-20 В
Цифровые Входы/Выходы	14 (6 из которых могут использоваться как выходы ШИМ)
Аналоговые входы	6
Постоянный ток через вход/выход	40 мА
Постоянный ток для вывода 3.3 В	50 мА
Флэш-память	32 Кб (ATmega328) из которых 0.5 Кб используются для загрузчика
ОЗУ	2 Кб (ATmega328)
EEPROM	1 Кб (ATmega328)
Тактовая частота	16 МГц

Микроконтроллер работает не только с информацией, но и с реальным миром. И главный канал взаимодействия – электрический ток. Работая с током нужно понимать основы электротехники (в том числе и техники безопасности).

Электрический ток – это направленное движение носителей заряда (заряженных частиц: электронов, ионов, катионов и др.).

Электрический ток течет через проводники при наличии разности потенциалов (напряжения) между концами проводника.

Ток бывает постоянным и переменным. В электрической сети используют переменный ток с напряжением 220 В и частотой 50 Гц. Ардуино работает на постоянном токе 5 В. Отсюда вывод – нельзя напрямую подключать плату к электрической сети.

Основные характеристики электрического тока:

I [А] – сила тока – количество заряда, проходящего через сечение проводника за единицу времени.

U [В] – напряжение или разность потенциалов (потенциальных энергий единичного заряда) между двумя точками электрической цепи.

R [Ом] – сопротивление; величина, характеризующая свойства проводника препятствовать прохождению электрического тока.

P [Вт] – мощность; величина, характеризующая скорость передачи или преобразования электрической энергии.

Для работы с электричеством стоит постоянно помнить о нескольких физических законах.

Закон Ома – ток в участке цепи пропорционален разности напряжений на концах участка и обратно пропорционален сопротивлению на участке цепи:

$$\mathbf{I=U/R} \quad (1)$$

Мощность, выделяемая на участке цепи пропорциональна произведению тока и напряжения:

$$P=U*I \quad (2)$$

При последовательном соединении проводников:

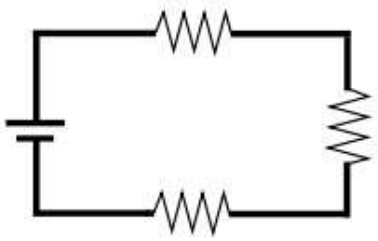


Рисунок 2 – Последовательное соединение

$$I=I1=I2=I3\ldots \quad (3)$$

$$U=U1+U2+U3+\ldots \quad (4)$$

$$R=R1+R2+R3+\ldots \quad (5)$$

При параллельном соединении:

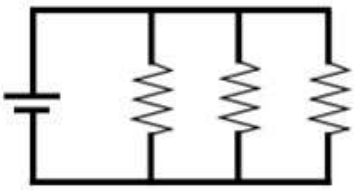


Рисунок 3 – Параллельное соединение

$$I=I1+I2+I3+\ldots \quad (6)$$

$$U=U1=U2=U3 \quad (7)$$

$$1/R=1/R1+1/R2+1/R3 \quad (8)$$

Интерфейс ввода/вывода общего назначения (*general-purpose input/output, GPIO*)

Это выводы микроконтроллера (рисунок 4), которые могут использоваться для обеспечения цифрового управления внешними устройствами. Контакт (пин) может быть сконфигурирован как ввод, так и как вывод. Эти интерфейсы являются основным каналом связи ардуино с внешним миром. Посредством электрических сигналов можно управлять эффекторами и считывать данные с сенсоров.

Рисунок 4 – Выводы микроконтроллера на плате *Arduino*

При работе с пинами стоит помнить, что большая часть из них работает в двоичной логике. То есть он может быть либо включенным, либо выключенным. В случае ввода – различать два состояния – напряжение и его отсутствие.

Но часть пинов ардуино имеют встроенный аналогово-цифровой преобразователь и могут использоваться как аналоговые вводы. На плате они обозначены символом «A».

Цифро-аналоговый преобразователь на ардуино не предусмотрен, но в качестве альтернативы используются выводы, поддерживающие широтно-импульсную модуляцию (ШИМ, отмечены на плате «~»). ШИМ-сигнал за счет быстрого чередования высокого и низкого напряжения с заданным соотношением позволяет управлять аналоговыми устройствами.

