Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Севастопольский государственный университет»
Институт радиоэлектроники и информационной безопасности

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

к практикуму по дисциплине
«Программируемые устройства на платформе Arduino»

для студентов очной и заочной форм обучения всех направлений

Севастополь
2021

Учебно-методическое пособие к практикуму по дисциплине «Программируемые устройства на платформе Arduino» для студентов очной и заочной форм обучения всех направлений / СевГУ ; сост. М. А. Дурманов. — Севастополь : Изд-во СевГУ, 2021. — ___ с.

Целью методических указаний является оказание помощи студентам при выполнении практических заданий по дисциплине «Программируемые устройства на платформе Arduino».

Рассмотрены и утверждены на заседании кафедры «Радиоэлектроника и телекоммуникации» «___» 2021 года, протокол № __.

Рецензент: канд. техн. наук, доцент кафедры «Радиоэлектроника и телекоммуникации» Савочкин А. А.

Ответственный за выпуск: доктор техн. наук, профессор, зав. кафедрой «Радиоэлектроника и телекоммуникации» Афонин И. Л.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дисциплина «Программируемые устройства на платформе Arduino» изучается студентами очной и заочной форм обучения разных направлений во втором, третьем и четвертом семестрах и является дисциплиной общеуниверситетского пула.

При изучении дисциплины студенты выполняют четыре практические работы, которые направлены на получение практических навыков программирования микроконтроллеров на платформе Arduino с использованием онлайн симулятора TinkerCAD и интегрированной среды разработки (ИСР) Arduino IDE и составления электрических схем, а также знакомство с основными компонентами радиоэлектронной аппаратуры. Итоговой формой контроля является зачет.

После изучения дисциплины «Программируемые устройства на платформе Arduino» студент должен знать: особенности работы микроконтроллера, средства программирования микроконтроллеров в ИСР Arduino, принцип работы основных радиокомпонентов.

В результате изучения данной дисциплины студент должен уметь: выбирать схемотехнические и алгоритмические методы, необходимые для решения поставленной задачи, оптимизировать код программы с точки зрения эффективности его работы.

Выполнение практических работ осуществляется фронтальным методом каждым студентом индивидуально в соответствии с вариантом задания, который выдается преподавателем.

На выполнение практических работ отводится по четыре часа. Выполняются практические работы в следующем порядке:

— студенты самостоятельно должны проработать по конспекту лекций и рекомендованной литературе, приведенной в библиографическом списке, основные теоретические положения практической работы;

— собрать электрическую схему согласно заданию, написать скетч и загрузить его в микроконтроллер;

— продемонстрировать результаты работы скетча преподавателю;

— сделать снимки схемы в процессе работы;

— оформить отчет по практической работе;

— защитить отчет по практической работе.

Выполнять и защищать практические работы студенты должны строго по графику, в соответствии с расписанием учебных занятий.

В отчет должны включаться: титульный лист, цель работы, текст задания, теоретические сведения, текст программы с комментариями и результаты выполнения программы в виде фотографии. Содержание отчета приведено в учебно-методическом пособии к каждой практической работе.

На титульном листе обязательно требуется указать фамилию, имя и отчество полностью, группу, номер варианта, название и номер работы. Образец оформления титульного листа приведен в приложении А.

Отчет по практической работе оформляется каждым студентом индивидуально на стандартных листах формата А4 с использованием персонального компьютера (ПК) и сдаются преподавателю в электронном виде. По краям листа должны быть оставлены поля: левое — 25 мм; верхнее и нижнее — 20 мм; правое — 15 мм.

Отчет следует оформлять в соответствии с методическими указаниями по оформлению текстовых работ [1].

Рисунки и таблицы должны быть пронумерованы и сопровождаться подписями и пояснениями.

При выполнении практических работ и подготовке к защите рекомендуется использовать учебную и справочную литературу [2 — 11].

1. ПРАКТИЧЕСКАЯ РАБОТА № 1

ПРОГРАММИРОВАНИЕ ЦИФРОВЫХ И АНАЛОГОВЫХ ВЫВОДОВ

1.1. Цель работы

Приобрести практические навыки работы с платформой Arduino Uno и в ИСР Arduino IDE.

Сформировать практические навыки составления простейших электрических схем.

1.2. Теоретические сведения

1.2.1. Интерфейс ИСР Arduino IDE

Среда разработки Arduino IDE представляет собой текстовый процессор. Окно IDE делится на три основные области: область управления, область ввода текста и область вывода сообщений (рис. 1.1).

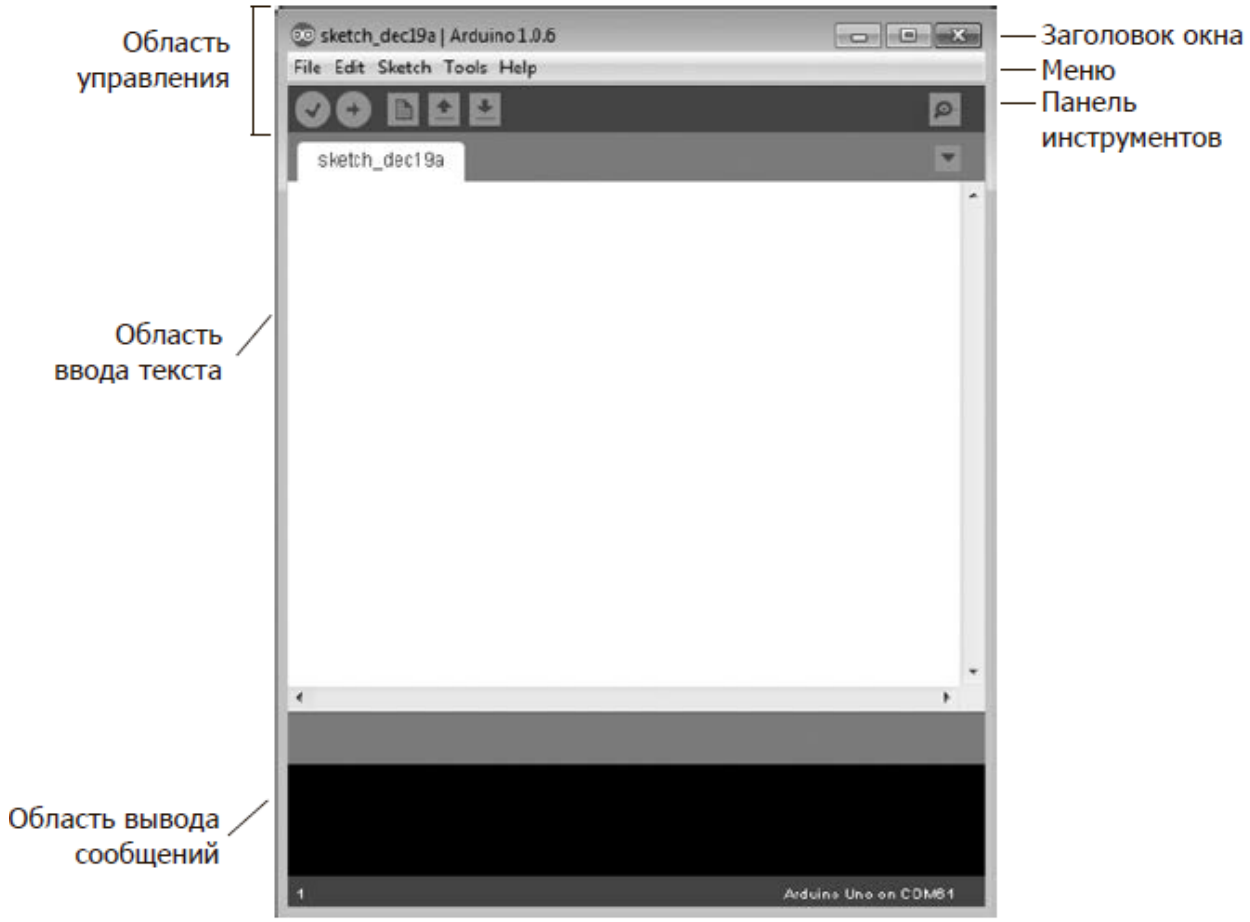


Рис. 1.1 — Окно среды разработки Arduino IDE

Область управления содержит заголовок окна, меню и панель инструментов. В **заголовке** окна отображается имя файла скетча и версия IDE. Ниже находится **Главное меню**: File (Файл), Edit (Правка), Sketch (Скетч), Tools (Ин-

струменты) и Help (Помощь) и панель инструментов с пиктограммами.

Разделы главного меню Arduino IDE имеют следующее назначение:

— **File** содержит команды для сохранения, загрузки и печати скетчей; набор примеров скетчей, которые можно открыть, а также подменю Preferences (Настройки);

— **Edit** содержит команды копирования, вставки и поиска;

— **Sketch** содержит команду для проверки скетча перед выгрузкой в плату, команду доступа к папке со скетчем и команды импортирования;

— **Tools** содержит множество различных команд, а также команды для выбора типа платы Arduino и порта USB;

— **Help** содержит ссылки на разные темы и версию IDE.

Панель инструментов включает в себя шесть пиктограмм, значение которых кратко описано ниже в порядке слева направо:

— **Verify** (Проверить) запускает проверку скетча на корректность и отсутствие программных ошибок;

— **Upload** (Загрузить) запускает проверку скетча и последующую его выгрузку в плату Arduino;

— **New** (Новый) открывает новый пустой скетч в новом окне;

— **Open** (Открыть) открывает ранее сохраненный скетч;

— **Save** (Сохранить) сохраняет открытый скетч. Если скетч еще не имеет имени, вам будет предложено ввести его;

— **Serial Monitor** (Монитор порта) открывает новое окно для обмена данными между платой Arduino и IDE.

Область ввода текста служит для ввода исходного кода скетчей. Имя текущего скетча отображается во вкладке вверху слева над областью ввода текста. (По умолчанию скетчу присваивается имя, содержащее текущую дату.) В этой области вводится исходный код скетча, как в обычном текстовом редакторе.

Область вывода сообщений имеет вид прямоугольника черного цвета в нижней части окна IDE. В ней выводятся самые разные сообщения, включая результаты проверки скетчей, успешность выгрузки в плату и др.

Справа внизу под областью вывода сообщений отображается тип подключенной платы Arduino и порт USB, к которому она подключена.

1.2.2. Понятие об электрической цепи

Электрическая цепь — совокупность устройств, элементов, предназначенных для протекания электрического тока**.** Простейшая электрическая цепь состоит из источника электрической энергии, ее потребителя и соединительных проводов. Любая замкнутая электрическая цепь делится на две части: внеш-

нюю, называемую внешним участком цепи, и внутреннюю, называемую внутренним участком цепи.

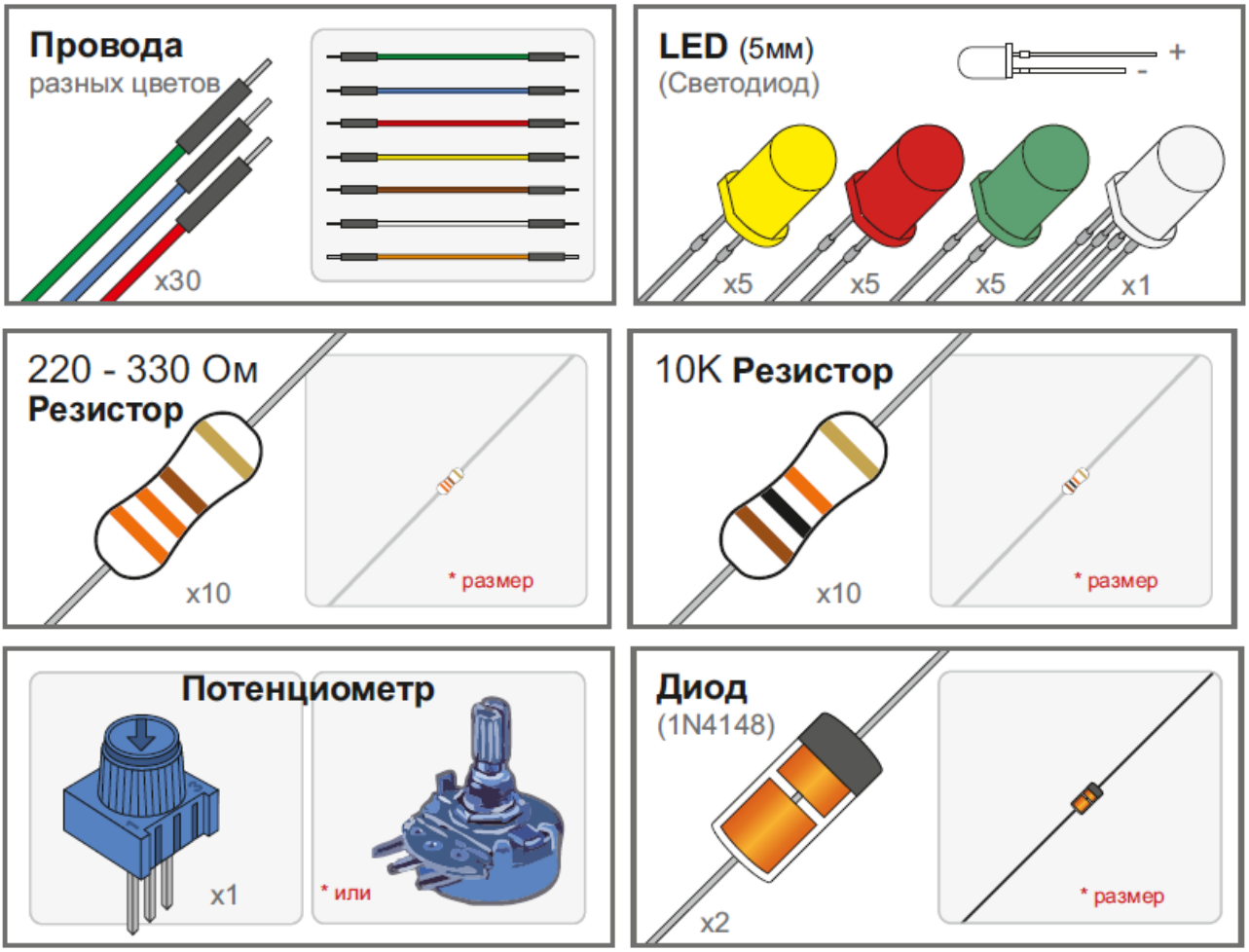
Внешний участок (внешняя цепь) состоит из одного или нескольких потребителей электрической энергии, соединительных проводов и различных приборов, включенных в эту цепь. Внутренний участок (внутренняя цепь) представляет собой сам источник электрической энергии.

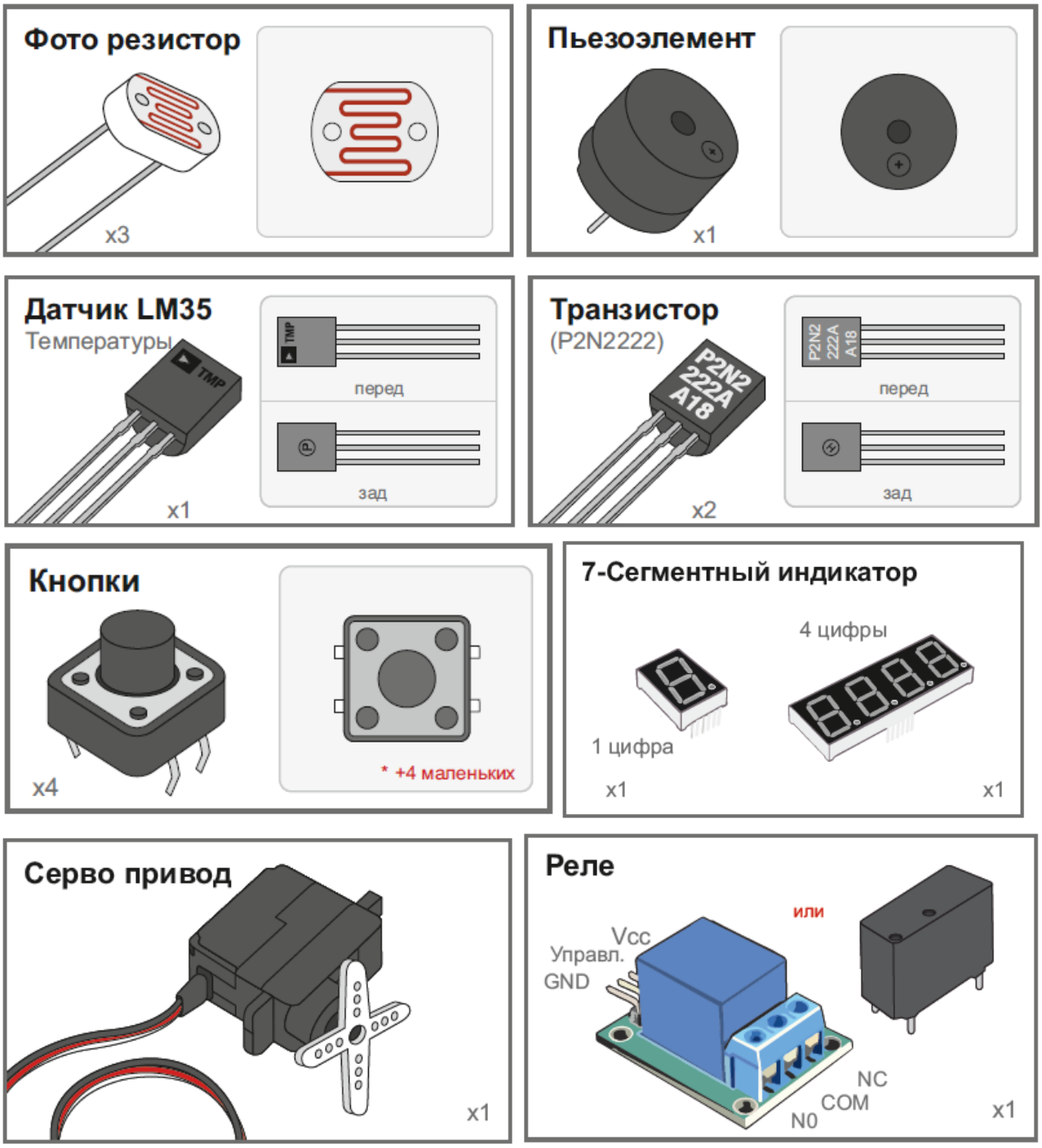
Источниками электрической энергии для питания радиотехнической аппаратуры служат гальванические элементы, аккумуляторы, генераторы и т. д.

Потребителями электрической энергии в электро- и радиотехнических устройствах являются электродвигатели, сельсины, реле, электронно-лучевые трубки, дискретные элементы (резисторы, диоды, транзисторы ...), интегральные схемы и т.п.

1.2.3. Компонентная база радиоэлектронных устройств

1.2.3.1. Основные компоненты электронных схем





1.2.3.2. Плата Arduino Uno

Arduino — это открытая аппаратная платформа для макетирования электронных устройств, основанная на гибком и простом в использовании аппаратном и программном обеспечении.

