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

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к практикуму по дисциплине
«Программируемые устройства на платформе Arduino»

для студентов очной и заочной форм обучения всех направлений

Севастополь
2019

Методические указания 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, 2019. — ___ 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» «___» 2019 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 в интегрированной среде разработки Arduino IDE и составления электрических схем, а также знакомство с основными элементами 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 ее 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 и подготовить ответы на контрольные вопросы;

— собрать электрическую схему согласно заданию, написать скетч и запрограммировать микроконтроллер;

— продемонстрировать результаты работы сценария преподавателю;

— сделать снимки схемы в процессе работы;

— распечатать полученные результаты;

— 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 перечислено в методических указаниях 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 индивиду-

ально 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
РАБОТА С ЦИФРОВЫМИ И АНАЛОГОВЫМИ КОНТАКТАМИ В РЕЖИМЕ ВВОДА-ВЫВОДА

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).

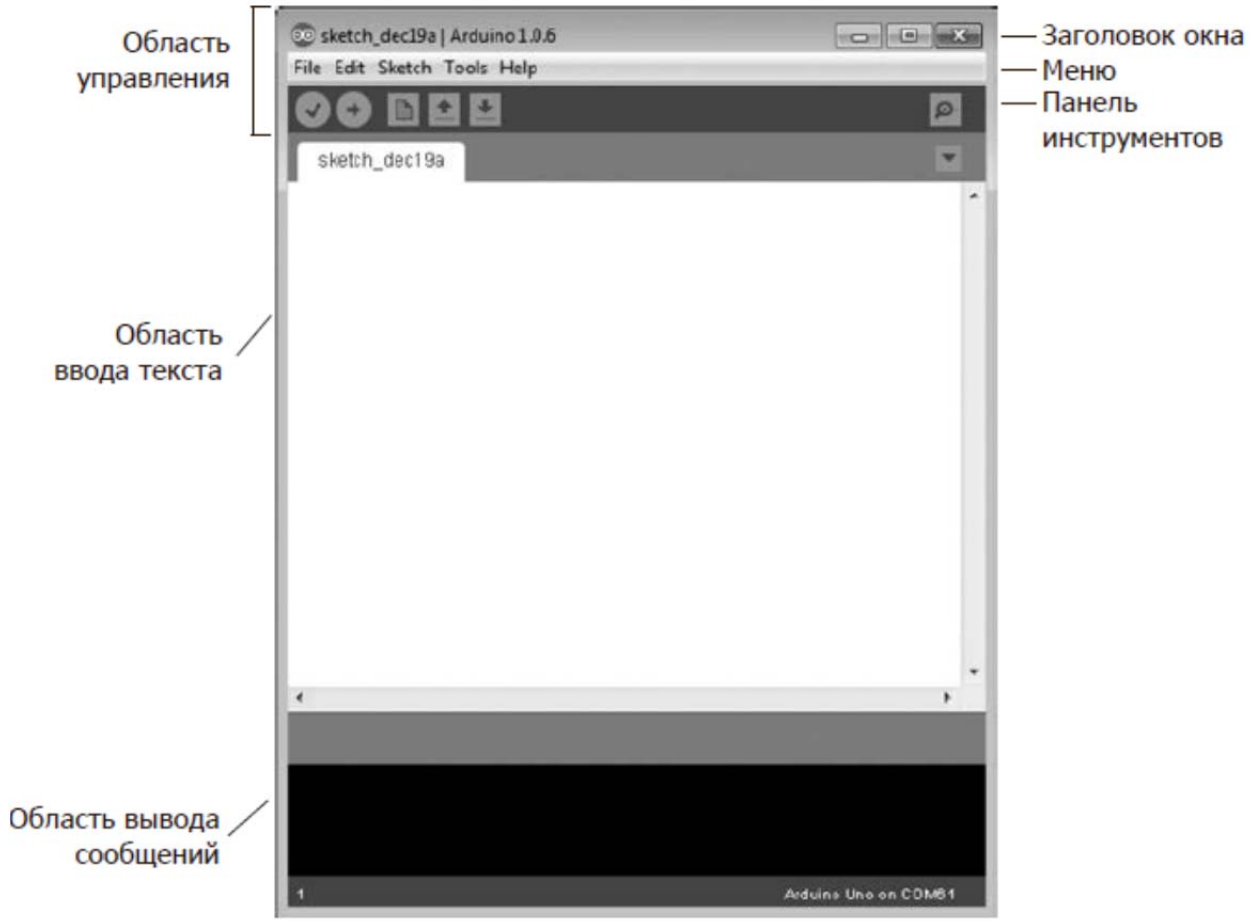


Рис. 1.1 — Окно среды разработки Arduino IDE

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 (Инструменты) 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 редакто-

ре.

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: внешнюю, называемую внешним участком цепи, и внутреннюю, называемую внутренним участком цепи.

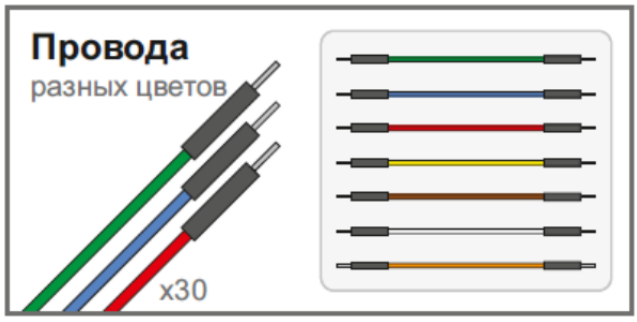
Внешний участок (внешняя цепь) состоит из одного или нескольких потребителей электрической энергии, соединительных проводов и различных приборов, включенных в эту цепь. Внутренний участок (внутренняя цепь) представляет собой сам источник электрической энергии.

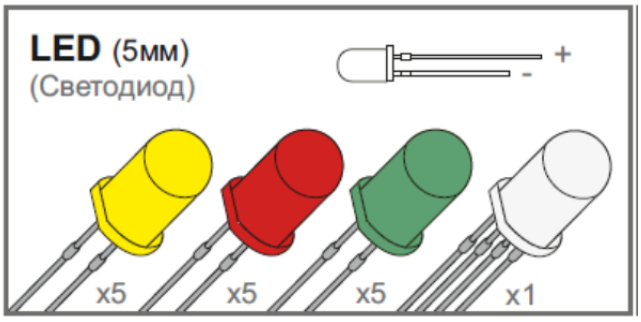
Источниками электрической энергии для питания радиотехнической аппаратуры служат гальванические элементы, аккумуляторы, генераторы и т. д.

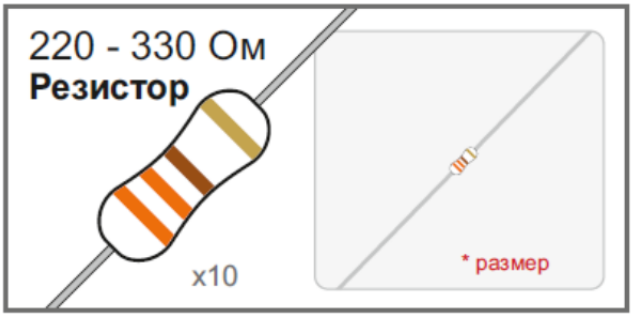
Потребителями электрической энергии в электро- и радиотехнических устройствах являются электродвигатели, сельсины, реле, электронно-лучевые трубки, дискретные элементы (резисторы, диоды, транзисторы ...), интегральные схемы и т.п.