Макетная плата

Макетная плата (*breadboard*) – это беспаечная плата для монтажа (рисунок 5). Универсальный инструмент для моделирования прототипов устройств. Монтажную плату применяют для конструирования, отладки и тестирования будущей схемы устройства при разных условиях подключения и эксплуатации. С помощью этого приспособления также проверяют новые детали и компоненты. **Обратите внимание** на рисунок 6, цветные линии, проведённые поверх круглых отверстий, показывают электрические соединения между этими отверстиями.

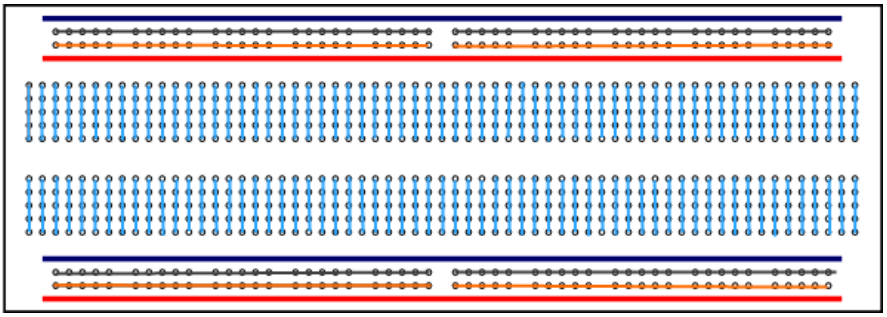


Рисунок 5 – Схема соединений контактов в макетной плате

2 ЯЗЫК ПРОГРАММИРОВАНИЯ ARDUINO

Среда программирования Arduino основана на языке Си++. В языке **Arduino в программе обязательно должны быть две функции: *setup* и *loop***. Пустая программа для Arduino выглядит так:

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Рисунок 6 – Шаблон программы в среде Arduino

Содержимое функции ***setup()*** выполняется сразу при включении или перезагрузке платы. Затем выполняется функция ***loop()***, которая циклически повторяет выполнение кода, записанного внутри функции.

Команды для работы с цифровыми пинами

pinMode (pin, mode) – установка пина на режим входа/выхода. **pin** – номер настраиваемого пина, **mode** – режим (**INPUT** – вход, **OUTPUT** – выход);

digitalWrite (pin, value) – вывод сигнала на указанный пин (**pin), value** – состояние сигнала (**HIGH** – высокий уровень 5 В, **LOW** – низкий уровень 0 В);

digitalRead (pin) – считывает значение с заданного входа (**pin**) – **HIGH** (на пин подано около 5В) или **LOW** (на пин подано низкое напряжение)

При работе с цифровыми пинами стоит помнить:

- Если контакт (*pin*) был установлен в режим вход (*INPUT*), то функция *digitalWrite* со значением *HIGH* будет активировать внутренний 20K нагрузочный резистор. Этот вариант можно использовать для подключения кнопок, однако рекомендуется использовать дополнительный параметр к функции *digitalWrite*, который будет рассмотрен во втором проекте. Подача LOW в свою очередь отключает этот резистор.
- Аналоговые входы (*analog pins*) могут быть использованы как цифровые вход/выходы (*digital pins*). Обращение к ним идет по номерам от 14 (для аналогового входа 0) до 19 (для аналогового входа 5).
- Пин 13 сложнее использовать как цифровой вход, так как он имеет встроенный в плату резистор и светодиод.
- Если вход не подключен, то *digitalRead* может возвращать значения *HIGH* или *LOW* случайным образом.

Аналоговые пины работают схожим образом:

analogRead (pin) – считывает значение с указанного аналогового входа **pin** – напряжение от 0 до 5 В преобразуется в целое число от 0 до 1023 (шаг 0.0049 В);

analogWrite (pin, value) – выдает аналоговую величину (ШИМ волну) на порт вход/выхода; **pin** – номер аналогового выхода, **value** – период рабочего цикла, значение между 0 (полностью выключено, соответствует 0 В) и 255 (сигнал подан постоянно, соответствует 5 В)

При работе с аналоговыми сигналами стоит учитывать:

- ШИМ сигнал не является аналоговым в классическом понимании. ШИМ представляет собой цифровые импульсы с регулируемой длительностью, но для многих применений он может заменить аналоговый сигнал;
- считывание значение с аналогового входа занимает примерно 100 микросекунд (0.0001 сек), т.е. максимальная частота считывания приблизительно 10 000 раз в секунду;
- аналоговый сигнал считывается с 10 битной точностью (1024 значения), а записывается с 8 битной (256 значений).