Плата Arduino UNO (рис. 1.2) имеет разъем подключения универсальной последовательной шины (Universal Serial Bus, USB) (2). С его помощью плата подключается к компьютеру, это позволяет подать на нее напряжение питания, выгрузить скетч с инструкциями и отправить или принять данные с компьютера.

В центре находится микроконтроллер, содержащий микропроцессор, выполняющий инструкции, и несколько видов памяти для хранения данных и инструкций и имеющий различные входы и выходы для вывода или ввода данных.

На плате имеются два ряда разъемов, или контактов. В первом ряду (10) находятся контакты электропитания и контакты внешней кнопки сброса. Во втором ряду (10) — шесть контактов аналоговых входов, они используются для измерения уровней напряжения электрических сигналов. Кроме того, контакты A4 и A5 служат для обмена данными с другими устройствами.

Контакты с номерами от 0 до 13 (6) — цифровые входы/выходы. С их помощью можно определять наличие или отсутствие входных электрических сигналов или генерировать выходные сигналы. Контакты с номерами 0 и 1 параллельно подключены к последовательному порту и могут использоваться для обмена данными с другими устройствами, например с компьютером, через схему подключения к разъему USB. Контакты, отмеченные знаком тильды (~), могут также генерировать электрический сигнал переменного напряжения, — это нужно, например, для создания световых эффектов или управления электродвигателями.

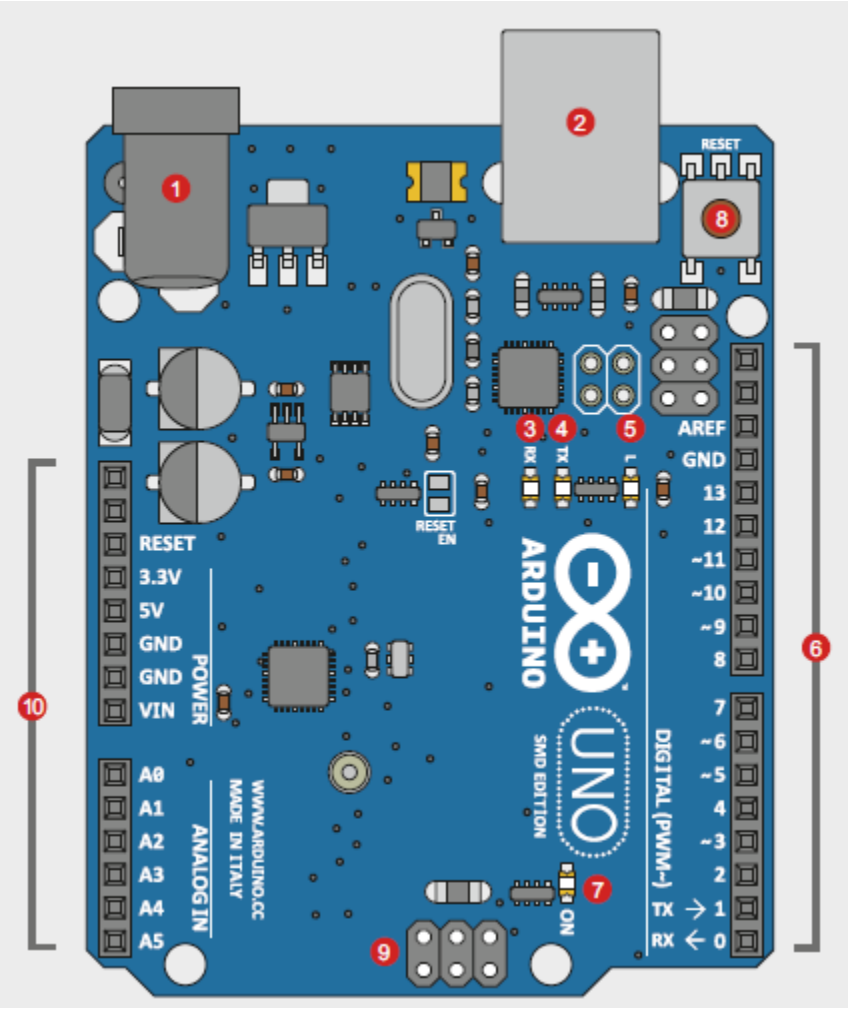


Рис. 1.2 — Внешний вид платы Arduino Uno на SMD компонентах

На плате Arduino имеется четыре светодиода: один, с подписью **ON** (7) служит индикатором подключенного к плате электропитания, светодиоды с подписями **RX** и **TX** (3,4) зажигаются в том случае, когда происходит обмен данными между платой Arduino и подключенными устройствами через последовательный порт и USB. Светодиод L (5) предназначен для нужд пользователя (он подключен к контакту цифрового входа/выхода с номером 13).

Кнопка **RESET** (8) на плате предназначена для перезапуска Arduino.

1.2.3.3. Макетная плата

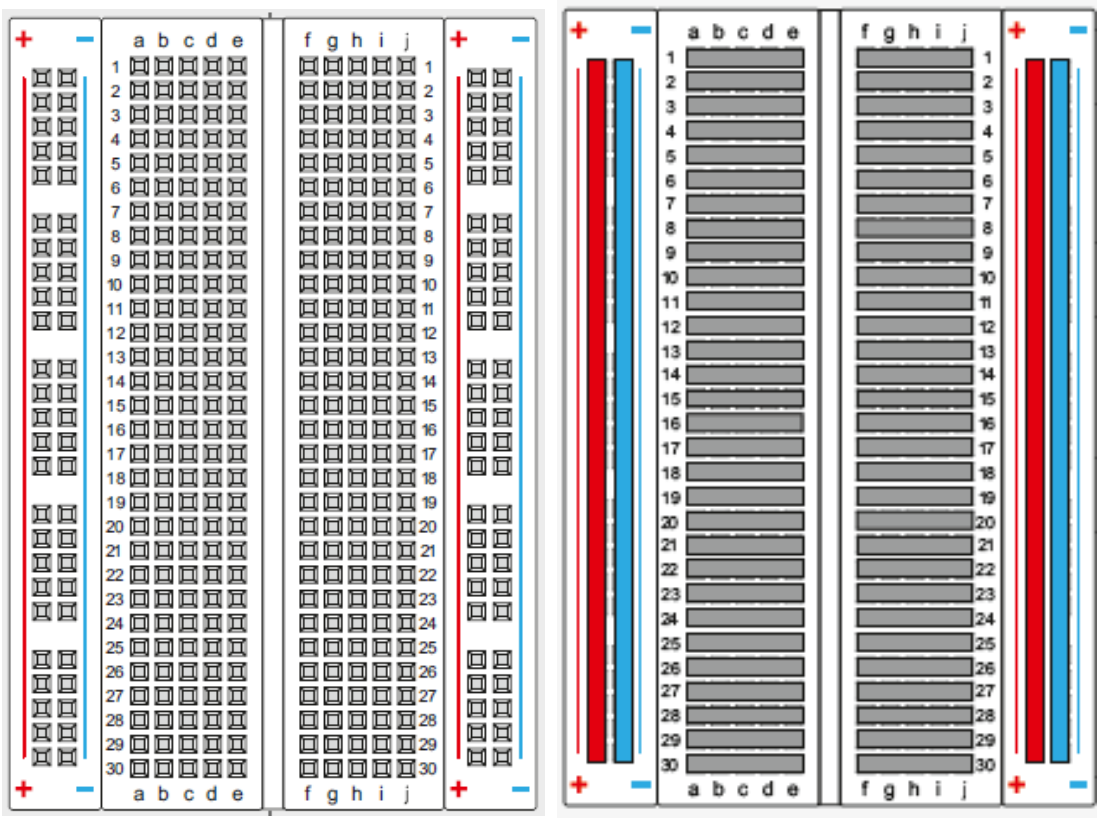


Рис. 1.3 — Внешний вид макетной платы и внутренней схемы соединения контактов

На рис. 1.3 красным цветом показано питание, каждый контакт соединен в любом месте вертикального столбца. Синим цветом показана «земля», каждый контакт соединен в любом месте вертикального столбца. Серым цветом показаны горизонтальные ряды контактов, Каждая строка с 1 по 30 состоят из 5 соединенных между собой ячеек. Электронные компоненты, подсоединенные в ряд из пяти ячеек, будут включены в один узел электрической цепи.

1.2.3.4. Монитор COM-порта

Чтобы открыть монитор порта, щелкните на пиктограмме **Serial Monitor** (Монитор порта) на панели инструментов, изображенной на рис. 1.1. В ответ на

это откроется окно монитора порта, показанное на рис. 1.4.

Рис. 1.4 — Окно монитора последовательного порта

Как показано на рис. 1.4, вверху в окне монитора порта имеется однострочное поле ввода с кнопкой **Send** (Отправить) и область вывода под ним, где отображаются данные, полученные с платы Arduino.

Прежде чем использовать монитор порта, нужно активировать последовательное соединение, добавив следующий вызов в функцию **`void setup():`**

```
Serial.begin(9600);
```

где **`9600`** — это скорость передачи данных между компьютером и платой Arduino, измеряемая в бодах (бит/с). Это число должно соответствовать значению, выбранному в раскрывающемся списке справа внизу в окне монитора порта (см. рис. 1.4).

Послать текст для отображения в области вывода в мониторе порта можно вызовом функции **`Serial.print`**:

```
Serial.print("Text Text Text Text");
```

Она отправит в монитор порта текст, заключенный в кавычки. Для вывода текста и принудительного перехода на новую строку следует воспользоваться функцией **`Serial.println:`**

```
Serial.println("Text Text Text Text");
```

Аналогично осуществляется вывод в монитор порта содержимого переменных. Например, ниже показано, как вывести содержимое переменной **`results`**:

```
Serial.println(results);
```

Значения переменных типа float по умолчанию выводятся с двумя знаками после десятичной точки. Количество знаков можно изменить, передав число от 0 до 6 во втором параметре после имени переменной. Например, чтобы вывести

значение вещественной переменной **results** с четырьмя десятичными знаками, следует выполнить следующий вызов:

```
Serial.print(results,4);
```

1.3. Порядок выполнения работы

1.3.1. Схема с мигающим светодиодом

Для выполнения настоящей и последующих практических работ необходимо использовать доступный онлайн симулятор «TinkerCAD», предоставляемый компанией AutoDesk, в котором производится моделирование работы электрической схемы, построенной на микроконтроллере платформы Arduino. Для использования указанного симулятора требуется произвести регистрацию для доступа в личный кабинет по ссылке: tinkercad.com. После регистрации и входа в учетную запись онлайн среды «TinkerCAD» на панели слева следует выбрать вкладку «Circuits», затем «Создать цепь», далее откроется рабочее пространство симулятора (рис. 1.5). Панель справа содержит набор компонентов и схемы, где следует выбрать: плату «Arduino Uno», светодиод и резистор номиналом 200 Ом (для этого следует кликнуть любой кнопкой мыши по выбранному резистору, после чего откроется панель параметров, и задать требуемый номинал) и собрать схему, показанную на рис. 1.5.

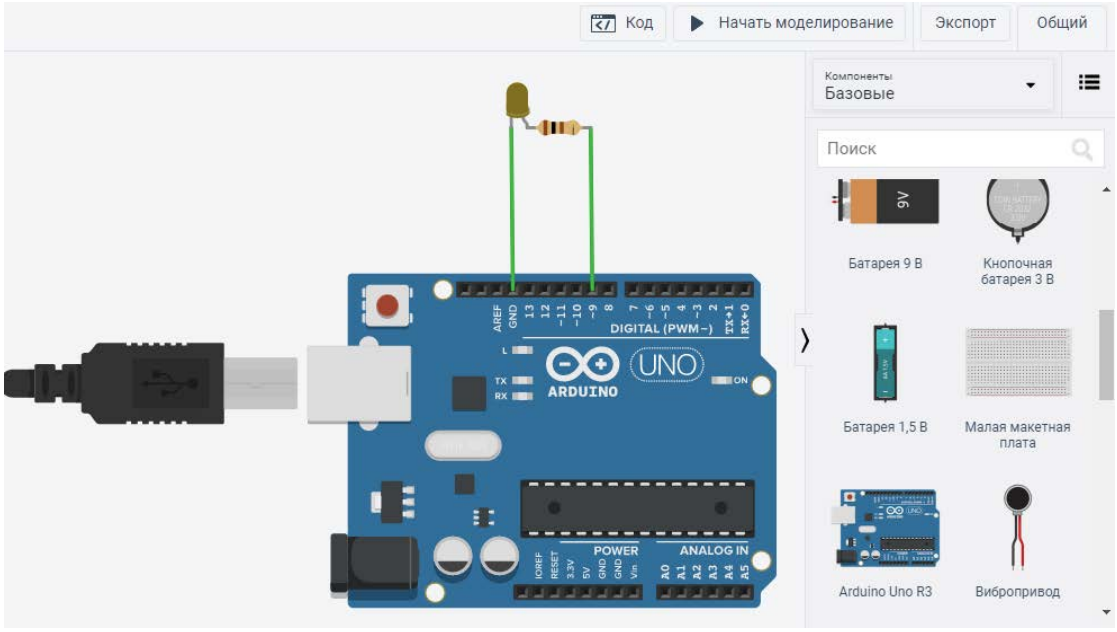


Рис. 1.5 — Рабочее пространство симулятора TinkerCAD

После сборки схемы необходимо написать программу для проведения симуляции. Для этого требуется открыть вкладку «Код» и слева в верхней части выпадающего меню выбрать способ представления кода «Текст» (рис. 1.6).

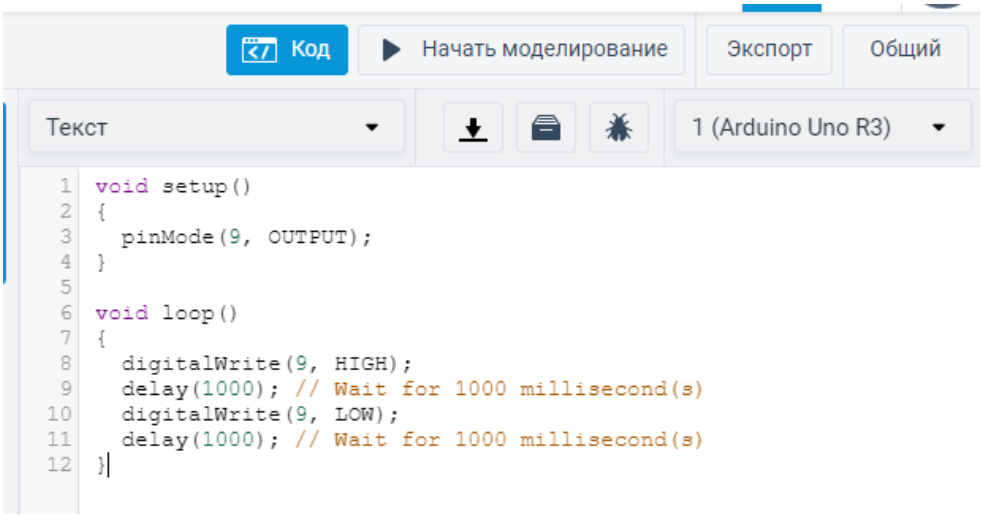


Рис. 1.6 — Поле программного кода онлайн симулятора

Программный код необходимо задействовать в режиме симуляции, нажав кнопку «Начать моделирование», после чего светодиод в собранной схеме должен начать мигать с интервалом 1 с на каждое состояние.

Монитор последовательны порта в онлайн симуляторе полностью аналогичен монитору последовательного порта в ИСР Ardunio IDE. Вызвается из вкладки «Код», нажав на текст «Монитор последовательного интерфейса» (рис. 1.7).

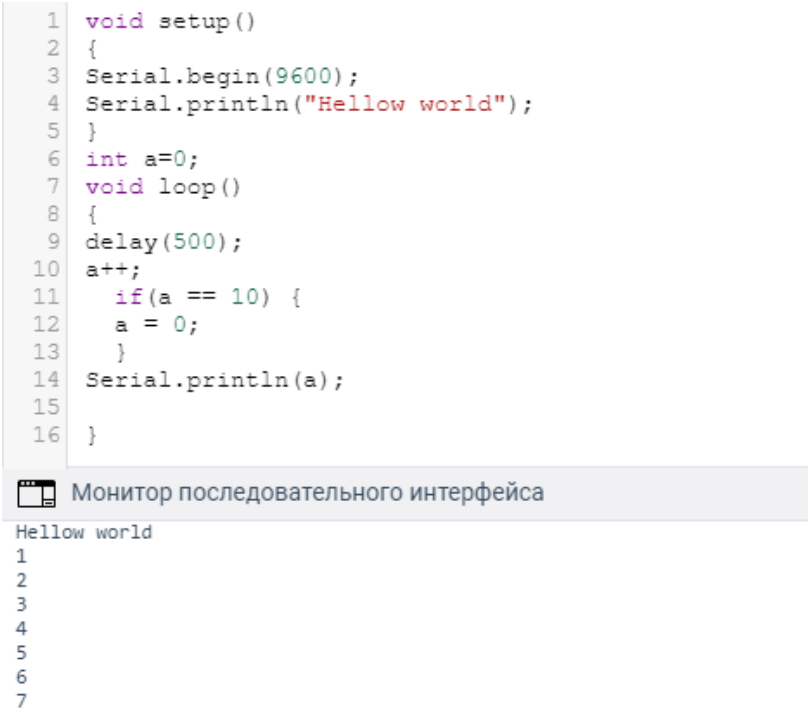


Рис. 1.7 — Пример программы с использованием монитора последовательного порта в TinkerCAD

Для программирования микроконтроллера в Arduino IDE, необходимо подключить плату Arduino к компьютеру с помощью кабеля USB. Затем запустить IDE, выбрать пункт меню **Tools → Serial Port** (Инструменты → Порт) и далее выбрать любой доступный порт кроме COM1. Это позволит убедиться, что плата правильно подключена.

Программу следует начинать с комментария, в котором дается название, назначение и краткое описание программы.

Чтобы добавить однострочный комментарий, введите два слеша (//…) и текст комментария. Среда разработки будет игнорировать тест, следующий за ними, во время проверки скетча.

Чтобы ввести комментарий, занимающий несколько строк, введите символы **/*** в строке, предшествующей комментарию, и символы */ в строке, следующей за комментарием.

Затем необходимо сохранит скетч, выбрав пункт меню **File → Save As** (Файл → Сохранить как...). Введите короткое имя скетча и затем нажмите OK. По умолчанию среда разработки сохраняет скетч в файлах с расширением .ino и добавляет его автоматически.

Любая программа в среде разработке должна содержать две обязательные функции: **`void setup()`** и **`void loop().`**

Функция **`void setup()`** содержит инструкции, которые плата Arduino выполняет только один раз после каждого сброса или включения питания.

Функция **`void loop()`** будет выполняться бесконечно, если не выключить питание или не нажать кнопку RESET (сброс).