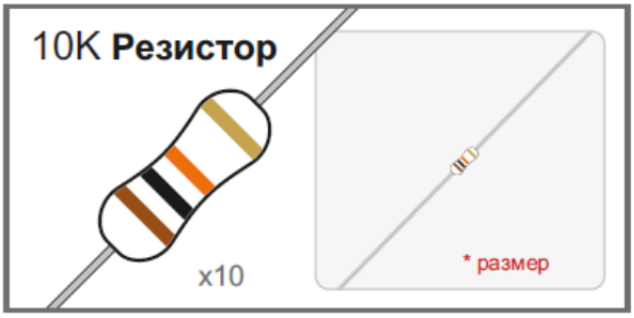
1.2.3. Компонентная база радиоэлектронных устройств

1.2.3.1. Основные компоненты электронных схем


Провода
разных цветов
x30


LED (5мм)
(Светодиод)
+
-
x5
x5
x5
x1


220 - 330 Ом
Резистор
x10
* размер


10K Резистор
x10
* размер


Потенциометр
x1
* или

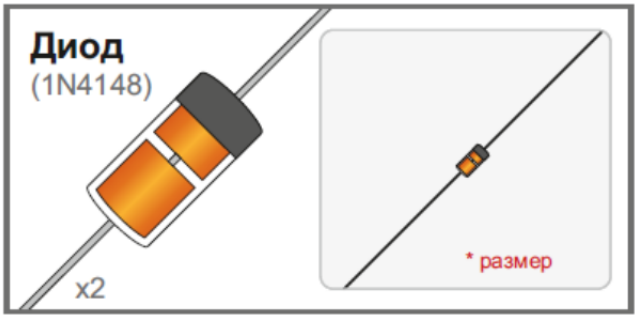
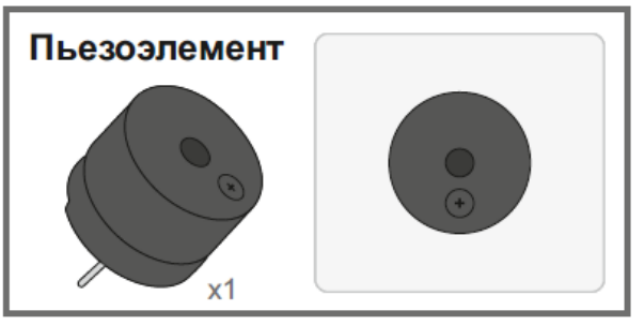
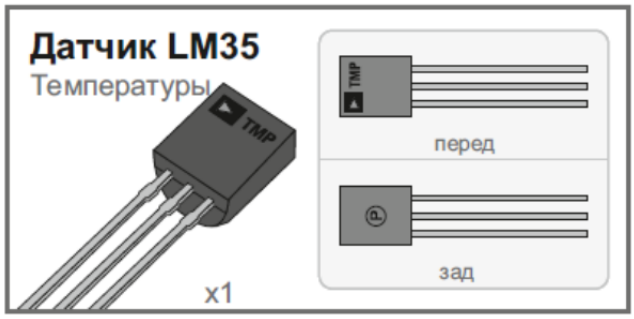
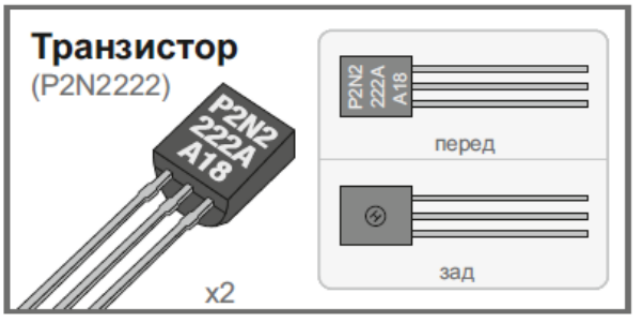
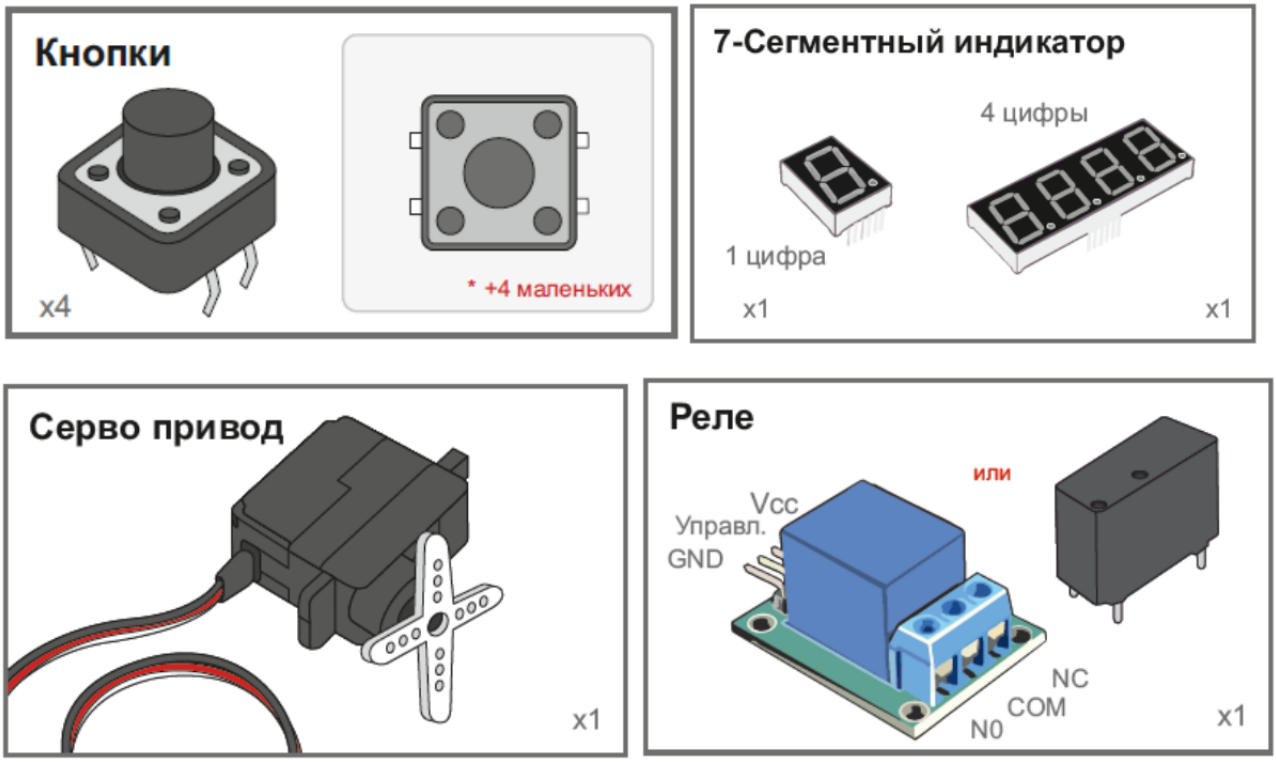