В ардуино нет встроенных часов, но за счет счетчика импульсов есть функции, позволяющие работать со временем.

millis ( ) – возвращает время с момента запуска скетча в миллисекундах. Переменная типа unsigned long переполняется примерно за 50 дней.

micros ( ) – возвращает количество микросекунд с момента запуска скетча (переполняется за 70 мин.).

delay (ms) – приостанавливает работу программы на указанное количество миллисекунд.

3 Проект «Подключение светодиода»

Рассмотрим схему подключения, приведённую на рисунке 7. На макетной плате последовательно включены в электрическую цепь светодиод и резистор. Чёрный провод подключён к выводу *GND* («земля», «минус») платы *Arduino* и к катоду («–») светодиода, а его анод («+») через резистор соединён красным проводом с цифровым контактом 2 платы *Arduino*. Определить катод и анод светодиода поможет рисунок 8, б.

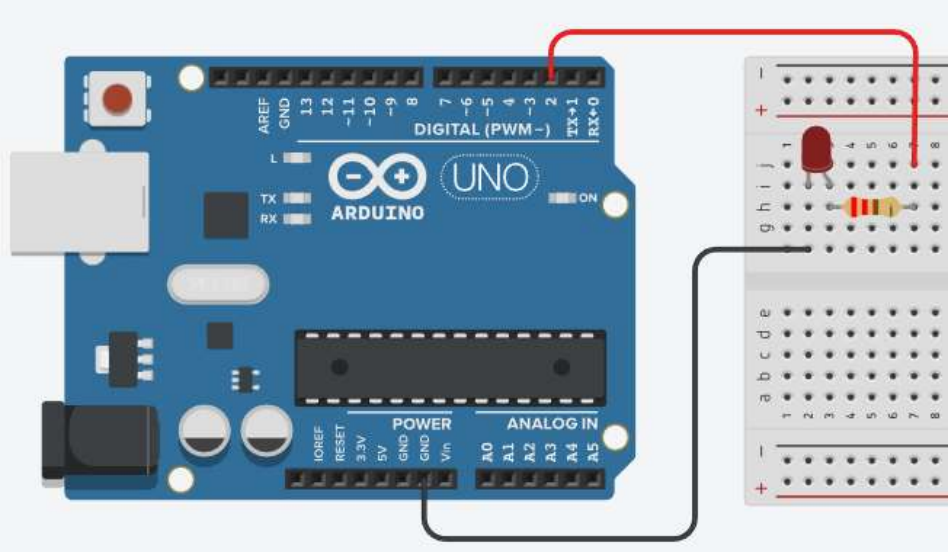
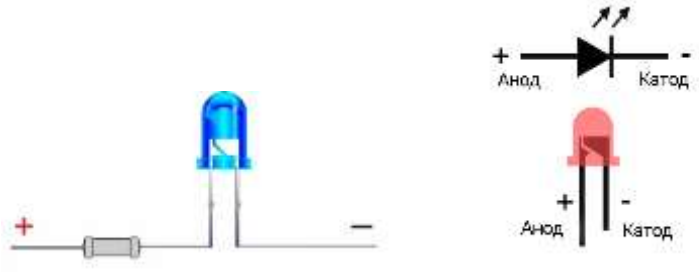


Рисунок 7 – Схема проекта «Подключение светодиода»



а б

Рисунок 8 – Подключение светодиода

Почему необходимо подключать светодиод через резистор? У платформы Arduino значение рабочего напряжения 5 В, значение постоянного тока на каждый вывод не более 40 мА (см. таблицу 1). Рабочее значение тока и напряжения для индикаторного светодиода составляет в среднем 20 мА и 2 В, соответственно. Для разных моделей светодиодов эти значения могут различаться, всегда смотрите справочную документацию на изделие. Превышение данных значений приведёт к выходу из строя светодиода.

Рассчитать значение сопротивления резистора можно воспользовавшись законом Ома. Рассмотрим следующий пример (рисунок 9).