Рассмотрим программу для мигания светодиодом, показанную в **Листинге 1.**

```
// Листинг 1. Программа для мигания светодиодом
void setup (){
pinMode(13, OUTPUT); //настроить контакт 13 как цифровой выход
}
void loop(){
digitalWrite(13, HIGH); // включить напряжение на выходе 13
delay(1000); // пауза в одну секунду
digitalWrite(13, LOW); // выключить напряжение на выходе 13
delay(1000); // пауза в одну секунду
}
```

Функция **`digitalWrite()`** управляет напряжением, которое подается на цифровой выход: в данном случае на контакт 13, к которому подключен светодиод L. Второй параметр в вызове этой функции (**`HIGH`**) указывает, что она должна установить «высокий» (high) цифровой уровень; в результате через

контакт и светодиод L потечет электрический ток, и светодиод включится.

После включения светодиода вызовом **`delay(1000)`** выполняется пауза длительностью в 1 секунду. Функция **`delay()`** приостанавливает работу скетча на указанный период времени — в данном случае на 1000 миллисекунд, или 1 секунду.

Затем мы снимаем напряжение со светодиода вызовом **`digitalWrite(13, LOW)`** (**`LOW`** — низкий цифровой уровень), и он выключится. В заключение вызовом **`delay(1000)`** выполняется еще одна пауза в одну секунду, в течение которой светодиод остается выключенным.

Проверка скетча позволяет убедиться, что он не содержит ошибок и понятен плате Arduino. Для проверки законченного скетча щелкните на пиктограмме **Verify** (Проверить) в IDE. Когда проверка скетча будет закончена, в области вывода сообщений должно появиться сообщение «Done compiling» (Компиляция завершена), которое говорит о том, что скетч проверен и готов к выгрузке в плату Arduino.

Убедившись, что скетч был введен без ошибок, сохраните его, проверьте подключение платы Arduino и щелкните на пиктограмме **Upload** (Загрузить) на панели инструментов IDE. В ответ на это среда разработки проверит скетч еще раз и затем загрузит его в плату. В процессе загрузки мигание светодиодов TX/RX на плате покажет, что идет обмен информацией между Arduino и компьютером.

Согласно приведенным ниже вариантам задания составить электрическую схему в симуляторе TinkerCAD и написать скетч для микроконтроллера Arduino Uno. Выполнить проверку работоспособности электрической схемы и скетча с помощью моделирования. Собрать электрическую схему с помощью макетной платы и показать работоспособность схемы преподавателю.

Задание 1

Соберите схему с внешним светодиодом как показано на рис. 1.8.

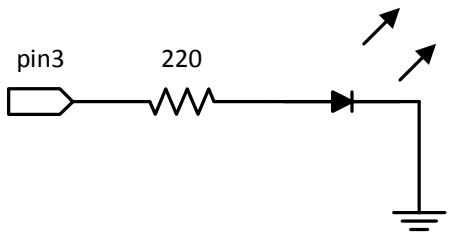


Рис. 1.8 — Схема подключения светодиода

Напишите программу, которая заставляет мигать светодиод с частотой 10/(n+1) Гц, где n — последняя цифра порядкового номера в списке группы.

Длительность одного из состояний светодиода определяется по формуле

$$\tau = \frac{1}{2f},$$

где f — частота мигания.

Продемонстрировать работу схемы преподавателю.

1.3.2. Схема с потенциометром

Переменные резисторы, также известные как потенциометры, позволяют изменять их сопротивление от 0 Ом до соответствующего им номинального значения. На принципиальных схемах переменные резисторы обозначаются, как показано на рис. 1.9.

Переменные резисторы имеют три контакта: один центральный и по одному с каждого конца. По мере вращения штока переменного резистора сопротивление между одним крайним и центральным контактом увеличивается, а между другим крайним и центральным контактом — уменьшается.

На рис. 1.10. показан линейный переменный резистор.

Рис. 1.9 — Обозначение переменного резистора

Рис. 1.10 — Внешний вид переменного резистора

Переменный резистор можно использовать для получения аналогового сигнала. Для этого переменный резистор нужно подключить, как показано на рис. 1.11, и загрузить код программы из Листинга 2.

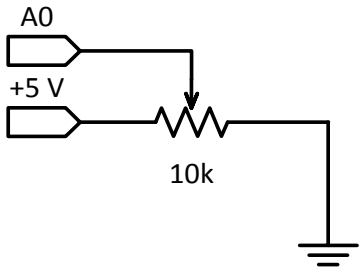


Рис. 1.11 — Схема подключения потенциометра

```
// Листинг 2. Программа чтения данных с потенциометра
const int POT = A0; // Аналоговый вход 0
                    // для подключения потенциометра
```

```
int val = 0; //Переменная для хранения значения с потенциометра
void setup (){
    Serial.begin(9600);
}
void loop(){
    val = analogRead(POT);
    Serial.print("value = " );
    Serial.println(val);
    delay(500);
}
```

В каждой итерации цикла переменная **val** получает аналоговое значение, считанное командой **analogRead()** с входа, соединенного со средним контактом потенциометра (вход A0). Далее это значение функция **Serial.println()** выводит в последовательный порт, соединенный с компьютером. Затем следует задержка в 500 мс (чтобы числа выводились не быстрее, чем вы можете их прочитать).

После запуска монитора последовательного порта на экране компьютера появится окно с отображением потока передаваемых чисел. Если повернуть ручку потенциометра, то выводимые значения будут меняться. Если повернуть ручку в одном направлении, числа приближаются к 0, если в другом — к 1023.

Функция map() для преобразования диапазона

В языке программирования Arduino есть функции для пропорционального преобразования значений от одного диапазона к другому: **map()** и **constrain()**. Синтаксис функции **map()** выглядит следующим образом:

```
output = map(value, fromLow, fromHigh, toLow, toHigh).
```

Здесь **value** — преобразуемое значение (напряжение на аналоговом входе). **fromLow** *и* **fromHigh** — это нижняя и верхняя границы текущего диапазона; **toLow** *и* **toHigh** — нижняя и верхняя границы нового диапазона.

Задание 2

Соберите схему, показанную на рис. 1.11, и выведите на экран монитора COM-порта число 20n.

Напишите программу, которая включает и выключает светодиод, подключенный к цифровому выводу 13, с интервалом от n/10 *с до n*+1 *с. Интервал задавать с помощью потенциометра.*

Продемонстрировать работу схемы преподавателю.

1.3.3. Схема с RGB светодиодом

RGB светодиод — это три светодиода в одном корпусе: красный, зеленый и синий (рис. 1.12). Эти светодиоды, меняя свою яркость, могут переливаться, выдавая различные оттенки и световые эффекты. RGB светодиод, как правило,

имеет четыре вывода: три из них — аноды светодиодов и четвертый — общий катод.

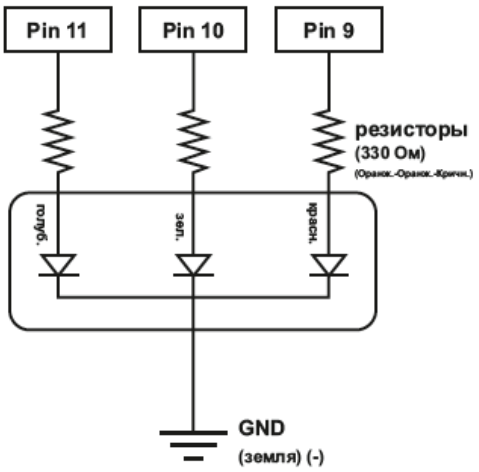


Рис. 1.12 — Схема RGB светодиода

Для изменения яркости светодиода используется широтно-импульсная модуляция (ШИМ). Для создания сигнала с ШИМ используется функция **`analogWrite(x, y),`** где x — номер цифрового выхода, а y — значение коэффициента заполнения, в диапазоне от 0 до 255, где 0 соответствует коэффициенту заполнения 0 %, а 255 — 100 %.

Только цифровые выходы 3, 5, 6, 9, 10 и 11 на плате Arduino UNO поддерживают ШИМ.

В Листинге 3 представлена программа, в которой яркость светодиода изменяется от своего минимального значения до максимального и наоборот.

```
// Листинг 3. Программа для управления яркостью светодиода
void setup (){
pinMode(11, OUTPUT); //настроить контакт 11 как цифровой выход
}
void loop() {
    for(int i=0; i<=255;i++) {
     analogWrite(11, i); // включить напряжение на выходе 13
     delay(10); // пауза в 10 мс
}
for(int i=255; i>=0;i--) {
     analogWrite(11, i); // включить напряжение на выходе 13
     delay(10); // пауза в 10 мс
}
}
```

В Листинге 4 представлена функция, которая зажигает определенным цветом RGB светодиод. Эта функция переводит число от 0 до 767 в определенный цвет. Если пройтись по этому числовому ряду, то светодиод будет плавно менять цвет через весь цветовой спектр: «0» соответствует красному, «255» соот-

ветствует зеленому, «511» синему, «767» опять красному. Числа, находящиеся между указанными выше значениями, дадут возможность отобразить промежуточные цвета. Например, число 640 находится в середине между 512 (синий) и 767 (красный). Это даст смесь синего и красного, в результате чего получится фиолетовый.

```
// Листинг 4. Функция, задающая цвет RGB светодиода
void showRGB(int color){
  int redIntensity;
  int greenIntensity;
  int blueIntensity;
  if (color <= 255)                // зона 1
  { redIntensity = 255 - color;
    greenIntensity = color;
    blueIntensity = 0;
  }
  else if (color <= 511)           // зона 2
  { redIntensity = 0;
    greenIntensity = 255 - (color - 256);
    blueIntensity = (color - 256);
  }
  else if (color >= 512)             // зона 3
  { redIntensity = (color - 512);
    greenIntensity = 0;
    blueIntensity = 255 - (color - 512);
  }
  analogWrite(RED_PIN, redIntensity);
  analogWrite(BLUE_PIN, blueIntensity);
  analogWrite(GREEN_PIN, greenIntensity);
}
```

Задание 3

Соберите схему с RGB светодиодом. Напишите программу, в которой создайте функцию ***void showSpectrum()****, которая будет плавно изменять цвет RGB светодиода по всем цветам радуги. В программе предусмотрите вызов функции* ***void showRGB(int color).***

Продемонстрировать работу схемы преподавателю.

1.3.4. Схема с восьмью светодиодами

К цифровым выводам платы Arduino можно подключить множество светодиодов и управлять их свечением в любой заданной последовательности.

Рассмотрим схему, включающую в себя восемь светодиодов. Каждый светодиод нужно подключать через токоограничивающий резистор, как показано на рис. 1.13.

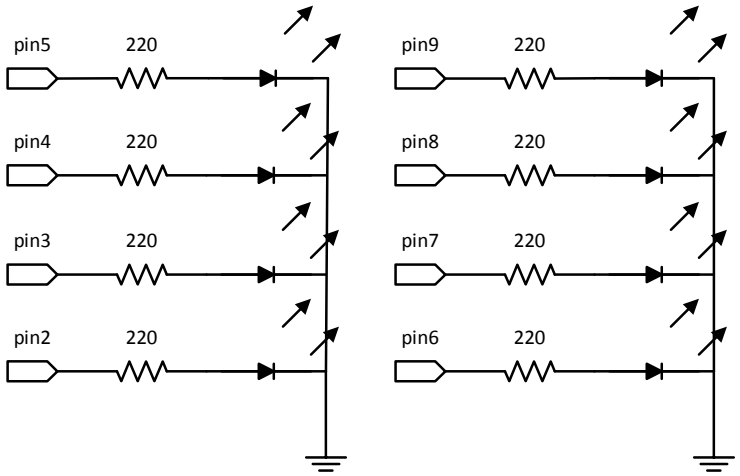


Рис. 1.13 — Схема подключения восьми светодиодов

Чтобы обратиться к любому светодиоду, мы будем использовать массив. Массивы позволяют хранить группу переменных, и обращаться к ним по индексу. Создадим массив из восьми целых чисел, и инициализируем их:

```
int ledPins[] = {2,3,4,5,6,7,8,9};
```

Значения в массиве определяют номера контактов, к которым подключены светодиоды.

Зададим режим работы восьми светодиодов на выход с помощью оператора цикла for( ):

```
int index;
for(index = 0; index <= 7; index++)
  {
    pinMode(ledPins[index],OUTPUT);
  }
```

В Листинге 5 рассмотрена функция **`void oneAfterAnotherLoop()`**, которая включает один светодиод, затем следует небольшая пауза, потом включает следующий светодиод, и так пока не загорятся все. Затем она их выключает в обратном порядке.

```
// Листинг 5. Поочередное включение 8 светодиодов
void oneAfterAnotherLoop()
{ int index;
  int delayTime = 100; // пауза delayTime, миллисек.
//Этот цикл последовательно переключает все светодиоды на вкл.
  for(index = 0; index <= 7; index++) {
    digitalWrite(ledPins[index], HIGH);
    delay(delayTime);
  }
//Этот цикл последовательно переключает все светодиоды на выкл.
  for(index = 7; index >= 0; index--) {
    digitalWrite(ledPins[index], LOW);
    delay(delayTime);
  }
}
```

Задание 4

Соберите схему из восьми светодиодов (см. рис. 1.13). Напишите программу, которая включает и выключает светодиоды в заданной последовательности в соответствии с таблицей 1.1.

Варианты	**Задание**
1	Поочередное включение светодиодов (**`void oneOnAtATime()`**)
2	Поочередное включение светодиодов в одном направлении и потом — в обратном (**`void pingPong()`**)
3,8	Одновременное поочередное включение светодиодов с обоих краев (**`void marquee()`**)
4,9	Одновременное поочередное включение светодиодов от края и от центра (**`void gate()`**)
5,0	Случайное включение светодиодов с случайным интервалом (**`void randomLED()`**)
6	Поочередное включение светодиодов ускоряется (**`void acceleration()`**)

7	Поочередное включение светодиодов замедляется (**`void deceleration()`**)

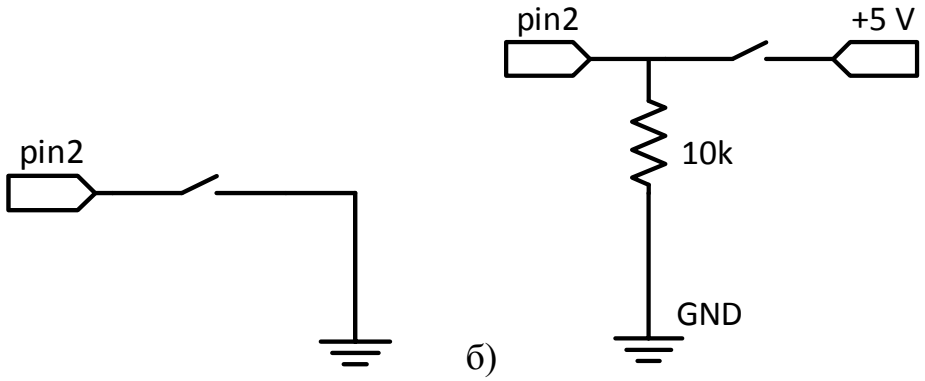
Для включения светодиодов в случайной последовательности необходимо воспользоваться функцией **`random()`**, которая возвращает псевдослучайное число. Вызывается как **`random(min, max)`**, где **`min`** — нижняя граница случайных значений, включительно (опционально); **`max`** — верхняя граница случайных значений. *Возвращаемое значение*: случайное число между **`min`** и **`max - 1`** (тип **`long`**).

Продемонстрировать работу схемы преподавателю.

1.3.5. Схема с кнопками

Рассмотрим работу цифровых выводов в режиме на вход. Простейшим цифровым датчиком является кнопочный переключатель (далее кнопка). С ее помощью можно передавать информацию микроконтроллеру через цифровой порт и управлять схемой.

Встречаются две схемы подключения кнопки к микроконтроллеру: с подтягивающим резистором (рис. 1.14, а) и со стягивающим резистором (рис. 1.14, б).



а) б)

Рис. 1.14 — Схема подключения кнопки с подтягивающим резистором (а) и со стягивающим резистором (б)

Для инициализации контакта, к которому подключается кнопка с подтягивающим резистором, в функции **`void setup()`** необходимо написать команду

```
pinMode(pin, INPUT_PULLUP);
```

Параметр INPUT_PULLUP в функции pinMode() используется для подключения внутреннего подтягивающего резистора номиналом 20 кОм, и по умолчанию на цифровом входе микроконтроллера будет высокий уровень (HIGH).

Для инициализации контакта, к которому подключается кнопка со стягивающим резистором, в функции **`void setup()`** необходимо написать команду

```
pinMode(pin, INPUT);
```

В этом случае по умолчанию на цифровом входе микроконтроллера будет низкий уровень (LOW).

В Листинге 6 приведена программа, которая считывает состояние двух кнопок и в соответствии с полученными значениями включает или выключает светодиод.

```
// Листинг 6. Работа с кнопками
const int button1Pin = 2;  // кнопка №1 - порт 2
const int button2Pin = 3;  // кнопка №2 - порт 3
const int ledPin = 13;     // порт 13, для светодиода

void setup()
{ // Установим порты кнопок на вход:
  pinMode(button1Pin, INPUT);
  pinMode(button2Pin, INPUT);
  // Установим порт светодиода на выход:
  pinMode(ledPin, OUTPUT);
}
void loop()
{ // переменные, в которые записываем состояния кнопок
  int button1State, button2State;
// Читаем текущее состояние кнопок и помещаем их значение
// в две переменные:
  button1State = digitalRead(button1Pin);
  button2State = digitalRead(button2Pin);
// сравниваем, нажата ли одна из кнопок
if (((button1State == LOW) || (button2State == LOW))
&& !
// и если нет сравниваем, нажаты ли обе кнопки
    ((button1State == LOW) && (button2State == LOW)))
    digitalWrite(ledPin, HIGH);  // включаем светодиод
else
    digitalWrite(ledPin, LOW);  // выключаем светодиод
}
```

Задание 5

Варианты	**Задание**
1, 6	С помощью кнопки включить светодиод на 1 с.
2, 7	С помощью кнопки включить светодиод и затем с помощью другой кнопки выключить его.
3, 8	С помощью кнопки включить светодиод и затем с помощью той же кнопки выключить его.
4, 9	С помощью двух кнопок управлять яркостью светодиода (одной кнопкой увеличивать яркость, другой —

	уменьшать)
0, 5	С помощью двух кнопок управлять периодом мигания светодиода (одной кнопкой увеличивать длительность, другой — уменьшать)

Продемонстрировать работу схемы преподавателю.

1.4.Содержание отчета

1.4.1. Сформулировать цель работы.

1.4.2. Привести постановку задачи и текст задания.

1.4.3. Привести краткие теоретические сведения.

1.4.4. Привести принципиальную схему устройства, соответствующую заданию.

1.4.5. Привести текст программы (программа должна содержать комментарии).

1.4.6. Привести фотографию собранного устройства.

1.4.7. Сделать выводы по работе.

1.5. Контрольные вопросы

1.5.1. Опишите состав окна интегрированной среды разработки Arduino IDE.

1.5.2. Раскройте понятие электрической цепи.