Диод
(1N4148)
x2
* размер


Фото резистор
x3


Пьезоэлемент
x1


Датчик LM35
Температуры
TMP
x1
TMP
перед
зад


Транзистор
(P2N2222)
P2N2
222A
A18
x2
P2N2
222A
A18
перед
зад



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.

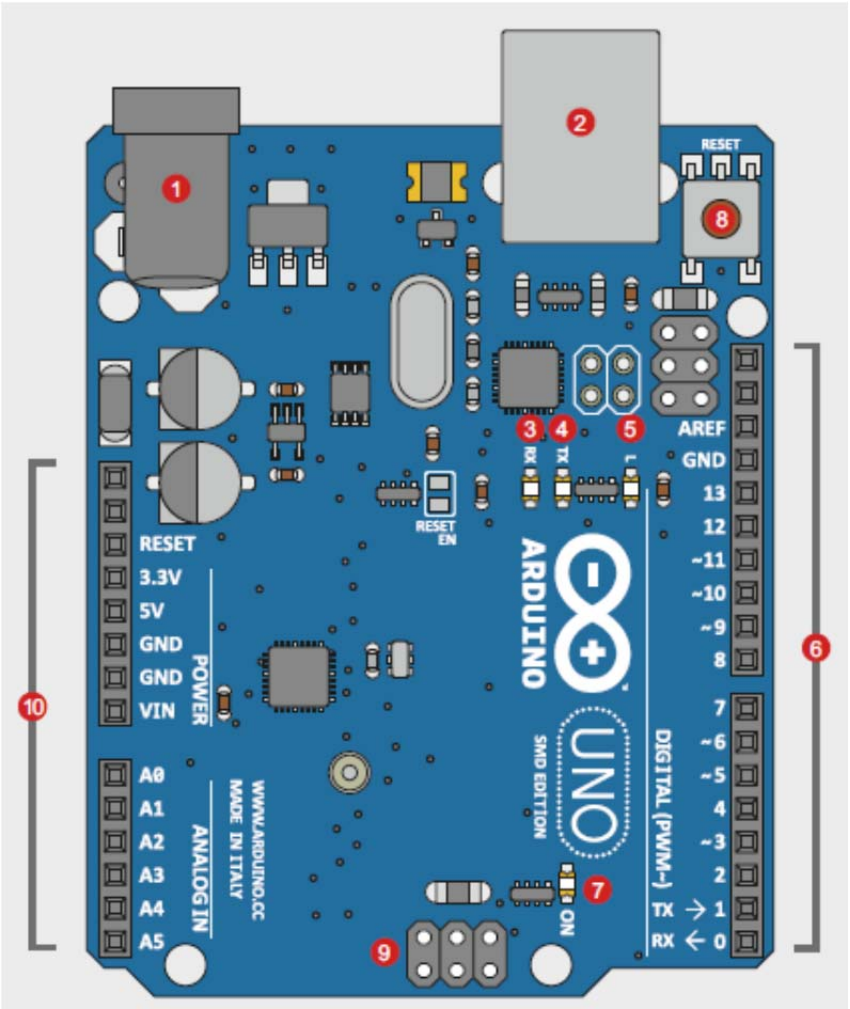


Рис. 1.2 — Внешний вид платы Arduino Uno на SMD компонентах

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

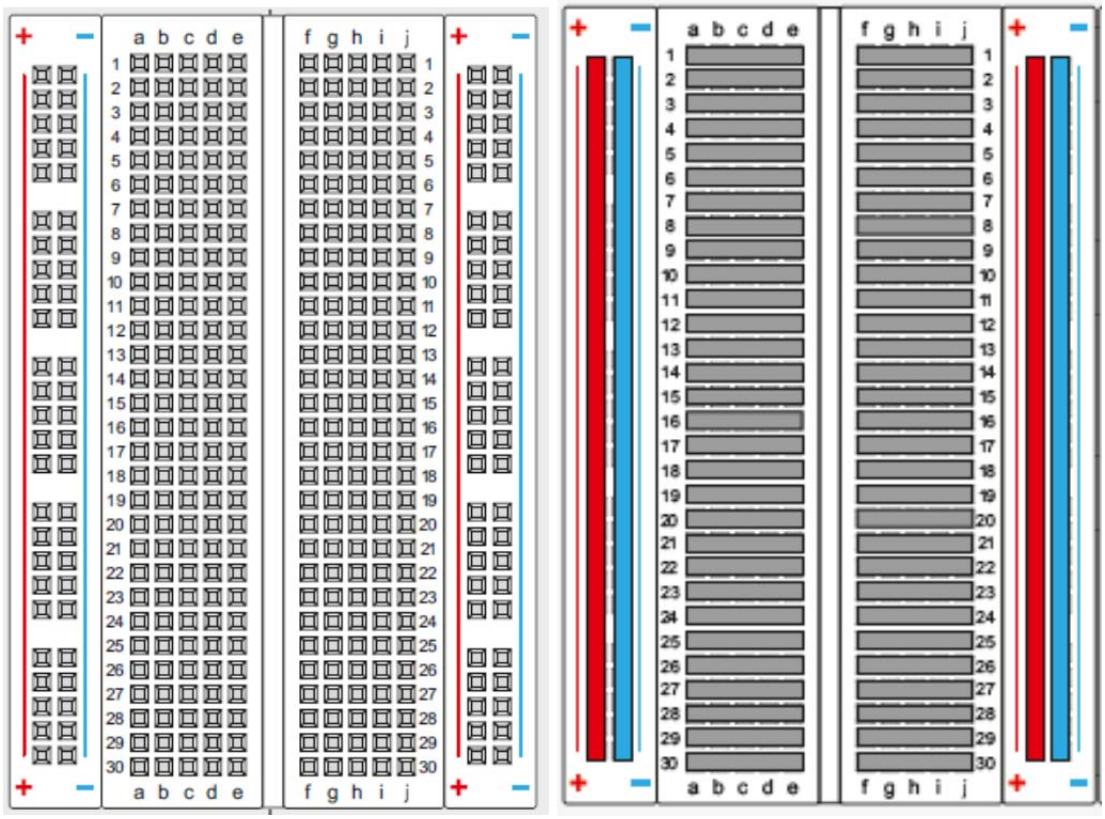


Рис. 1.3 — Внешний вид макетной платы и внутренней схемы соединения контактов

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

Прежде чем начать писать программу, 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.

Задание 1

Соберите схему с внешним светодиодом как показано на рис. 1.5.

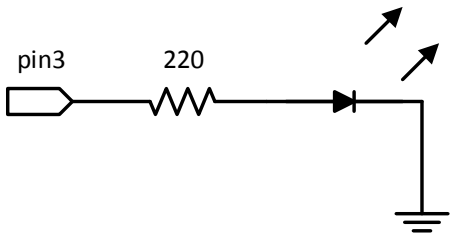


Рис. 1.5 — Схема подключения светодиода

Напишите программу, которая заставляет мигать светодиод с частотой 10/(n+1) Гц, где n — последняя цифра в зачетной книжке.