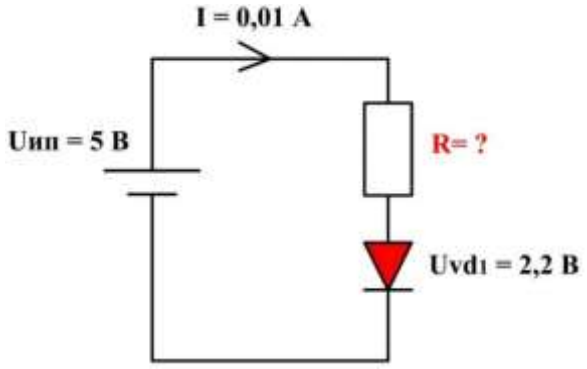


Рисунок 9 – Расчёт резистора

Определим сопротивление R для светодиода VD1 при Uип = 5 В. Для расчета величины сопротивления, согласно закону Ома, нужно знать ток и напряжение: R = U / I. Величина тока, протекающего в цепи и в том числе через VD1 нам известна из заданного условия I_{VD1} = 0,01 А, поэтому следует определить падение напряжения на R. Оно равно разности подведенного напряжения Uип = 5 В и падения напряжения на светодиоде U_{VD1} = 2,2 В:

$$U_R = Uип – U_{VD1} = 5 – 2,2 = 2,8 В$$

Теперь находим R:

$$R = U_R / I_{VD1} = 2,8 В / 0,01 А = 280 Ом$$

Задание. Аналогичным образом определите сопротивление резистора для следующих параметров:

1) Uип = 5 В; I_{VD1} = 0,02 А; U_{VD1} = 3,5 В.
2) Uип = 12 В; I_{VD1} = 0,01 А; U_{VD1} = 2,2 В.
3) Обратная задача, определить I_{VD1} при Uип = 5 В; U_{VD1} = 2 В; R = 220 Ом.

Программа для управления светодиодом

Рассмотрим два варианта программ, представленных на рисунке 10. Содержимое функции ***setup*** выполняется сразу при включении или перезагрузке платы. Затем постоянно повторяясь выполняется функция ***loop***.

На рисунке *10, а* в строке 3 функция **pinMode (2, OUTPUT)** устанавливает вывод 2 в качестве выхода. Следующая в строке 4 функция **digitalWrite (2, HIGH)** устанавливает состояние вывода в положение HIGH, что означает подачу напряжение 5 В на вывод 2. Блок функции ***loop*** пуст, поэтому программа заканчивает своё выполнение.

После загрузки этой программы в память микроконтроллера она будет выполняться при подключении платформы *Arduino* к питанию. В результате выполнения программы светодиод будет всегда светиться.

```
1  void setup()
2  {
3    pinMode(2, OUTPUT);
4    digitalWrite(2, HIGH);
5  }
6
7  void loop()
8  {
9
10 }
```

а

```
1  void setup()
2  {
3    pinMode(2, OUTPUT);
4  }
5
6  void loop()
7  {
8    digitalWrite(2, HIGH);
9    delay(250);
10   digitalWrite(2, LOW);
11   delay(250);
12 }
```

б

Рисунок 10 – Листинг программ

Теперь рассмотрим листинг программы на рисунке *10, б*. Аналогично с предыдущей программой в 3 строке (блок setup) устанавливаем назначение вывода 2 в качестве выхода. Остальная часть программы записана в функции *loop*. В 8 строке также подаём 5 В на вывод 2, как и в первой программе. Затем с помощью функции **delay (250)** делаем паузу в выполнении программы на 250 миллисекунд. С помощью этой функции можно создавать временные задержки в простых проектах. Далее в 10 строке программы устанавливаем состояние вывода 2 в положение LOW, что снимает поданное ранее напряжение 5 В и устанавливает его значение в 0 В. После чего формируем задержку в выполнении программы с помощью функции **delay (250)**. Затем код функции loop будет выполняться снова и снова от начала (восьмая строка) до конца (одиннадцатая строка) пока питание не будет отключено. Таким образом, светодиод, подключённый к выводу 2 будет постоянно мигать.

Обычно блок **setup** используют для настройки состояния выводов (вход или выход) и кода, который должен быть выполнен только один раз после включения микроконтроллера. Основную часть программы пишут в блоке **loop** для бесперебойного функционирования программируемого устройства.

Практическое задание на макетах

1) Повторить проект «Подключение светодиода» с двумя вариантами программы.
2) Добавить на макетную плату второй светодиод и через резистор подключить к любому цифровому выводу кроме 13.
3) Написать программу так, чтобы светодиоды постоянно мигали синхронно через 1 с; 0,5 с; 0,2 с.
4) Написать программу так, чтобы светодиоды постоянно мигали один за другим через 1 с; 0,5 с; 0,2 с.
5) Написать программу так, чтобы светодиоды светились попеременно 2,5 с.