1.5.3. Перечислите основные элементы радиоэлектронной аппаратуры и кратко охарактеризуйте их.

1.5.4. Опишите состав платы Arduino UNO.

1.5.5. Для чего и как используется макетная плата?

1.5.6. Опишите работу с монитором последовательного порта.

1.5.7. Основные правила написания скетча в Arduino IDE.

1.5.8. Охарактеризуйте функции **`pinMode()`, `digitalWrite()`, `delay()`.**

1.5.9. Охарактеризуйте функции **`analogRead()`, `analogWrite()`, `random()`.**

1.5.10. Что такое RGB светодиод?

1.5.11. Изобразите схему с подтягивающим резистором и объясните принцип ее работы.

1.5.12. Изобразите схему со стягивающим резистором и объясните принцип ее работы.

2. ПРАКТИЧЕСКАЯ РАБОТА № 2

ПРОГРАММИРОВАНИЕ АНАЛОГОВЫХ ДАТЧИКОВ И ЗВУКА

2.1. Цель работы

Изучить принципы работы аналоговых и цифровых датчиков; приобрести практические навыки управления электрической схемой с помощью датчиков.

2.2. Теоретические сведения

2.2.1. Работа с аналоговыми датчиками

Большинство наблюдаемых явлений вокруг нас имеет аналоговый характер, что предполагает бесконечное число возможных состояний, будь то солнечный свет, или температура океана, или концентрация загрязняющих веществ в воздухе. Данная практическая работа посвящена изучению методов преобразования аналоговых сигналов в цифровые значения, которые могут быть проанализированы микроконтроллером.

На рис. 2.1 показаны графики цифрового и аналогового сигналов. Слева — прямоугольные импульсы, амплитуда которых принимает только два значения: 0 и 5 вольт. Справа изображен фрагмент гармонического сигнала. Несмотря на то, что его амплитуда находится в тех же границах (0 и 5 вольт), аналоговый сигнал принимает бесконечное число значений между этими двумя.

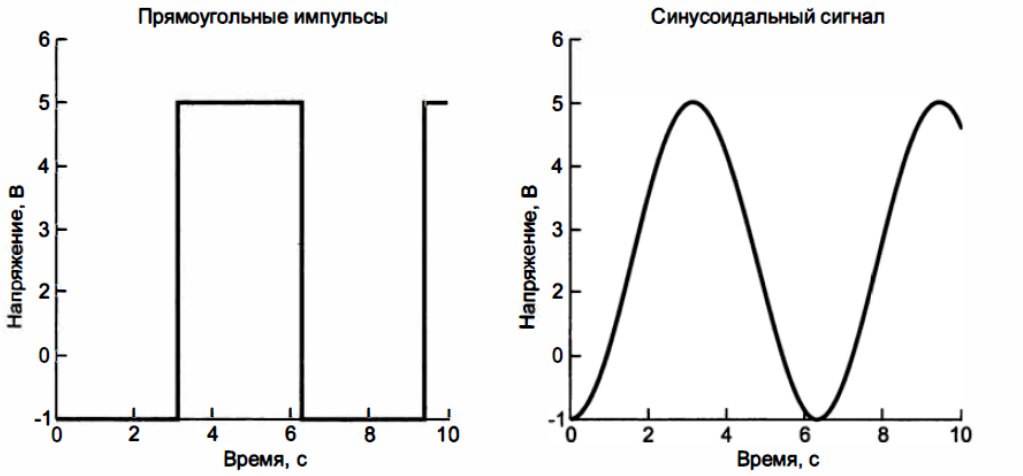


Рис. 2.1 — Вид цифрового и аналогового сигналов

Аналоговые сигналы нельзя представить конечным числом состояний, теоретически они могут иметь бесконечное число значений в пределах некоторого диапазона.

Компьютер не может оперировать с бесконечным числом десятичных разрядов для аналогового значения, потому что объем его памяти и производительность ограничены. Поэтому соединение интерфейса цифрового контролле-

ра Arduino с аналоговым реальным миром возможно благодаря **аналого-цифровому преобразователю (АЦП)**, который преобразует аналоговые значения в цифровые с заданной точностью.

Точность АЦП зависит от его разрядности. На плате Arduino Uno установлен 10-разрядный АЦП. Это означает, что АЦП может разделить аналоговый сигнал на 2^{10} различных значений. Следовательно, Arduino может присвоить $2^{10} = 1024$ аналоговых значений, от 0 до 1023.

Опорное напряжение определяет максимальное напряжение на входе АЦП, его значение соответствует коду 1023. Для опорного напряжения, равного 5 В, при нулевом входном напряжении АЦП выдает на выходе 0, при входном напряжении 2,5 В на выходе будет значение 512 (половина от 1023), при входном напряжении 5 В выходной код равен 1023.

Считывание данных с аналоговых датчиков с помощью Arduino

Для чтения аналоговых значений предусмотрена функция **`analogRead()`**. Самый простой аналоговый датчик, с которого можно получить аналоговый сигнал, — это потенциометр. Их используют в стереосистемах, звуковых колонках, термостатах и в других устройствах. Потенциометры действуют как регулируемые делители напряжения и снабжены ручкой-регулятором. Они бывают разных размеров и форм, но всегда имеют три вывода.

Для подключения потенциометра к Arduino необходимо один крайний вывод потенциометра подключить к земле, а другой к шине 5 В. Потенциометры симметричны, так что не имеет значения, с какой стороны вы подключите шину питания, а с какой — землю. Средний вывод соединяется с аналоговым контактом A0 на плате Arduino. При повороте ручки потенциометра аналоговый входной сигнал будет плавно меняться от 0 до 5 В.

Схема подключения и программа, реализующая считывание с аналогового входа сигнал и вывод значений на монитор последовательного порта, показаны в методических указания к практической работе №1.

Использование переменных резисторов для создания аналоговых датчиков

Благодаря достижениям в области физики, мы имеем множество материалов, способных изменять сопротивление в результате физического воздействия. Например, проводящие краски изменяют свое сопротивление при изгибе и скручивании, полупроводники меняют сопротивление под действием света (фоторезисторы), сопротивление некоторых материалов зависит от нагрева и охлаждения (термисторы).

Это всего лишь несколько примеров, которые позволят создать свои собственные аналоговые датчики. Поскольку упомянутые датчики меняют сопро-

тивление, а не напряжение, в схеме потребуется создать делитель напряжения, чтобы можно было измерить изменение сопротивления.

Резистивный делитель напряжения состоит из двух резисторов, от соотношения сопротивлений которых зависит выходное напряжение. Так, если один из резисторов переменный, то на выходе можно получить изменение напряжения. Другой резистор определяет чувствительность схемы, если это подстроечный резистор, то чувствительность можно корректировать.

Предположим, что один из этих резисторов переменный, например фоторезистор (рис. 2.2). Сопротивление фоторезистора зависит от интенсивности падающего на него света. Номинальное сопротивление фоторезистора равно 200 кОм. В полной темноте его сопротивление около 200 кОм, при ярком свете оно падает почти до нуля. От того, в какое плечо поставить фоторезистор, и от номинала постоянного резистора будет зависеть масштаб и точность показаний.

Рис. 2.2 — Внешний вид фоторезистора

Можно поэкспериментировать с различными конфигурациями и посмотреть через монитор последовательного порта, как меняются показания.

В качестве примера возьмем делитель напряжения, в котором в одном плече находится фоторезистор, а в другом — резистор с постоянным номиналом 10 кОм (рис. 2.3).

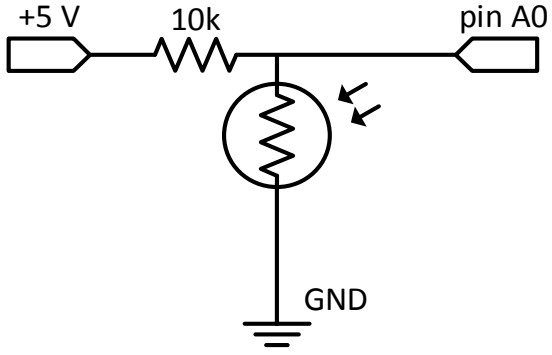


Рис. 2.3 — Схема подключения фоторезистора

Листинг программы такой же, как и при чтении значений с потенциометра.

Если вы загрузите программу считывания аналоговых данных, выведете

результат в последовательный порт и будете менять освещенность фоторезистора, вы не сможете получить весь диапазон значений от 0 до 1023, потому что у фоторезистора никогда не будет нулевого сопротивления. В результате вы определите минимальное и максимальное значения напряжения на выходе. Выбрав аналоговые значения, соответствующие темноте и максимальной освещенности, можно собрать «интеллектуальный» ночник, который будет светить более ярко в темном помещении, и наоборот.

Для того чтобы изменять яркость светодиода используется функция **`analogWrite()`**. Однако аргумент этой функции 8-разрядный, т. е. находится в диапазоне от 0 до 255, в то время как АЦП выдает значения от 0 до 1023. Для преобразования одного диапазона чисел в другой воспользуемся функцией **`map()`**.

В данном случае минимальная и максимальная освещенность в помещении соответствует границам исходного диапазона, а значения яркости — границами нового диапазона. Аргумент функции **`analogWrite()`** должен быть в диапазоне от 0 до 255.

В данном примере необходимо сопоставить меньшей освещенности большую яркость светодиода, т. е. минимальным значениям на аналоговом входе должны соответствовать максимальные значения на выводах светодиода.

Аналоговый датчик температуры

Температура может быть представлена аналоговым сигналом, например, генерируемым температурным датчиком TMP36 (рис. 2.5), который производится компанией Analog Devices.

Рис. 2.4 — Внешний вид температурного датчика TMP36

Датчик TMP36 выдает напряжение, пропорциональное температуре, благодаря чему текущую температуру можно определять простым преобразованием. Например, температуре 25 °C соответствует выходное напряжение 750 мВ, и каждое изменение температуры на 1 градус влечет изменение напряжения на

10 мВ. Датчик TMP36 может измерять температуры в диапазоне от –40 °C до 125 °C.

Функция **analogRead()** возвращает значение в диапазоне от 0 до 1023, который соответствует диапазону напряжения от 0 до (почти) 5000 мВ (5 В). Если умножить результат вызова **analogRead()** на (5000/1024), получится напряжение, фактически возвращаемое сенсором. Далее нужно вычесть 500 (смещение, используемое датчиком TMP36, чтобы обеспечить возможность измерения отрицательных температур), и разделить на 10. В результате получится температура в градусах Цельсия. Если вы предпочитаете использовать шкалу Фаренгейта, умножьте температуру в градусах Цельсия на 1,8 и прибавьте 32.

Скетч, выполняющий считывание напряжения с температурного датчика и преобразующий напряжение в градусы, показан в Листинге 2.

```
float voltage = 0;
float celsius = 0;
float sensor = 0;

void loop(){
// прочитать напряжение с датчика
// и преобразовать в градусы Цельсия
sensor = analogRead(0);
// преобразовать в милливольты
voltage = (sensor*5000)/1024;
voltage = voltage-500; // учесть смещение
// преобразовать милливольты в градусы
celsius = voltage/10;
}
```

2.2.2. Сервопривод

Сервопривод (servo), рулевая машинка — устройство, обеспечивающее преобразование сигнала в строго соответствующее этому сигналу перемещение (как правило, поворот) исполнительного устройства. Представляет собой прямоугольную коробку с мотором, схемой и редуктором внутри и выходным валом, который может поворачиваться на строго фиксированный угол, определяемый входным сигналом (рис. 2.5). Как правило, этот угол имеет предел в 60 градусов, иногда в 180. Бывают сервоприводы и постоянного вращения.

На вал надевается рычаг в форме круга, крестовины или перекладинки для передачи вращающего движения на рабочий орган. После поворота вал остается в том же положении, пока не придет иной управляющий сигнал. Смысл сервопривода в гарантированном выполнении заданной команды. Если

внешняя сила не позволит выполнить поворот на нужный угол, сервопривод все равно закончит движение после окончания действия мешающего внешнего воздействия. Воспрепятствовать этому может лишь разрушение сервопривода, снятие внешнего управляющего сигнала или пропадание напряжения питания.

Рис. 2.5 — Устройство сервопривода

Сервопривод управляется с помощью импульсов переменной длительности. Для посылки импульсов используется сигнальный провод. Для управления сервоприводом в Arduino имеется стандартная библиотека Servo.h.

Сервопривод подключается тремя проводами: питание (Vcc), "земля" (Gnd) и сигнальный (C). Питание — красный провод, он может быть подключен к выводу +5 В на плате Arduino. Черный или коричневый провод — "земля" подключается к выводу Arduino GND, сигнальный (оранжевый/желтый/белый) провод подключается к цифровому выводу контроллера Arduino.

Функция **`attach()`** подключает переменную **`servo`** к указанному выходу, с которого осуществляется управление приводом: **`servo.attach(pin).`**

Функция **`write(int)`** передает значения для управления приводом. Для стандартного сервопривода это угол поворота:

```
servo.write(angle);
```

где **`angle`** — значение, записываемое в **`servo`** (от 0 до 180).

2.2.3. Пьезоэлектрический зуммер

Звук распространяется по воздуху в виде волны. Работа звуковых колонок, удар в барабан или колокол создают вибрацию воздуха. Частицы воздуха за счет колебаний передают энергию все дальше и дальше. Волна давления пе-

редается от источника к вашей барабанной перепонке через реакцию вибрирующих частиц.

Можно управлять двумя параметрами этих колеблющихся частиц: частотой и амплитудой. Под частотой понимают скорость вибрации частиц воздуха, а под амплитудой — размах их колебаний. В физическом смысле звуки с большой амплитудой громче, чем с малой. Тон высокочастотных звуков выше (например, сопрано), а низкочастотных — ниже (например, бас).

Плата Arduino позволяет задать частоту звукового сигнала в форме меандра. Меандр является цифровым периодическим сигналом с равными по длительности высоким и низким состояниями.

Что касается амплитуды, ею можно управлять, изменяя силу тока в динамике. Подключение потенциометра последовательно с динамиком позволяет регулировать уровень громкости звука.

Пьезоэлектрический зуммер — это устройство в цилиндрическом корпусе, которое можно использовать для подачи громких и раздражающих звуковых сигналов, например для предупреждения об аварии. На рис. 2.6 изображен зуммер TDK PS1240 рядом с 25-центовой монетой США, чтобы получить представление о его размерах.

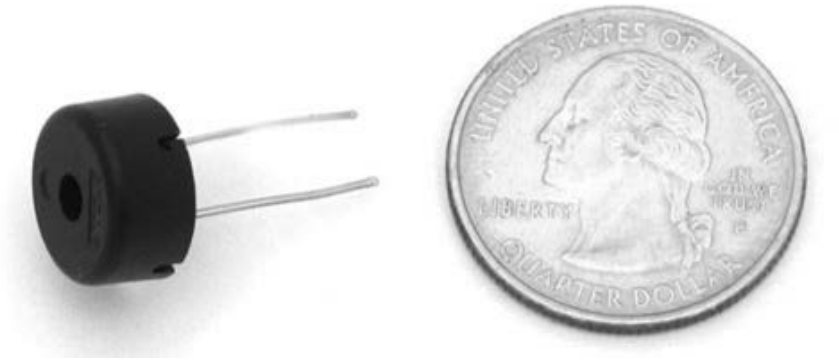


Рис. 2.6 — Внешний вид пьезоэлектрического зуммера

Внутри зуммера находится очень тонкая пластина, которая изменяет форму при подаче напряжения. Когда ток подается на зуммер импульсами, пластина начинает вибрировать, генерируя звуковые волны определенной частоты.

Значок, обозначающей зуммер на схеме, напоминает динамик (рис. 2.7), что упрощает его идентификацию.

Рис. 2.7 — Обозначение пьезоэлектрического зуммера на схемах

Генерирование звуков с помощью Arduino. Функция tone()

В Arduino IDE есть встроенная функция для генерации звуков произвольной частоты. Функция tone () формирует меандр с заданной частотой и выдает его на выбранный вами выходной контакт Arduino.

Аргументы tone ():

— первый аргумент устанавливает номер контакта Arduino для генерации волны;

— второй аргумент задает частоту сигнала;

— третий (необязательный) аргумент определяет продолжительность звучания; если этот аргумент не установлен, звук продолжается до тех пор, пока не вызвана функция noTone().

Функция tone() взаимодействует с одним из аппаратных таймеров контроллера ATmega, поэтому ее можно вызвать и продолжать работать с Arduino, а звук будет играть в фоновом режиме.

Заголовочный файл, содержащий значения частот для всех нот pitches.h

Когда дело доходит до воспроизведения музыкальных звуков, полезно создать заголовочный файл, определяющий частоты для музыкальных нот. Это делает программу более понятной при составлении простых музыкальных мелодий. Те, кто знаком с нотными знаками, знают, что ноты обозначаются буквами. В Arduino IDE есть специальный файл, содержащий значения частот для всех нот.

Arduino IDE создает новый файл внутри папки с одноименным названием. Добавляя в эту папку новые файлы, вы можете включать их в свою программу, в результате код будет лучше структурирован. Скопируйте файл pitches.h в папку, созданную Arduino IDE, для нового проекта. Теперь заново откройте в Arduino IDE этот файл.

Перейдите на вкладку pitches.h, чтобы увидеть содержимое файла. Обратите внимание, что это всего лишь список операторов определений, которые задают соответствие названий нот и значений частот. Чтобы использовать эти определения при компиляции программы для Arduino, необходимо сообщить компилятору, где искать данный файл. Для этого добавьте соответствующую строку кода в начало файла #include "pitches.h".

Для компилятора это, по существу, то же самое, что копирование и вставка содержимого файла заголовка в начало основного файла. Тем не менее, код становится аккуратнее и проще для чтения.

2.3. Порядок выполнения работы

Согласно приведенным ниже вариантам задания составить электрическую схему в симуляторе TinkerCAD и написать скетч для микроконтроллера Ar-

duino Uno. Выполнить проверку работоспособности электрической схемы и скетча с помощью моделирования. Собрать электрическую схему с помощью макетной платы и показать работоспособность схемы преподавателю.

Задание 1

Соберите схему с фоторезистором и светодиодом, показанную на рис. 2.8.

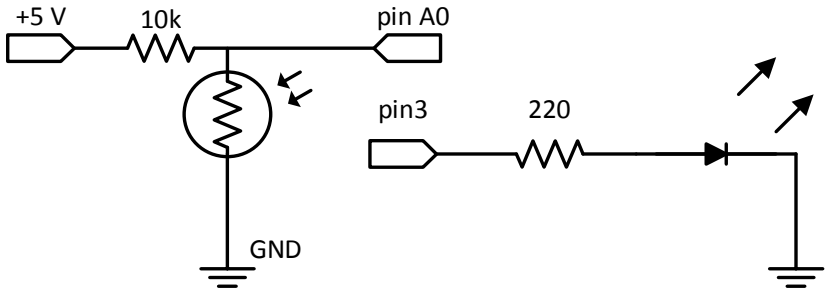


Рис. 2.8 — Схема подключения фоторезистора и светодиода

Напишите программу, которая будет управлять яркостью светодиода в зависимости от освещенности (чем темнее в помещении, тем ярче горит светодиод и наоборот).