4 Проект «Управление светодиодами кнопкой»

Рассмотрим схему подключения (рисунок 11). Катоды светодиодов подключены к выводу «GND» через общий провод. Аноды светодиодов через резистор подключены к выводам 6 и 7. Кнопка подключена одной стороной к выводу «GND» через общий провод, а другой – к выводу 7.

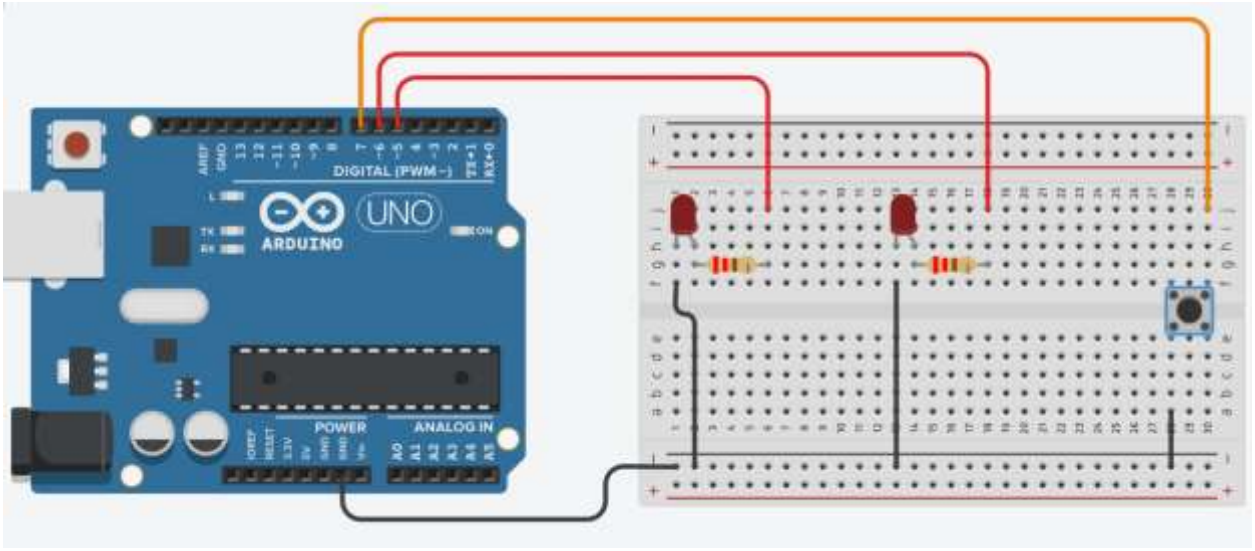


Рисунок 11 – Схема проекта «Управление светодиодами кнопкой»

Программа для проекта

Рассмотрим программу, представленную на рисунке 12.

```
1  void setup()
2  {
3    pinMode(5, OUTPUT);
4    pinMode(6, OUTPUT);
5    pinMode(7, INPUT_PULLUP);
6  }
7
8  void loop()
9  {
10   if (digitalRead(7) == 1)
11   {
12     digitalWrite(6, LOW);
13     digitalWrite(5, HIGH);
14     delay(150);
15   }
16   else
17   {
18     digitalWrite(5, LOW);
19     digitalWrite(6, HIGH);
20     delay(150);
21   }
22 }
```

Рисунок 12 – Листинг программы

В строках 3, 4 определяем выводы 5 и 6 в качестве выхода. Следующая в строке 5 функция **pinMode (7, INPUT_PULLUP)** устанавливает вывод 7 как вход с высоким потенциалом. PULLUP означает «подтягивающий» резистор. Он встроен в микроконтроллер, имеет номинал порядка 20 КОм и используется для «подтягивания» потенциала к выводу. Применение внутреннего подтягивающего резистора предпочтительно, так как такой вариант имеет значительную помехоустойчивость, поскольку для изменения высокого потенциала на

низкий, вывод необходимо напрямую соединить с землей или общим проводом. Обратите внимание, что подтягивающий резистор еще защищает цепь от короткого замыкания при нажатой кнопке.

В блоке кода функции *loop* мы видим условный оператор **if**. В условии оператора функция **digitalRead(7)** считывает значение с указанного цифрового вывода, HIGH или LOW, 1 или 0 соответственно. Это значит, что функция проверяет подано ли напряжение на указанный вывод (7 в данном примере) или нет. Затем она возвращает 1 или 0 в то место, откуда была вызвана и далее проверяется условие в операторе *if*. Далее выполнение кода будет зависеть от того, нажата ли кнопка. Если кнопка не нажата, то потенциал на выводе 7 остаётся в положении HIGH (что есть 1) и включается светодиод на выводе 5 (**digitalWrite (5, HIGH)**), а светодиод на выводе 6 выключается. И наоборот.

Для чего нужна задержка при использовании кнопки?

Существует такое явление как «дребезг контактов». Дребезг контактов – явление, происходящее в электромеханических коммутационных устройствах и аппаратах (кнопках, реле, герконах, переключателях, контакторах, магнитных пускателях и др.), длящееся некоторое время после замыкания электрических контактов. После замыкания происходят многократные неконтролируемые замыкания и размыкания контактов за счёт упругости материалов и деталей контактной системы – некоторое время контакты отскакивают друг от друга при соударениях, размыкая и замыкая электрическую цепь. В зависимости от размеров, массы, материала и конструкции контактной системы время дребезга (время от первого соприкосновения до затухания механических колебаний и установления стабильного контакта) составляет 0,5–2 мс у миниатюрных герконов и до сотен миллисекунд у мощных контакторов.

Поэтому, для программного устранения этого эффекта мы можем использовать задержку в выполнении кода. Попробуйте выполнить этот код без использования задержки, и вы поймёте, как это выглядит.

Практическое задание на макетах

1) Повторить проект «Управление светодиодом с кнопкой».
2) Определить минимальное время задержки, необходимое для устранения «дребезга контактов» и продемонстрировать преподавателю два варианта работы макета – с «дребезгом контактов» и без него.
3) Написать программу так, чтобы светодиоды постоянно мигали синхронно через 0,25 с. При удержании кнопки один светодиод продолжает мигать, второй светится постоянно.
4) Написать программу так, чтобы светодиоды постоянно дважды мигали поочерёдно по 0,2 с. При удержании кнопки оба светодиода однократно мигают поочерёдно по 0,5 с.
5) Написать программу так, чтобы светодиоды светились поочерёдно при однократном нажатии на кнопку (переключение светодиодов).

5 Проект «Управление яркостью светодиода»

Широтно-импульсная модуляция

В современной преобразовательной технике преимущественно используются импульсное регулирование мощности на нагрузке. Одним из способов реализации импульсного регулирования является широтно-импульсная модуляция (ШИМ). В англоязычной литературе PWM – pulse-width modulation. Основными элементами любого типа импульсного регулятора мощности являются полупроводниковые ключи – транзисторы или тиристоры. В простейшем виде схема импульсного источника питания имеет следующий вид (рисунок 13). Источник постоянного напряжения Uип с ключом К подсоединяется к нагрузке Н. Ключ К переключается с определенной частотой и остается во включенном состоянии определенную длительность времени. С целью упрощения схемы на ней не изображены другие обязательные элементы. В данном контексте нас интересует только работа ключа К.

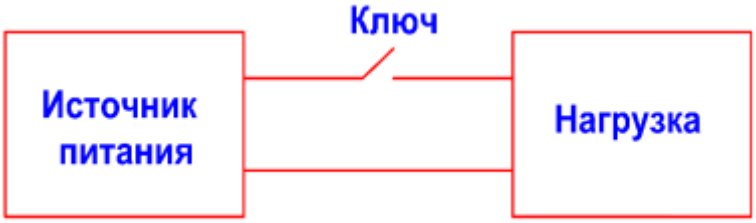


Рисунок 13 – Упрощённая схема импульсного питания

Чтобы понять принцип ШИМ воспользуемся следующим графиком (рисунок 14). Разобьем ось времени на равные промежутки, называемые периодом Т. Теперь, например, половину периода мы будем замыкать ключ К. Когда ключ замкнут, к нагрузке Н подается напряжение от источника питания Uип. Вторую часть полупериода ключ находится в закрытом состоянии, а нагрузка останется без питания.