Задание 2

Соберите схему с датчиком температуры TMP36, показанную на рис. 2.9.

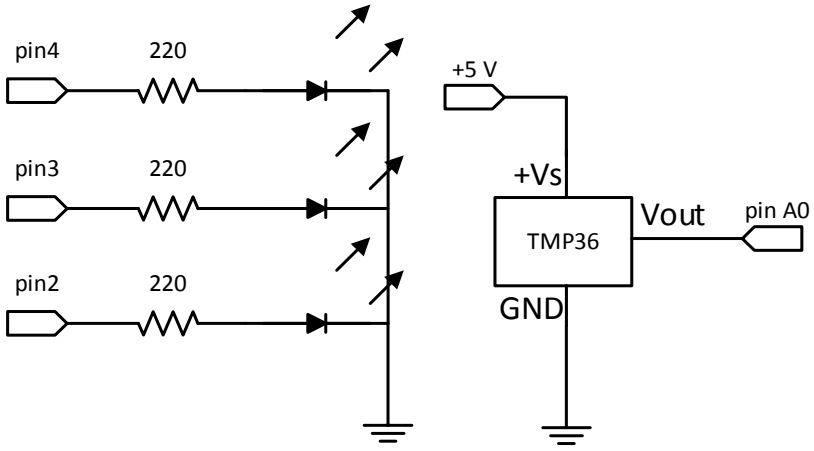


Рис. 2.9 — Схема подключения светодиодов и температурного датчика TMP36

Напишите программу, которая будет обрабатывать данные с температурного датчика и когда температура опустится ниже +20 °C, должен включиться синий светодиод. При температуре в диапазоне от +20 °C до +26 °C должен быть включен зеленый светодиод, а когда температура превысит +26 °C, должен включаться красный светодиод.

Задание 3

Соберите схему с сервоприводом, как показано на рис. 2.10.

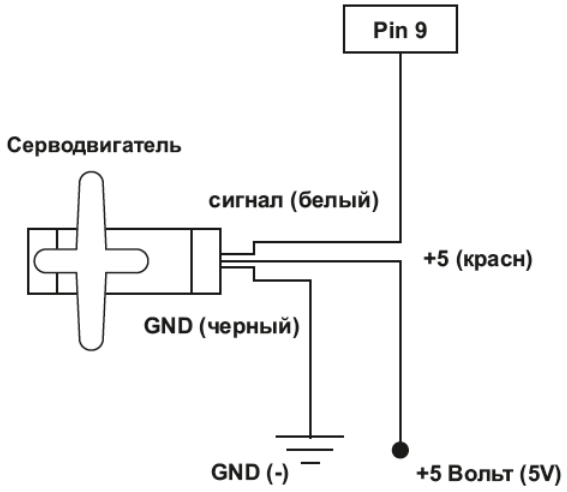


Рис. 2.10 — Схема подключения сервопривода

Напишите программу, которая с помощью сервопривода выполняет действия, представленные в таблице 2.1.

Таблица 2.1 — Варианты заданий для работы с сервоприводом

Варианты	**Задание**
1,5,9,13,17,21,25,29	Осуществить последовательно поворот сервопривода на полной скорости на углы 90, 180, 0 градусов
2,6,10,14,18,22,26,30	Осуществить поворот сервопривода на позицию 180 градусов с шагом два градуса
3,7,11,15,19,23,27	Осуществить поворот сервопривода с позиции 180 градусов на позицию 0 градусов с шагом один градус
4,8,12,16,20,24,28	С помощью потенциометра осуществлять поворот сервопривода в пределах от 0 до 180 градусов

Задание 4

Соберите схему с зуммером как показано на рис. 2.11.

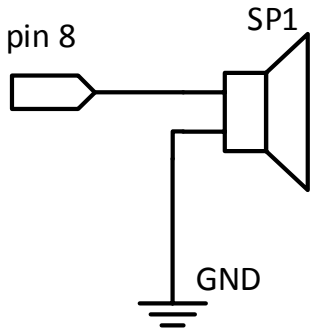


Рис. 2.11 — Подключение зуммера

Загрузите приведенный ниже скетч в Arduino и воспроизведите мелодию, соответствующую массиву **melody[].**

```
#include "pitches.h"
int melody[] = { NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3,
NOTE_G3, 0, NOTE_B3, NOTE_C4 };
int noteDurations[] = { 4, 8, 8, 4, 4, 4, 4, 4 };
void setup() {
   for (int thisNote = 0; thisNote < 8; thisNote++) {
   int noteDuration = 1000 / noteDurations[thisNote];
   tone(8, melody[thisNote], noteDuration);
   int pauseBetweenNotes = noteDuration * 1.30;
   delay(pauseBetweenNotes);
   noTone(8);
   }
}
void loop() {
}
```

Замените содержание массивов **melody[]** *и* **noteDurations[]** *значениями и длительностей нот рингтона «Nokia». Загрузите скетч в Arduino и воспроизведите мелодию. Обратите внимание, что в рингтоне «Nokia» мелодии 13 нот, поэтому в условии выполнения цикла* **for** *необходимо заменить условие на* **thisNote < 13.**

Ноты и их длительности рингтона «Nokia»: E6(8), D6(8), FS5(4), GS5(4), CS6(8), B5(8), D5(4), E5(4), B5(8), A5(8), CS5(4), E5(4), A5(1).

Задание 5

Соберите схему, подключив потенциометр к аналоговому выводу A0 и пьезо-динамик к цифровому выводу 3. Напишите скетч для изменения высоты звука с помощью потенциометра.

Задание 6

Соберите схему терменвокса, показанную на рис. 2.12.

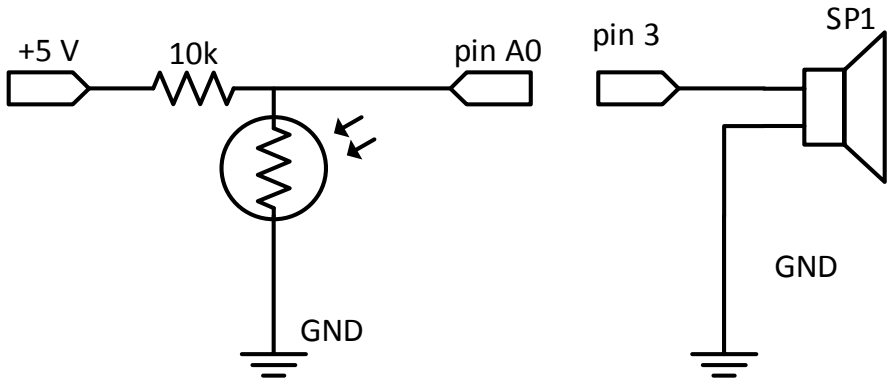


Рис. 2.12 — Схема подключения фоторезистора и динамика

Терменвокс — это музыкальный инструмент, в котором частота звучания изменяется бесконтактным путем.

Для этой цели используется фоторезистор, который можно закрывать рукой, тем самым уменьшая или увеличивая на него световой поток. Сопротивление фоторезистора, а значит и падение на нем напряжения, будет определять частоту звучания динамика.

Напишите скетч в Arduino для воспроизведения звука, высота которого изменяется с помощью руки.

2.4.Содержание отчета

2.4.1. Сформулировать цель работы.

2.4.2. Привести постановку задачи и текст задания.

2.4.3. Привести краткие теоретические сведения.

2.4.4. Привести принципиальную и функциональную схему устройства, соответствующую заданию.

2.4.5. Привести текст программы (программа должна содержать комментарии).

2.4.6. Привести фотографию собранного устройства.

2.4.7. Сделать выводы по работе.

2.5. Контрольные вопросы

2.5.1. Что такое аналоговый и цифровой сигналы?

2.5.2. Что такое аналого-цифровой преобразователь?

2.5.3. Охарактеризуйте функцию **`analogRead()`**.

2.5.4. Опишите подключение фоторезистора и способ определения освещенности с его помощью.

2.5.5. Охарактеризуйте функцию **`map()`**.

2.5.6. Как с помощью Arduino можно измерять температуру?

2.5.7. Что такое сервопривод? Опишите принцип его работы.

2.5.8. Опишите подключение сервопривода к Arduino и основные функции класса Servo.

2.5.9. Что такое пьезоэлектрический зуммер?

2.5.10. Как сгенерировать звук с помощью Arduino?

3. ПРАКТИЧЕСКАЯ РАБОТА № 6 ПРОГРАММИРОВАНИЕ ЦИФРОВЫХ СЕМИСЕГМЕНТНЫХ ИНДИКАТОРОВ

3.1. Цель работы

Приобретение практических навыков работы с одноразрядным семисегментным индикатором; изучение принципов использования динамической индикации для работы с многоразрядными индикаторами.

3.2. Варианты заданий

Согласно приведенным ниже вариантам задания составить электрическую схему в симуляторе TinkerCAD и написать скетч для микроконтроллера Arduino. Выполнить проверку работоспособности электрической схемы и скетча с помощью моделирования. Собрать электрическую схему с помощью макетной платы и показать работоспособность схемы преподавателю.

Задание 1

Согласно вариантам из таблицы 3.1. выполните программирование одноразрядного семисегментного индикатора (ССИ).

Таблица 3.1 — Варианты заданий для программирования одноразрядного ССИ

Варианты	Задание
1, 9	Последовательно выводить на ССИ цифры от 0 до 9 с интервалом времени 1 с, используя функцию задержки
2, 0	Последовательно выводить на ССИ цифры от 9 до 0 с интервалом времени 1 с, используя функцию задержки
3	Последовательно выводить на ССИ цифры от 0 до 9 с интервалом времени 1 с, используя функцию времени
4	Последовательно выводить на ССИ цифры от 9 до 0 с интервалом времени 1 с, используя функцию времени
5	Увеличивать на единицу значение на ССИ с помощью кнопки
6	Уменьшать на единицу значение на ССИ с помощью кнопки
7	С помощью потенциометра изменять значение на ССИ в пределах от 0 до 9
8	С помощью двух кнопок изменять значение на ССИ вверх и вниз

Задание 2

Выполните задание согласно вашему варианту из таблицы 3.1, используя регистр сдвига (МС 74HC595) для подключения ССИ к Arduino.

Задание 3

Согласно вариантам из таблицы 3.2. выполните программирование двух- и четырехразрядного ССИ, используя динамическую индикацию и для подключения ССИ — регистр сдвига.

Таблица 3.2 — Варианты заданий для программирования двух- и четырехразрядного ССИ

Варианты	Задание
1, 7	Вывести на четырехразрядный ССИ значение с датчика температуры с точностью до десятых
2, 8	С помощью кнопки останавливать и продолжать отсчет времени
3, 9	Последовательно выводить на двухразрядный ССИ цифры от 0 до 59 с интервалом времени 1 с, используя функцию времени
4, 0	Последовательно выводить на двухразрядный ССИ цифры от 59 до 0 с интервалом времени 1 с, используя функцию времени
5	Последовательно выводить на четырехразрядный ССИ секунды и минуты, используя функцию времени
6	С помощью потенциометра изменять значение на четырехразрядном ССИ в пределах от 0 до 9999

3.3. Теоретические сведения

3.2.1. Одноразрядный семисегментный индикатор

Цифровые семисегментные индикаторы используются для отображения чисел (рис. 3.1). Именно поэтому они используются при конструировании цифровых будильников, спидометров и других устройств, отображающих числовую информацию. Каждый индикатор имеет семисегментный дисплей с восемью светодиодами. Выпускаются индикаторы со светодиодами разного цвета. Чтобы уменьшить число выводов у индикаторов, все аноды или катоды светодиодов соединены с общим выводом, который называется общим анодом или общим катодом соответственно.

Светодиодные сегменты снабжены метками от A до G и DP (Decimal Point — десятичная точка). Каждому сегменту соответствует свой вывод, соединенный с анодом, а катоды подключены к общему выводу. Размещение сегментов всегда соответствует изображенному на рис. 3.2, где сегмент A находится вверху, B — справа и т. д. То есть, например, чтобы показать цифру 7, следует подать напряжение на сегменты A, B и C.

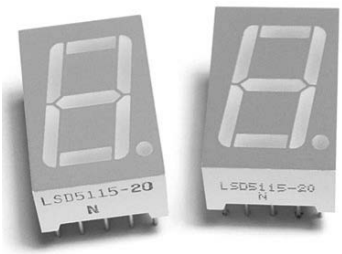
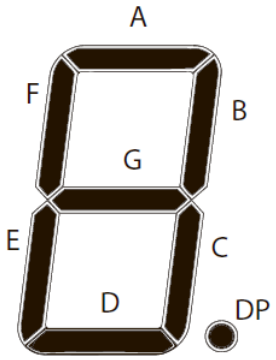


Рис. 3.1 — Внешний вид семисегментных индикаторов

Порядок размещения выводов на корпусе светодиодного индикатора может отличаться у разных производителей, но они всегда соответствуют шаблону, изображенному на рис. 3.2.

Для изображения семисегментных индикаторов на принципиальных схемах используется значок, показанный на рис. 3.3.



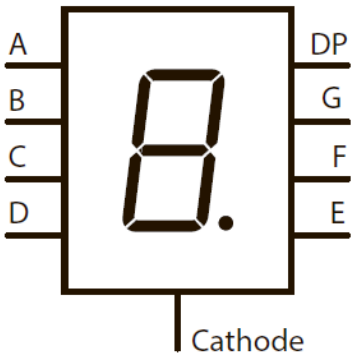


Рис. 3.2 — Карта светодиодов индикатора Рис. 3.3 — Изображение индикатора на принципиальной схеме

Для выполнения заданий практической работы будут использоваться семисегментные индикаторы модели 5161ah, назначение выводов которой показаны на рис. 3.4.

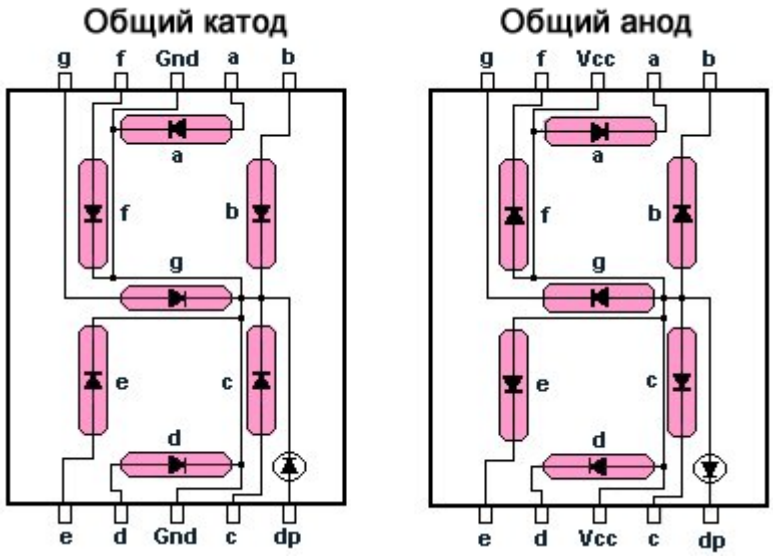


Рис. 3.4 — Назначение выводов семисегментного индикатора с общим катодом и общим анодом

Для того чтобы зажечь нужный сегмент, необходимо вывод соответствующий этому сегменту подключить к цифровому выводу платы Arduino и подать на него высокий уровень напряжения с помощью команды **`digitalWrite(pin,HIGH)`** для индикатора с общим катодом и низкий уровень напряжения с помощью команды **`digitalWrite(pin,LOW)`** для индикатора с общим анодом. В дальнейшем будем рассматривать схему только с общим катодом.

Чтобы вывести цифру на индикатор, нужно подключить выводы, обозначенные на рис. 3.4 как a, b, c, d, e, f, g, через токоограничивающие резисторы

220 Ом к цифровым выводам с 2 по 8 и подать на них комбинацию нулей и единиц в соответствии с таблицей 3.1.

Таблица 3.1 — Матрица для отображения цифр на семисегментном индикаторе

Цифра	A	B	C	D	E	F	G	DP	Десятичное
0	1	1	1	1	1	1	0	0	252
1	0	1	1	0	0	0	0	0	96
2	1	1	0	1	1	0	1	0	218
3	1	1	1	1	0	0	1	0	242
4	0	1	1	0	0	1	1	0	102
5	1	0	1	1	0	1	1	0	182
6	1	0	1	1	1	1	1	0	190
7	1	1	1	0	0	0	0	0	224
8	1	1	1	1	1	1	1	0	254
9	1	1	1	1	0	1	1	0	246
A	1	1	1	0	1	1	1	0	238
B	0	0	1	1	1	1	1	0	62
C	1	0	0	1	1	1	0	0	156
D	0	1	1	1	1	0	1	0	122
E	1	0	0	1	1	1	1	0	158
F	1	0	0	0	1	1	1	0	142

Например, чтобы отобразить цифру 7, как показано на рис. 3.5, нужно включить сегменты A, B и C, которые подключены к выводам 2, 3, 4. Чтобы установить высокий уровень на первых трех выводах, соответствующих перечисленным сегментам, мы должны сформировать посылку B1110000.

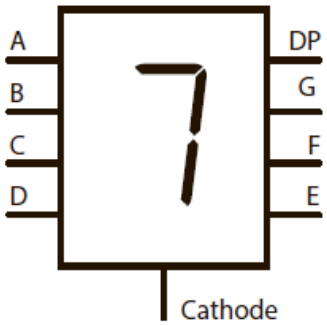


Рис. 3.5 — Отображение на индикаторе цифры 7

3.2.2. Четырехразрядный семисегментный индикатор

Динамическая индикация — это метод отображения целостной картины через быстрое последовательное отображение отдельных элементов этой картины. Причем, «целостность» восприятия получается благодаря инерционности человеческого зрения.

Рассмотрим принцип динамической индикации на примере четырехразрядного семисегментного индикатора (рис. 3.6).

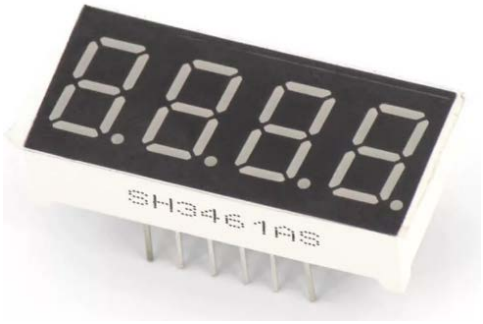


Рис. 3.6 —Внешний вид четырехразрядного семисегментного индикатора

Данный индикатор имеет четыре цифровых разряда, каждый из которых состоит из семи светодиодов. Если подключать все светодиоды напрямую к микроконтроллеру, то нам понадобилось бы как минимум 28 выводов. Динамическая индикация позволяет уменьшить число выводов, благодаря тому, что все четыре разряда можно подключить параллельно, то есть посадить выводы сегментов на общую шину, и последовательно подавать напряжение на адресные входы индикаторов и одновременно выдавать в шину данных 7-сегментный код, соответствующий индикатору, активному в данный момент.

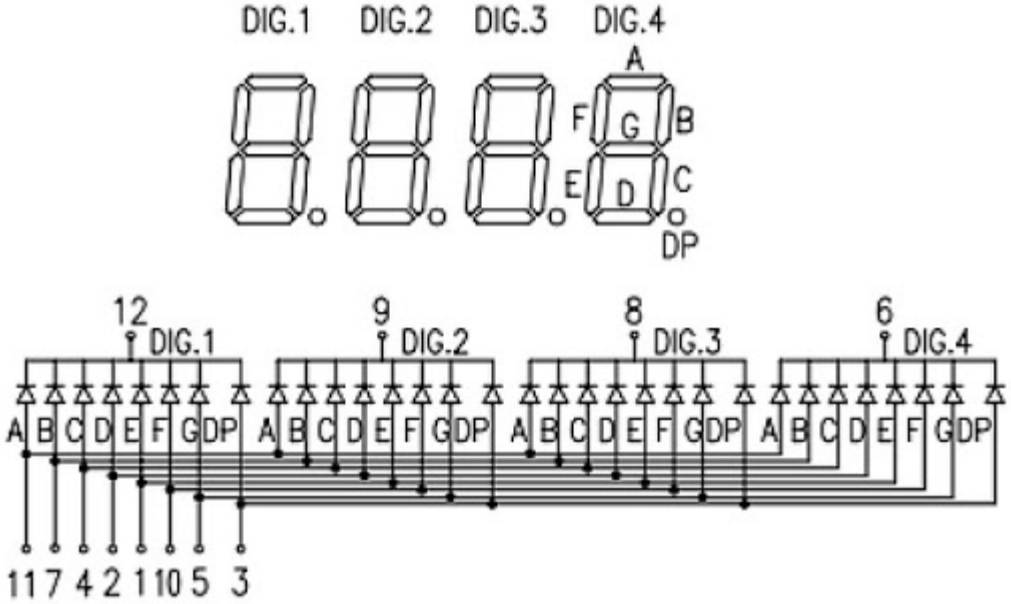


Рис. 3.7 — Схема и обозначение выводов четырехразрядного семисегментного индикатора

3.3. Порядок выполнения работы

3.3.1. Подключение семисегментного индикатора (ССИ)

Схема подключения одноразрядного ССИ показана на рис. 3.8.

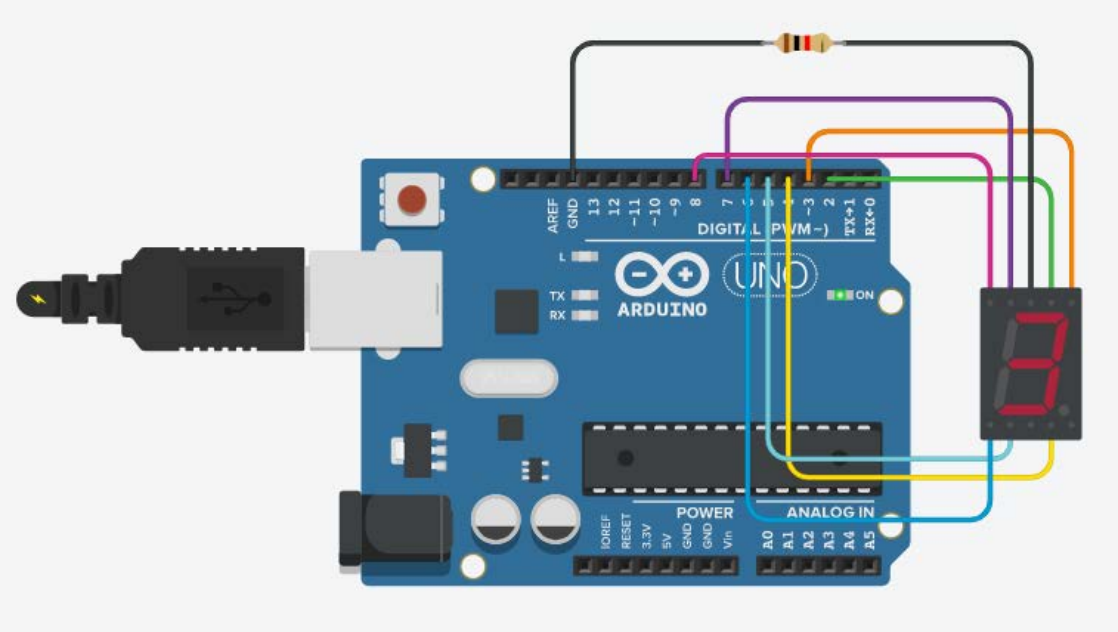


Рис. 3.8 — Схема подключения одноразрядного ССИ

3.3.2. Вывод на ССИ десятичных цифр

Для вывода десятичных цифр на индикатор необходимо загрузить приведенный ниже скетч.

```
int segmentsPins[] = {2, 3, 4, 5, 6, 7, 8, 9};
//Задаем пины для каждого сегмента (из 7 + 1(точка))

void setup() {
  // Все выходы программируем как OUTPUT
  for (int i = 0; i < 8; i++) {
    pinMode(segmentsPins[i], OUTPUT);
  }
}

//{A, B, C, D, E, F, G,DP} - распиновка сегментов
int seg[10][8] = {
  {1, 1, 1, 1, 1, 1, 0, 0}, //Цифра 0
  {0, 1, 1, 0, 0, 0, 0, 0}, //Цифра 1
  {1, 1, 0, 1, 1, 0, 1, 0}, //Цифра 2
  {1, 1, 1, 1, 0, 0, 1, 0}, //Цифра 3
  {0, 1, 1, 0, 0, 1, 1, 0}, //Цифра 4
  {1, 0, 1, 1, 0, 1, 1, 0}, //Цифра 5
  {1, 0, 1, 1, 1, 1, 1, 0}, //Цифра 6
  {1, 1, 1, 0, 0, 0, 0, 0}, //Цифра 7
  {1, 1, 1, 1, 1, 1, 1, 0}, //Цифра 8
  {1, 1, 1, 1, 0, 1, 1, 0}  //Цифра 9
};

void loop() {
```

```
for (int i = 0; i < 10; i++) {//Перебираем цифры от 0 до 9
    for (int j = 0; j < 8; j++) {
// Каждый сегмент по очереди - исходя из заданной карты
      digitalWrite(segmentsPins[j],
      ((seg[i][j] == 1) ? HIGH: LOW));
    }
delay(1000);
}
}
```

Для упрощения кода в приведенном скетче можно заменить двумерный массив одномерным, используя таблицу 3.1. Для этого в массиве **seg[][]** нужно заменить строки на соответствующие двоичные числа:

```
int seg[10] = {
  B11111100, //Цифра 0
  B01100000, //Цифра 1
  B11011010, //Цифра 2
  B11110010, //Цифра 3
  B01100110, //Цифра 4
  B10110110, //Цифра 5
  B10111110, //Цифра 6
  B11100000, //Цифра 7
  B11111110, //Цифра 8
  B11110110  //Цифра 9
};
```

или на их десятичные эквиваленты:

```
int seg[10] = { 252, 96, 218, 242, 102, 182, 190, 224, 254, 246 };
```

Для вывода десятичных цифр на индикатор необходимо использовать следующие команды:

```
for (int i = 0; i < 10; i++) { // Перебираем цифры от 0 до 9
    for (int j = 0; j < 8; j++) {
// Каждый сегмент по очереди - исходя из заданной карты
      digitalWrite(segmentsPins[j],
         ((bitRead(seg[i],7-j) == 1) ? HIGH: LOW));
    }
delay(1000);
}
```

3.3.3. Одноразрядный секундомер

Для организации счета без использования функции задержки **delay(),** воспользуемся функцией работы со временем **millis().** Функция **millis()** возвращает число миллисекунд с момента начала выполнения текущей программы на плате Arduino.

Для удобства написания скетчей в дальнейшем создадим функцию, которая будет выводить нужную цифру на ССИ **void showNum(int number)**.

```
void showNum(int number)
{
for (int j = 0; j < 8; j++) {
      digitalWrite(segmentsPins[j],
      (bitRead(seg[number],7-j) == 1) ? HIGH: LOW));
}
```

Скетч для одноразрядного ССИ приведен ниже.

```
int segmentsPins[] = {2, 3, 4, 5, 6, 7, 8, 9};
//Задаем пины для каждого сегмента (из 7 + 1(точка))

void setup() {
  // Все выходы программируем как OUTPUT
  for (int i = 0; i < 8; i++) {
    pinMode(segmentsPins[i], OUTPUT);
  }
}
int seg[10] = { 252, 96, 218, 242, 102, 182, 190,
224, 254, 246 };
void loop() {
int number = millis()/1000%10;
showNum(number);
}
void showNum(int number){
for (int j = 0; j < 8; j++) {
      digitalWrite(segmentsPins[j],
      (bitRead(seg[number],7-j) == 1) ? HIGH: LOW));
}
}
```

3.3.4. Подключение семисегментного индикатора, используя регистр сдвига 74CH595

Микросхема 74CH595 представляет собой восьмиразрядный регистр сдвига с последовательным вводом, последовательным или параллельным выводом информации, с триггером-защелкой и тремя состояниями на выходе. Регистр сдвига принимает поток последовательных битов и одновременно выводит их значения на параллельных контактах ввода-вывода. Данный регистр применяется для экономии выходов микроконтроллера: позволяет управлять напряжением на своих восьми выходах, заняв всего три выхода микроконтроллера. Таким образом, количество рабочих выводов увеличивается на пять.

Регистры сдвига позволяют легко конвертировать последовательные и параллельные методы передачи данных.

Обозначение выводов микросхемы 74CH595 показаны на рис. 3.9.

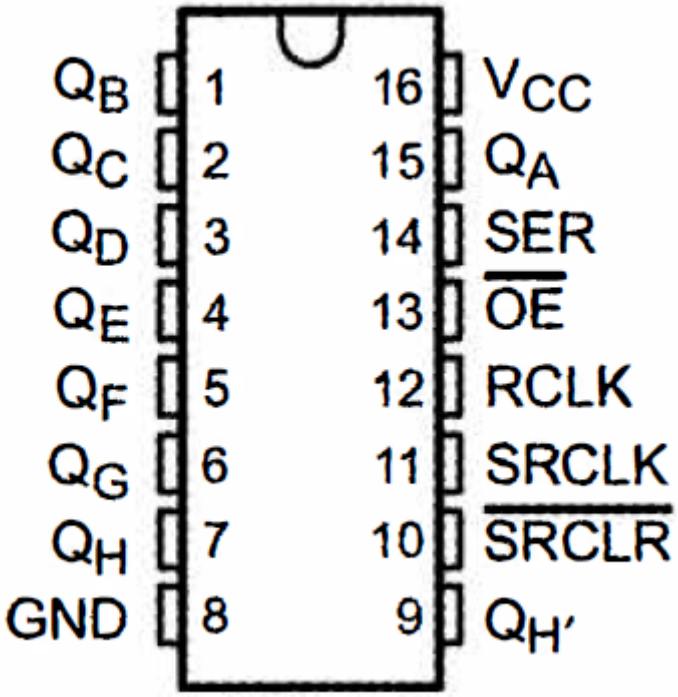


Рис. 3.9 — Обозначение выводов микросхемы 74CH595

Назначение выводов микросхемы 74CH595 показаны в таблице 3.3.

Таблица 3.3 — Назначение выводов микросхемы 74CH595

QA — QH	восемь параллельных выходов сдвигового регистра
GND	соединяется с землей платы Arduino
Vcc	соединяется с выводом 5 В платы Arduino
SER	вход данных. По этому входу передаются 8 последовательных битов данных для установки значений на параллельных выходах
SRCLK	тактовый вход. При подаче импульса высокого напряжения HIGH на этот вход происходит считывание одного бита данных с входа SER в сдвиговый регистр. Для получения всех 8 битов данных необходимо подать 8 импульсов на этот контакт
RCLK	этот вход называется защелкой и служит для одновременного вывода последовательных данных на параллельные выходы.
OE	разрешение вывода данных на параллельные выходы. Черта сверху означает, что активный уровень для этого входа — низкий. Когда на этом входе низкий уровень, параллельные выходы будут включены, когда высокий — выключены.
SRCLR	это вход сброса. Подача на него напряжения низкого уровня очищает содержимое регистра сдвига.

Схема подключения ССИ к плате Arduino, используя регистр сдвига, показана на рис. 3.10.

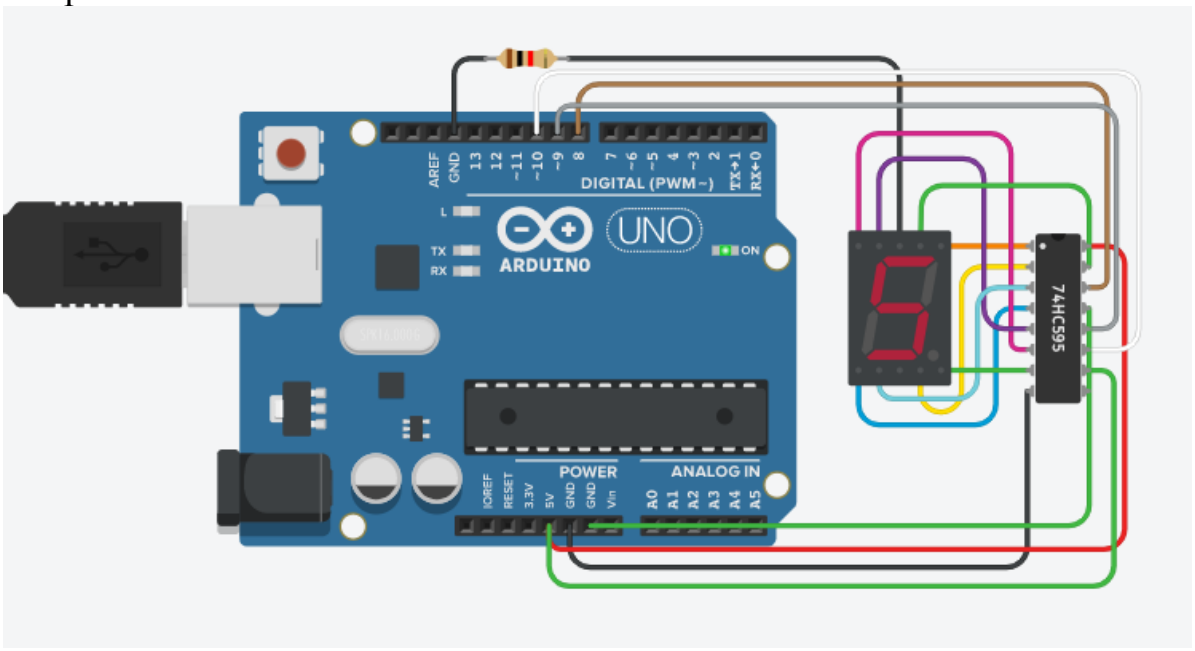


Рис. 3.10 — Схема подключения одноразрядного ССИ через регистр сдвига

Для передачи данных из Arduino в сдвиговые регистры используется функция **`shiftOut().`**

Эта функция принимает четыре аргумента:

- номер контакта DATA;
- номер контакта CLOCK;
- порядок выдачи битов;
- значение, выдаваемое на выход.

Например: **`shiftOut(DATA, CLOCK, MSBFIRST, B10101010);`**

Ниже представлен скетч для последовательного вывода цифр на ССИ, используя регистр сдвига.

```
const int SER =8;
const int LATCH =9;
const int CLK =10;
int number =0;
int seg[10] = { 252, 96, 218, 242, 102, 182, 190, 224, 254, 246 };

void setup() {
    pinMode(SER, OUTPUT);
    pinMode(LATCH, OUTPUT);
    pinMode(CLK, OUTPUT);
    digitalWrite(LATCH, HIGH);
}

void loop() {
```

```
  number++;
  if(number==10)
    number=0;
showNum(number);
  delay(1000);
}

void showNum (int number){
  digitalWrite(LATCH, LOW); // LATCH - низкий
  shiftOut(SER, CLK, LSBFIRST, seg[number]);
  digitalWrite(LATCH, HIGH);

}
```

3.3.5. Подключение двухразрядного семисегментного индикатора

Схема подключения двухразрядного ССИ показана на рис. 3.11.

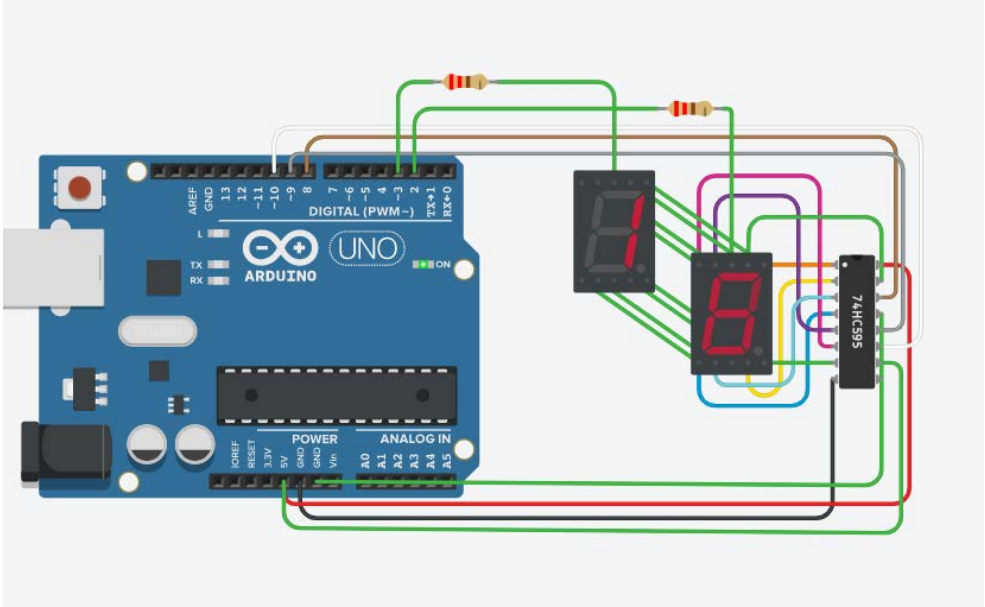


Рис. 3.11 — Схема подключения двухразрядного ССИ

Для динамического вывода цифр на двухразрядный индикатор необходимо с цифрового вывода платы Arduino подать логический сигнал (низкого для ССИ с общим катодом или высокого уровня для ССИ с общим анодом) на адресный вход индикатора первого разряда и противоположный сигнал на адресный вход индикатора второго разряда и восьмибитный код, соответствующий какой-нибудь цифре. Выждать какое-то время, например, 5 мс, и повторить эту операцию для другого разряда.

Добавим в функцию **`showNum()`** еще один аргумент **`digit`**, который будет задавать номер подключаемого разряда: 0 — правый разряд, 1 — левый.