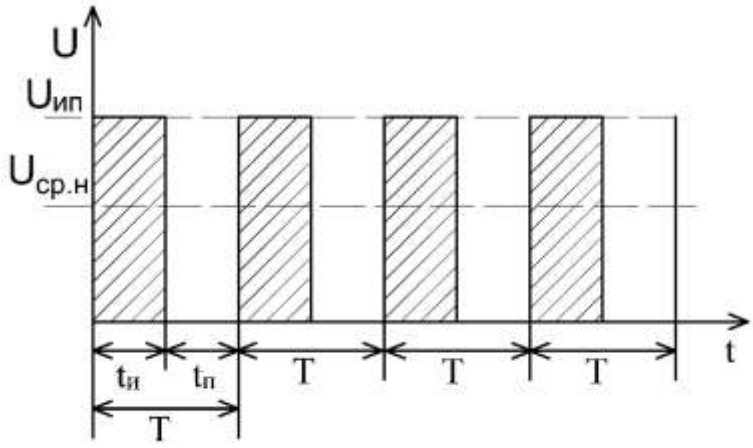


Рисунок 14 – График зависимости подаваемого напряжения от времени

Время, в течение которого ключ замкнут, называется временем импульса tи. А время длительности разомкнутого ключа называют временем паузы tп. Если измерить напряжение на нагрузке, то оно будет равно половине Uип. Среднее значение напряжения на нагрузке можно выразить следующей зависимостью:

$$U_{ср.н} = U_{ип} \times t_и / T \quad (9)$$

Отношение времени импульса tи к периоду T называют коэффициентом заполнения D. А величина, обратная ему называется скважностью S:

$$S = 1/D = T/tи \quad (10)$$

На практике удобнее пользоваться коэффициентом заполнения, который зачастую выражают в процентах. Когда транзистор полностью открыт на протяжении всего времени, то коэффициент заполнения D равен единице или 100 %.

Если D = 50 %, то это означает, что половину времени за период транзистор находится в открытом состоянии, а половину в закрытом. В таком случае форма сигнала называется меандр.

Следовательно, изменяя коэффициент D от 0 до единицы или до 100 % можно изменять величину Uср.н от 0 до Uип:

$$Uср.н = Uип \times D \quad (11)$$

Рассмотрим схему подключения на рисунке 15.

К выводу «~3» платы *Arduino* через резистор подключён анод светодиода, катод соединён с контактом «GND». Переменный резистор (потенциометр) имеет три вывода – два для подключения «+» и «–», один управляющий контакт. Для привычного (интуитивного) регулирования параметра, т. е. в крайнем левом положении вращательной рукоятки регулируемый параметр имеет минимальное значение, а в крайнем правом наоборот – максимальное значение, необходимо крайний левый контакт потенциометра подключить к контакту «GND», крайний правый – к выводу «5V», центральный контакт подключается к одному из шести аналоговых входов платы, например, «A5».

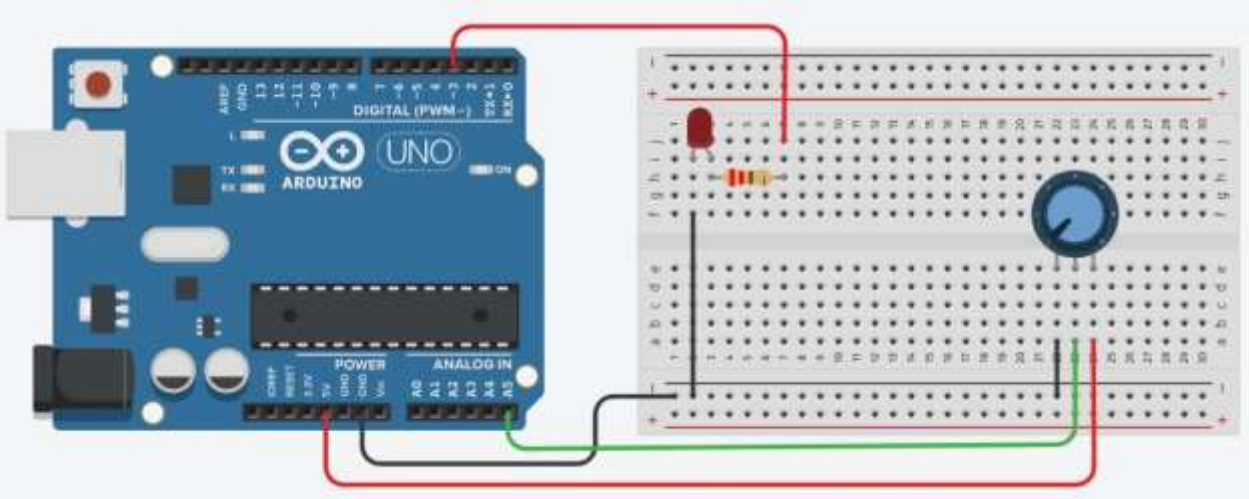


Рисунок 15 – Схема проекта «Управление яркостью светодиода»

Рассмотрим текст программы для проекта (рисунок 16).