Ниже представлен пример функции **showNum()** для двухразрядного индикатора:

```
int katodPins[2] = {2, 3};
int digits[2] = {1, 2};
.....................
void showNum(int number, int digit){
for(int i=0; i < 2; i++)
   digitalWrite(katodPins[i],
   (bitRead(digits[digit],i) == 1) ? HIGH: LOW));
digitalWrite(LATCH, LOW); // LATCH – низкий
shiftOut(SER, CLK, LSBFIRST, seg[number]);
digitalWrite(LATCH, HIGH);
}
```

В скетче массив **digits[]** определяет порядок включения разрядов индикатора. Элемент массива **digits[0]** равен числу 1_{10} или 01_2; элемент массива **digits[1]** равен числу 2_{10} или 10_2.

В цикле for происходит включение нужного разряда для вывода цифры и выключение ненужного разряда.

Для реализации секундомера необходимо обеспечить вывод секунд и минут на соответствующие разряды индикатора. На нулевой разряд выводятся единицы секунд, на первый разряд — десятки секунд, на второй разряд — единицы минут и т.д.

Для вывода единиц секунд на индикатор необходимо значение, получаемое с помощью функции **millis()** преобразовать следующим образом:

```
sec_one = millis()/1000%10;
```

Для вывода десятков секунд на индикатор необходимо значение, получаемое с помощью функции **millis()** преобразовать следующим образом:

```
sec_dec = millis()/10000;
```

Для вывода минут на индикатор необходимо значение, получаемое с помощью функции **millis()** преобразовать следующим образом:

```
minutes = millis()/1000/60;
```

Скетч для секундомера на двухразрядном индикаторе представлен ниже:

```
const int SER = 8;
const int LATCH = 9;
const int CLK = 10;
int number =0;
int seg[10] = { 252, 96, 218, 242, 102, 182, 190, 224,
                254, 246 };
int katodPins[2] = {2, 3};
int digits[2] = {1, 2};

void setup() {
    pinMode(SER, OUTPUT);
    pinMode(LATCH, OUTPUT);
```

```
    pinMode(CLK, OUTPUT);
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    digitalWrite(LATCH, HIGH);
}

void loop() {
  number=millis()/1000;
  if(number==60)
     number=0;
  showNum(number%10,0); //единицы
  showNum(number/10,1); //десятки
}

void showNum (int number, int digit){
  for(int i=0; i < 2; i++)
     digitalWrite(katodPins[i],
     bitRead(digits[digit],i) == 1 ? HIGH: LOW);
  delay(5);
  digitalWrite(LATCH, LOW);
  shiftOut(SER, CLK, LSBFIRST, seg[number]);
  digitalWrite(LATCH, HIGH);
  }
```

3.4.Содержание отчета

3.4.1. Сформулировать цель работы.

3.4.2. Привести постановку задачи и текст задания.

3.4.3. Привести краткие теоретические сведения.

3.4.4. Привести принципиальную и функциональную схему устройства, соответствующую заданию.

3.4.5. Привести текст программы (программа должна содержать комментарии).

3.4.6. Привести фотографию собранного устройства.

3.4.7. Сделать выводы по работе.

3.5. Контрольные вопросы

3.5.1. Перечислите устройства вывода. Для чего они применяются?

3.5.2. Что представляет собой семисегментный индикатор? Опишите его структуру и подключение.

3.5.3. Каким образом отображать цифры на семисегментном индикаторе?

3.5.4. Что такое динамическая индикация. Чем отличается динамическая индикация от статической.

3.5.5. Опишите принцип работы трехразрядного семисегментного индикатора.

3.5.6. Как задается матрица для вывода цифр на семисегментный индикатор?

3.5.7. Охарактеризуйте функцию **`millis()`** при реализации вывода секунд и минут на семисегментный индикатор.

3.5.8. Опишите работу пользовательской функции **`showNum()`**.

3.5.9. Что такое регистр сдвига. Для чего он применяется?

3.5.10. Опишите назначение выводов регистра сдвига 74CH595.

3.5.11. Как подключить семисегментный индикатор к плате Arduino с помощью регистра сдвига 74CH595?

3.5.12. Как запрограммировать вывод цифр на семисегментный индикатор, подключенный к плате Arduino с помощью регистра сдвига 74CH595?

3.5.13. Как с помощью динамической индикации реализовать вывод двух разрядных десятичных чисел?

4. ПРАКТИЧЕСКАЯ РАБОТА № 4

ПРОГРАММИРОВАНИЕ ЖИДКОКРИСТАЛЛИЧЕСКИХ ИНДИКАТОРОВ

4.1. Цель работы

Изучить принцип работы жидкристаллического индикатора; приобрести практические навыки подключения жидкристаллического индикатора к микроконтроллеру и вывода на него информации.

4.2. Теоретические сведения

4.2.1. Жидкристаллический индикатор LCD1602

Жидкокристаллические индикаторы (ЖКИ), отображающие символьную информацию, например текст и числа, являются самыми недорогими и простыми в использовании среди всех ЖКИ. Они имеют разные размеры, измеряемые числом и длиной отображаемых строк. Некоторые имеют подсветку и позволяют выбирать цвет символов и фона. Для работы будем использовать ЖКИ с подсветкой, способный отображать 2 строки по 16 символов в каждой, показанный на рис. 4.1.

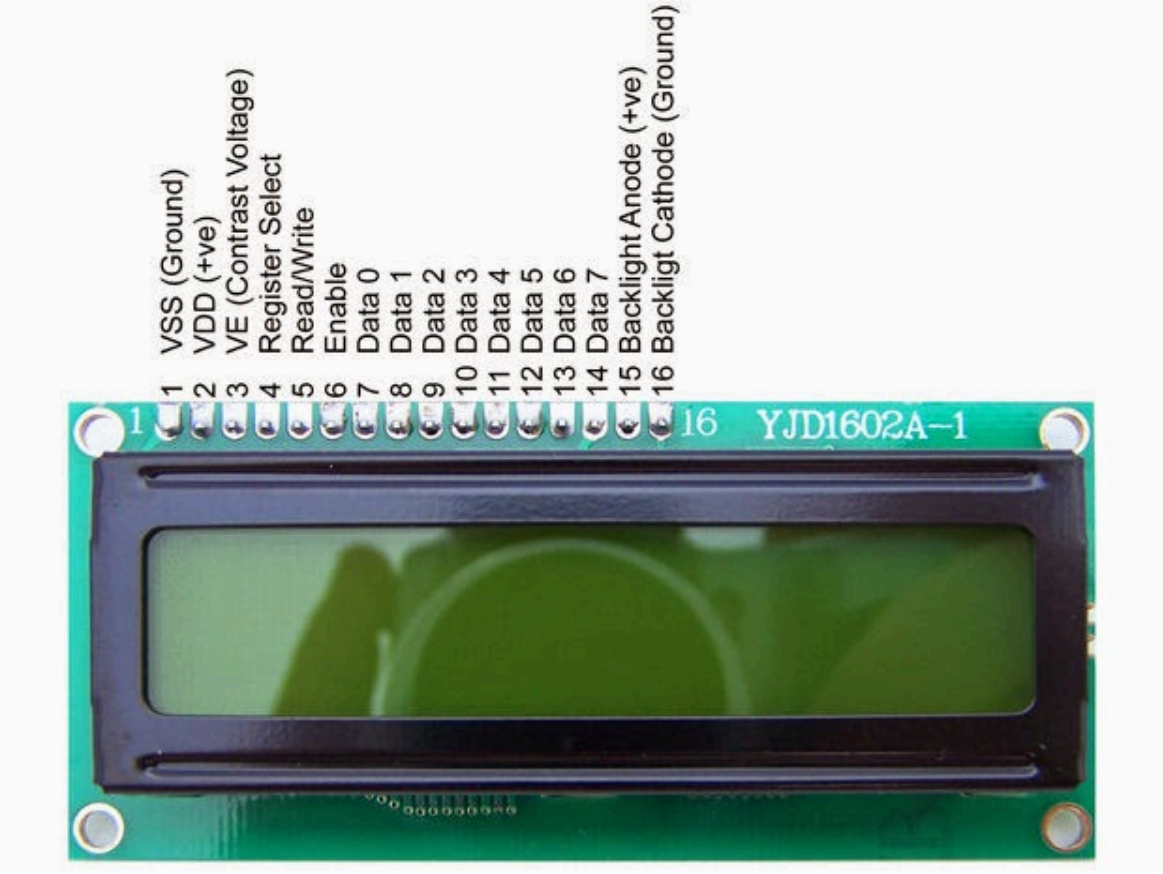


Рис. 4.1 — Жидкокристаллический индикатор LCD1602

Назначение выводов индикатора LCD1602 приведены в таблице 4.1.

Таблица 4.1 — Назначение выводов индикатора LCD1602

Номер вывода	Обозначение вывода	Назначение вывода
1	VSS	Общий вывод (GND)
2	VCC	Питание (+5 В)
3	VE	Установка контрастности экрана
4	RS	Команда, данные
5	R/W	Запись и чтение данных
6	E	Разрешить
7 — 14	DB0 — DB7	Линия данных
15	LED+	«Плюс» подсветки
16	LED-	«Минус» подсветки

Схема подключения LCD индикатора к плате Arduino показана на рис. 4.2.

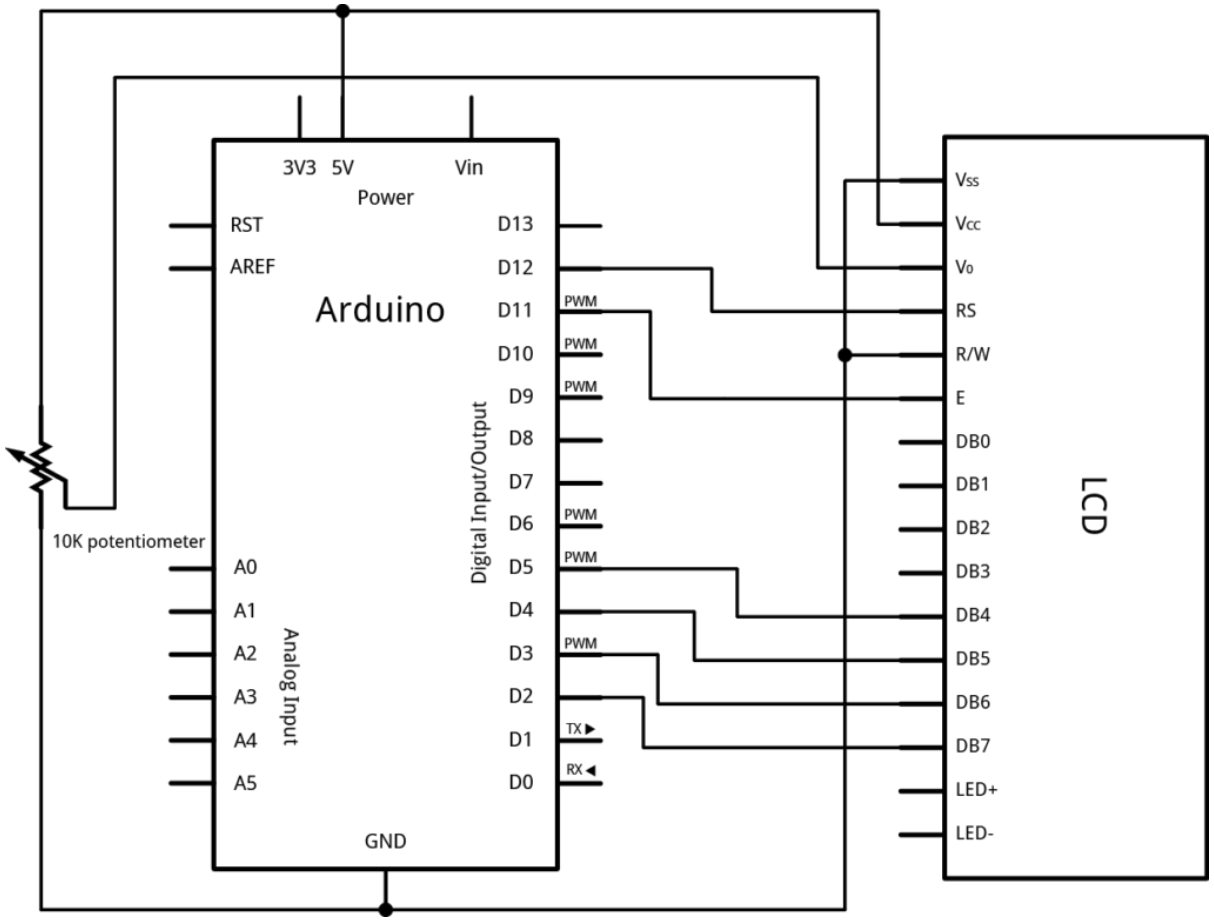


Рис. 4.2 — Схема подключения LCD индикатора к плате Arduino

4.2.2. Вывод текстовой и числовой информации на ЖКИ

Рассмотрим пример скетча, который выводит на ЖКИ текст «Hello world!», который приведен в Листинге 4.1.

//Листинг 4.1

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(4, 5, 6, 7, 8, 9);
// к выводам RS, E, DB4, DB5, DB6, DB7
void setup()
{
   lcd.begin(16, 2);    // инициализация индикатора
   lcd.clear();         // очистка экрана
}
void loop()
{
lcd.setCursor(0, 5);// установка курсора на первую
                    // строку со смещением на 5 символов
   lcd.print("Hello");
lcd.setCursor(1, 6); // установка курсора на вторую
                    // строку со смещением на 6 символов
   lcd.print("world!");
   delay(10000);
}
```

На рис. 4.3 показан результат работы скетча из листинга 4.1.

Рис. 4.3 — Демонстрация работы ЖКИ

В листинге 4.1 подключается библиотечный файл **LiquidCrystal.h** для работы с жидкокристаллическими индикаторами (он автоматически устанавливается в составе Arduino IDE). Далее создается объект класса **lcd,** в котором сообщается, к каким контактам на плате Arduino подключен ЖКИ.

Далее, в функции **void setup()** с помощью команды **lcd.begin(16, 2);** сообщается размер экрана ЖКИ в столбцах и строках (две строки по 16 символов в каждой). Команда **lcd.clear();** очищает экран ЖКИ.

Затем курсор устанавливается в позицию начала вывода текста вызовом функции

```
lcd.setCursor(x, y);
```

Здесь x — это номер столбца в строке (от 0 до 15), а y — номер строки (0 или 1). После этого производится вывод текста вызовом функции **lcd.print(),** например, чтобы вывести слово «text», скетч должен выполнить команду:

```
lcd.print("text");
```

Для отображения содержимого переменных на экран ЖКИ используется вызов

```
lcd.print(variable);
```

При выводе переменной типа **float** необходимо указать количество десятичных знаков для вывода. Например, **lcd.print(pi, 3)** отобразит значение **pi** с тремя десятичными знаками, как показано на рис. 4.4:

```
float pi = 3.141592654;
lcd.print("pi: ");
lcd.print(pi, 3);
```

Рис. 4.4 — Отображение значения вещественной переменной на экран ЖКИ

Содержимое целочисленной переменой можно отобразить на экране ЖКИ не только в десятичном, но также в двоичном и шестнадцатеричном виде.

4.2.3. Определение собственных символов

Помимо букв, цифр и других символов из стандартного набора в каждом скетче можно определять и собственные символы. Обратите внимание, что на экране ЖКИ каждый символ отображается в виде матрицы, содержащей восемь рядов по пять точек, или пикселей. На рис. 4.5 эти матрицы показаны крупным планом.

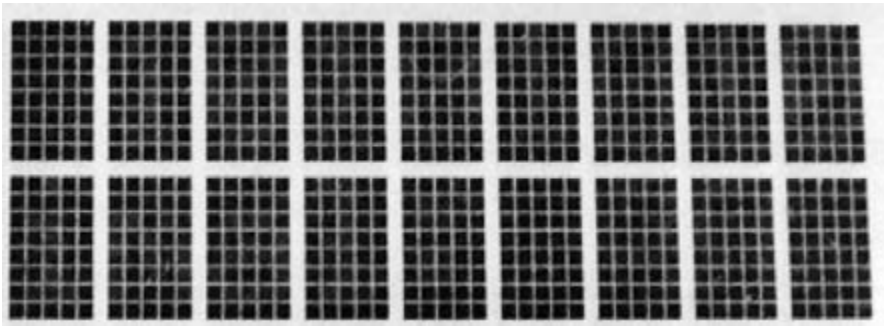


Рис. 4.5 — Набор символов ЖКИ в виде пикселей

Чтобы отобразить собственный символ, его нужно определить в виде массива. Например, символ-смайлик определяется следующим образом:

```
byte a[8] = { B00000,
          B01010,
          B01010,
          B00000,
          B00100,
          B10001,
          B01110,
          B00000 };
```

Каждая цифра в этом массиве соответствует пикселю: 0 — выключенному, 1 — включенному. Элементы массива представляют строки пикселей на экране.

LCD1620 дисплей поддерживает 8 новых символов (пронумерованных от 0 до 7) размером 5 на 8 пикселей. Чтобы использовать созданный символ, сохраненный в виде массива **a**, запишем в первый из восьми слотов, предназначенных для пользовательских символов, применив команду:

```
lcd.createChar(0, a); // сохранить массив a[8] в слот 0
```

Чтобы отобразить символ, запишем следующий вызов:

```
lcd.write(byte(0));
```

Пример отображения нестандартных символов показан в Листинге 4.2.

```
// Листинг 4.2
#include <LiquidCrystal.h>
LiquidCrystal lcd(4, 5, 6, 7, 8, 9);
byte a[8] = {   B00000,
                B01010,
                B01010,
                B00000,
                B00100,
                B10001,
                B01110,
                B00000 };
void setup()
{
   lcd.begin(16, 2);
   lcd.createChar(0, a);
}
void loop()
{
```

```
    lcd.write(byte(0)); // вывести нестандартный
// символ из слота 0 в следующую позицию курсора
}
```

После выполнения программы, показанной в Листинге 4.2, на ЖКИ будут отображены два ряда «смайликов».

4.2.4 Подключение ЖКИ, используя шину I²C

Взаимодействие с другими устройствами плата Arduino осуществляет посредством шины данных — системы соединений, позволяющей двум и более устройствам обмениваться данными упорядоченным образом.

I²C / IIC (Inter-Integrated Circuit), шина связи интегральных схем, обеспечивает устойчивую высокоскоростную двустороннюю связь между устройствами и требует минимального числа контактов ввода-вывода. Разработка принадлежит фирме Philips. Передача данных между устройствами и Arduino осуществляется по двум линиям, которые называют линией данных (Serial Data Line, SDA) и тактовой линией (Serial Clock Line, SCL).

В модели Arduino Uno линия SDA выведена на контакт A4, а линия SCL — на контакт A5.

Плата Arduino считается ведущим устройством, а все остальные устройства — ведомыми.

Каждое ведомое устройство имеет свой адрес — шестнадцатеричное число, — позволяющий плате Arduino обращаться и взаимодействовать с каждым устройством по отдельности.

Для подключения ЖКИ с помощью шины I²C необходим модуль-переходник, показанный на рис. 4.6.

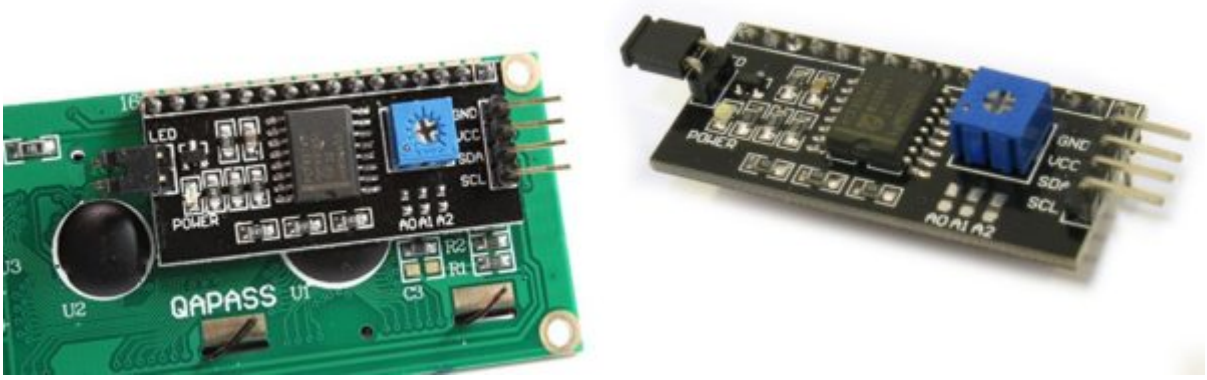


Рис. 4.6 — Модуль-переходник I²C для ЖКИ

Порядок подключения ЖКИ к плате Arduino по шине I²C приведен в таблице 4.2.

Таблица 4.2 — Порядок подключения ЖКИ к плате Arduino по шине I^{2}C

Arduino Uno	LCD i2c
GND	GND
5V	VCC
A4	SDA
A5	SCL

Схема подключения ЖКИ по шине I^{2}C показана на рис. 4.7.

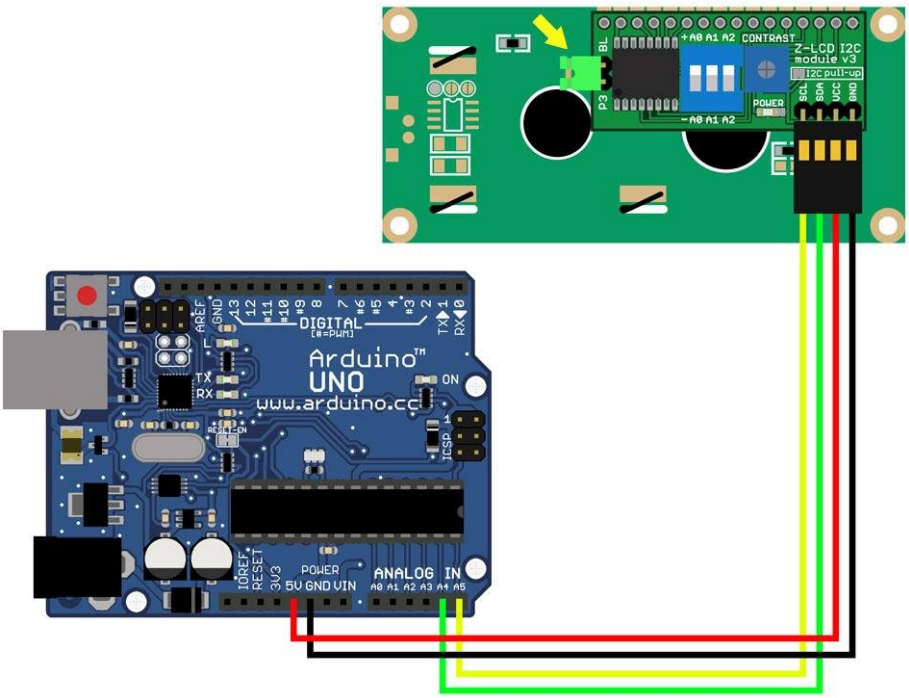


Рис. 4.7 — Схема подключения ЖКИ по шине I^{2}C

Для работы с ЖКИ по шине I^{2}C необходимо установить библиотеку LiquidCrystal_I2C.h.

Пример программы работы с ЖКИ по шине I^{2}C показан в Листинге 4.3.

```
// Листинг 4.3
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);  // Установка дисплея

void setup()
{
  lcd.init();
  lcd.backlight();// Включаем подсветку дисплея
  lcd.print("Hello, world!");
```

```
  lcd.setCursor(8, 1);
  lcd.print("LCD 1602");}

void loop()
{
   LCD.noDisplay(); // выключаем и включаем
                    // надпись на дисплее
   delay(1000);
   LCD.display();
   delay(1000);
}
```

Описание функций и методов библиотеки LiquidCrystal_I2C показано в таблице 4.3.

Таблица 4.3 — Описание функций и методов библиотеки LiquidCrystal_I2C

Название функции	Назначение функции
home(), clear()	первая функция позволяет вернуть курсор в начало экрана, вторая тоже, но при этом удаляет все, что было на мониторе до этого
write(ch)	позволяет вывести одиночный символ **ch** на экран
cursor(), noCursor()	показывает/скрывает курсор на экране
blink(), noBlink()	курсор мигает/не мигает (если до этого было включено его отображение)
display(), noDisplay()	позволяет подключить/отключить дисплей
scrollDisplayLeft(), scrollDisplayRight()	прокручивает экран на один знак влево/вправо
autoscroll(), noAutoscroll()	позволяет включить/выключить режим автопрокручивания. В этом режиме каждый новый символ записывается в одном и том же месте, вытесняя ранее написанное на экране
leftToRight(), rightToLeft()	Установка направление выводимого текста – слева направо или справа налево

4.3. Задания для практической работы

Задание 1

Выполните подключение LCD индикатора к Arduino Uno по схеме, показанной на рис. 4.2., используя симулятор TinkerCAD, и напишите скетч для

микроконтроллера Arduino Uno, который выводит на экран LCD индикатора вашу фамилию (первая строка) и ваше имя (вторая строка). Для вывода русских символов необходимо воспользоваться функциями создания собственных символов.

Выполните проверку работоспособности электрической схемы и скетча с помощью моделирования. Соберите электрическую схему с помощью макетной платы и покажите работоспособность схемы преподавателю.

Задание 2

Выполните подключение LCD индикатора к Arduino Uno по схеме, показанной на рис. 4.3., используя интерфейс I^2C, и напишите скетч в Arduino IDE, который отображает ваши фамилию и имя в виде бегущей строки слева направо.

Выполните проверку работоспособности электрической схемы и скетча с помощью макетирования и продемонстрируйте работу схемы преподавателю.

Задание 3

Используя подключение LCD индикатора к Arduino Uno из задания 2, выполните вывод на экран время в секундах и минутах с момента запуска программы, используя функцию **`millis()`**.

Выполните проверку работоспособности электрической схемы и скетча с помощью макетирования и продемонстрируйте работу схемы преподавателю.

Задание 4

Используя подключение LCD индикатора к Arduino Uno из задания 2, выполните посимвольную заливку экрана индикатора в соответствии с положением ручки потенциометра: крайнему левому положению соответствует отсутствие заливки, крайнему правому — заливка двух строк. Для заливки одного символа нужно создать символ, в котором все пиксели будут включенными. После заполнения первой строки начинает заполняться вторая строка.

Выполните проверку работоспособности электрической схемы и скетча с помощью макетирования и продемонстрируйте работу схемы преподавателю.

Задание 5

Используя подключение LCD индикатора к Arduino Uno из задания 2, выполните вывод на экран значения температуры и влажности, получаемые с помощью датчика DHT11. Сведения по подключению и работе с датчиком DHT11 приведены в приложении Б.

Выполните проверку работоспособности электрической схемы и скетча с помощью макетирования и продемонстрируйте работу схемы преподавателю.

4.4. Содержание отчета

4.4.1. Сформулировать цель работы.

4.4.2. Привести постановку задачи и текст задания.

4.4.3. Привести краткие теоретические сведения.

4.4.4. Привести принципиальную и/или функциональную схему устройства, соответствующую заданию.

4.4.5. Привести текст программы (программа должна содержать комментарии).

4.4.6. Привести фотографию собранного устройства.

4.4.7. Сделать выводы по работе.

4.5. Контрольные вопросы

4.5.1. Для чего применяются жидкокристаллические индикаторы? Опишите назначение выводов индикатора LCD 1602.

4.5.2. Перечислите и охарактеризуйте функции и методы библиотеки LiquidCrystal.h/ LiquidCrystal_I2C.h.

4.5.3. Как выводить на жидкокристаллический индикатор текст и числа.

4.5.4. Как создать свой символ для вывода на жидкокристаллический индикатор?

4.5.5. Опишите подключение устройств к интерфейсу I^2C.

4.5.6. Опишите порядок подключения и работы с жидкокристаллическим индикатором с помощью интерфейсной шины I^2C.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Оформление текстовых работ : методические указания / СевГУ ; сост. В.Г. Слёзкин, П.П. Ермолов. — Севастополь : Изд-во СевГУ, 2015. — 19 с.

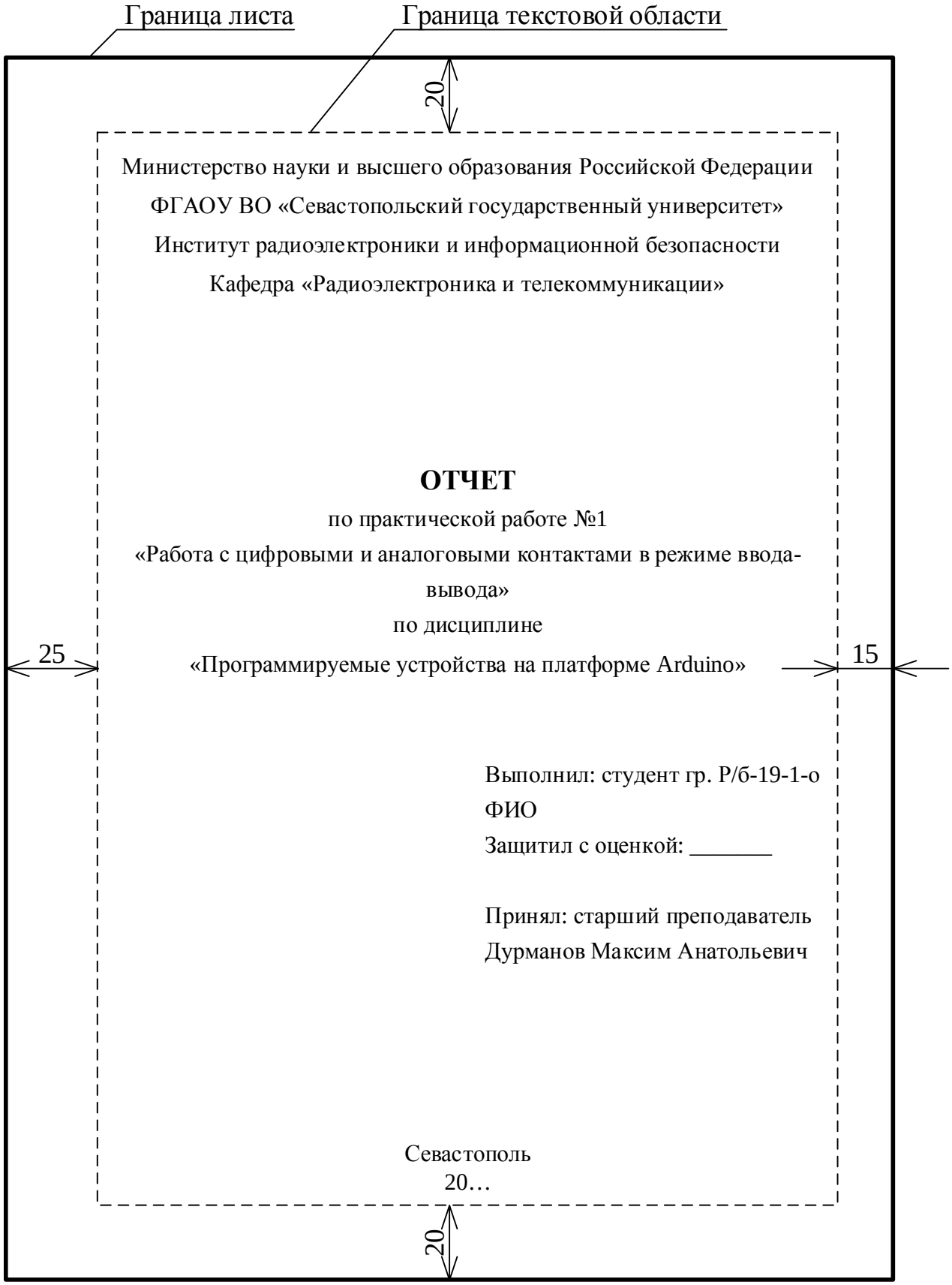
2. Быстрый старт. Первые шаги по освоению Arduino [Электронный ресурс] / MaxKit.ru. — 2015. — Режим доступа: maxKit.ru.

3. Бокселл, Дж. Изучаем Arduino. 65 проектов своими руками / Дж. Бокселл. — СПб. : Питер, 2017. — 400 с.

4. Блум, Дж. Изучаем Arduino: инструменты и методы технического волшебства. — СПб. : БХВ-Петербург, 2015. — 336 с.

ПРИЛОЖЕНИЕ А

ПРИМЕР ОФОРМЛЕНИЯ ТИТУЛЬНОГО ЛИСТА ОТЧЕТА ПО ПРАКТИЧЕСКОЙ РАБОТЕ



ПРИЛОЖЕНИЕ Б
ПОРЯДОК ПОДКЛЮЧЕНИЯ И РАБОТА С ДАТЧИКОМ DHT11

А.1. Общие сведения о датчике

DHT11 — это цифровой датчик, состоящий из термистора и емкостного датчика влажности. Наряду с невысокой стоимостью DHT11 имеет следующие характеристики: питание осуществляется от 3,5-5V, определение температуры от 0 до 50 градусов с точностью 2 град, определение влажности от 20% до 95% с 5% точностью.

Термистор — это термический резистор, сопротивление которого изменяется с температурой, т.е. увеличение температуры приводит к падению его сопротивления. Относится к измерительной технике и может быть использован для автоматического измерения температуры в различных средах.

Емкостной датчик влажности — это конденсатор с переменной емкостью, который содержит токопроводящие обкладки из медной фольги на текстолите. Этот конденсатор заключен в герметичный чехол, поверх которого расположен влагопоглощающий слой. При попадании частиц воды на этот слой, меняется его диэлектрическая проницаемость, что приводит к изменению емкости конденсатора.

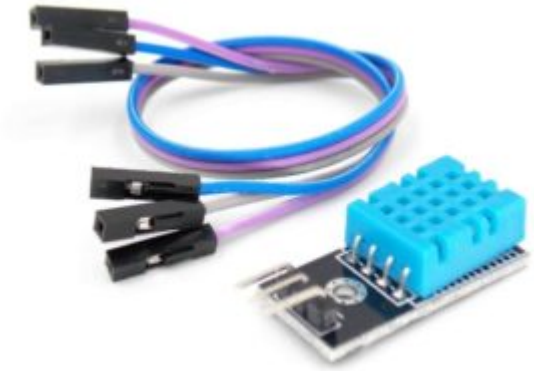


Рис. А.1 — Внешний вид датчика DHT11

Модуль DHT11 оборудован трех-четырех выводным разъемом и подключается по схеме:

GND (–) — подключается к выводу GND;

VDD (+) — подключается к выводу +5V;

DATA (S) — подключается к цифровому выводу Arduino.

А.1. Подключение датчика DHT11

Схема подключения модуля датчика DHT11 показана на рис. А.2. На схеме используется подтягивающий резистор сопротивлением 4,7 кОм.

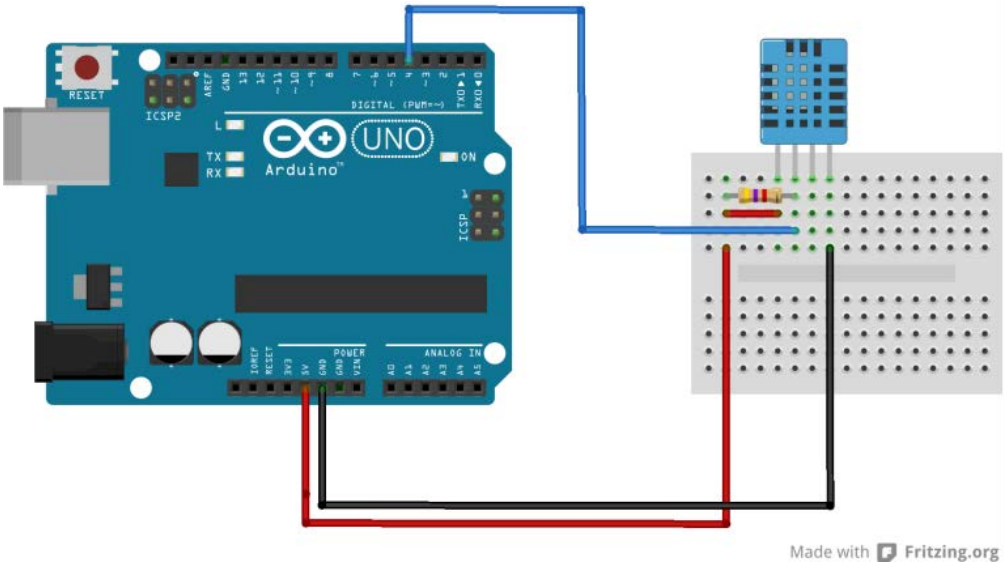


Рис. А.2 — Схема подключения датчика DHT11 к Arduino Uno

Перед написанием скетча необходимо скачать и установить библиотеку Dht.h (ссылка для скачивания: https://github.com/adafruit/DHT-sensor-library). Кроме того может понадобится библиотека Adafruit_Sensor.h (ссылка для скачивания: https://github.com/adafruit/Adafruit_Sensor).

Пример скетча для работы с датчиком DHT11 приведен в Листинге А.1.

Листинг А.1

```
#include "DHT.h"
#define DHTPIN 2 // номер цифрового вывода,
       //к которому подключен сигнальный вывод датчика
DHT dht(DHTPIN, DHT11);

void setup() {
  Serial.begin(9600);
  dht.begin(); // инициализация датчика
}
void loop() {
  delay(2000);
  float h = dht.readHumidity(); // измеряем влажность
  float t = dht.readTemperature();
                                  // измеряем температуру
  if (isnan(h) || isnan(t)) { // проверка чтения дан-
ных
    Serial.println("Ошибка считывания");
    return;
  }
  Serial.print("Влажность: ");
```

```
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Температура: ");
  Serial.print(t);
  Serial.println(" *C "); //Вывод показаний на экран
}
```