В первой строке создадим и инициализируем глобальную переменную *n*. Эта переменная будет хранить значения, считываемые с аналогового входа «A5», определённого в строке 5 программы функцией **pinMode**. В строке 6 новая функция **Serial.begin(9600)**. Функция используется для связи между платой *Arduino* и компьютером или другими устройствами. Все платы Arduino имеют как минимум один последовательный порт (также известный как UART или USART). Использование такой записи активирует эту возможность

для нашего проекта. Число **9600** устанавливает скорость передачи данных в битах в секунду для последовательной передачи данных (9600 – стандартное значение).

```
1  int n = 0;
2  void setup()
3  {
4    pinMode(3, OUTPUT);
5    pinMode(A5, INPUT);
6    Serial.begin(9600);
7  }
8
9  void loop()
10 {
11   n = analogRead(A5);
12   n = map(n, 0, 1023, 0, 255);
13   Serial.println(n);
14   analogWrite(3, n);
15 }
16
```

Рисунок 16 – Листинг программы для проекта «Управление яркостью светодиода»

В строке 11 в управляющую переменную *n* запишем значение, считываемое с аналогового входа «A5» с помощью функции **analogRead(A5)**. Далее необходимо вспомнить особенности работы микроконтроллера из раздела 1: считывание и запись аналогового сигнала происходит с разной точностью – аналоговый сигнал считывается с 10 битной точностью (1024 значения), а записывается с 8 битной (256 значений). Поэтому, для корректной передачи аналогового сигнала, необходимо привести в один размерный диапазон считываемый и записываемый сигнал. Следующая функция **map(n, 0, 1023, 0, 255)** преобразует число из одного диапазона в другой. Синтаксис функции следующий (аргументы функции):

1) указывается преобразуемая переменная (в нашем случае *n*);
2) минимальное и максимальное значения исходного диапазона через запятую (в нашем случае 0, 1023);
3) минимальное и максимальное значения диапазона, в который необходимо выполнить преобразование (в нашем случае 0, 255).

В строке 13 использована функция **Serial.println(n)**. Она выводит данные на последовательный порт в виде текста ASCII, за которым следует переход на новую строку. В нашем проекте выводится значение переменной *n*. Посмотреть актуальные значения выводимых переменных можно в окне «Мониторинг последовательного порта» отмеченного специальным значком (рисунок 17).

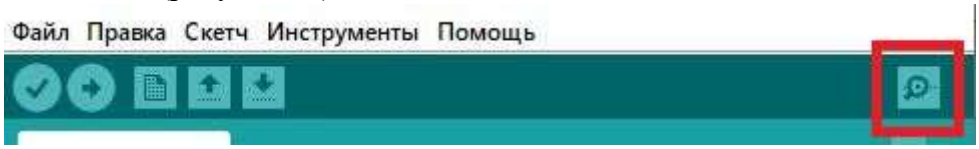


Рисунок 17 – Расположение кнопки «Мониторинг порта» на панели инструментов

В следующей строке (14) функция **analogWrite(3, n)** записывает аналоговое значение (волну ШИМ) на контакт 3. После вызова analogWrite() вывод будет генерировать устойчивую прямоугольную волну с указанным рабочим циклом до следующего вызова analogWrite() (или вызова digitalRead() или digitalWrite()) на том же выводе.

Практическое задание на макетах

1) Повторить проект «Управление яркостью светодиода».

2) Написать программу управления временем задержки мигающего светодиода с помощью потенциометра. В крайнем левом положении – мигает часто (задержка 100), в крайнем правом – редко (задержка 1500).

3) Подключить RGB-светодиод вместо одиночного светодиода с резистором и кнопку. Обратите внимание, на плате RGB-светодиода уже установлены необходимые резисторы. Проверить схему подключения у преподавателя.

4) Написать программу управления яркостью каждого цвета светодиода. Переключение между цветами осуществлять нажатием кнопки. Далее смешайте цвета так, чтобы получить фиолетовый, жёлтый и белый свет.

Список использованных источников

1. Смирнов, Ю. А. Основы микроэлектроники и микропроцессорной техники : учебное пособие / Ю. А. Смирнов, С. В. Соколов, Е. В. Титов. — 2-е изд., испр. — Санкт-Петербург : Лань, 2022. — 496 с. — ISBN 978-5-8114-1379-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/211292 (дата обращения: 26.05.2022). — Режим доступа: для авториз. пользователей.