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. 1.6.

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. 1.7. показан линейный переменный резистор.

Рис. 1.6 — Обозначение переменного резистора

Рис. 1.7 — Внешний вид переменного резистора

Переменный резистор можно использовать для получения аналогового сигнала. Для этого переменный резистор нужно подключить, как показано на рис. 1.8, и загрузить код программы из Листинга 2.

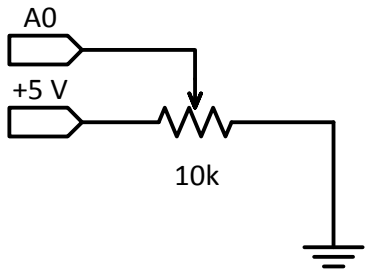


Рис. 1.8 — Схема подключения потенциометра

```
// 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.

Задание 2

Соберите схему, показанную на рис. 1.8, 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. 1.9). 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.

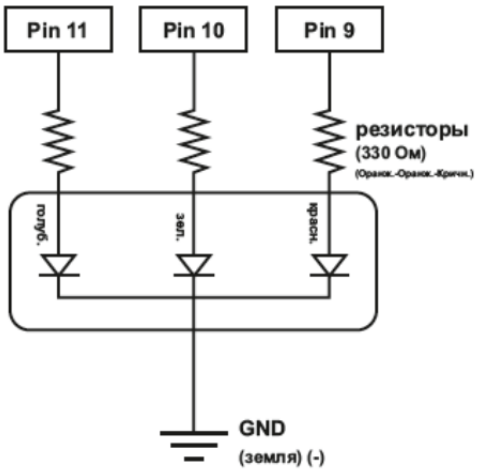


Рис. 1.9 — Схема RGB светодиода

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» соответствует 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. 1.10.

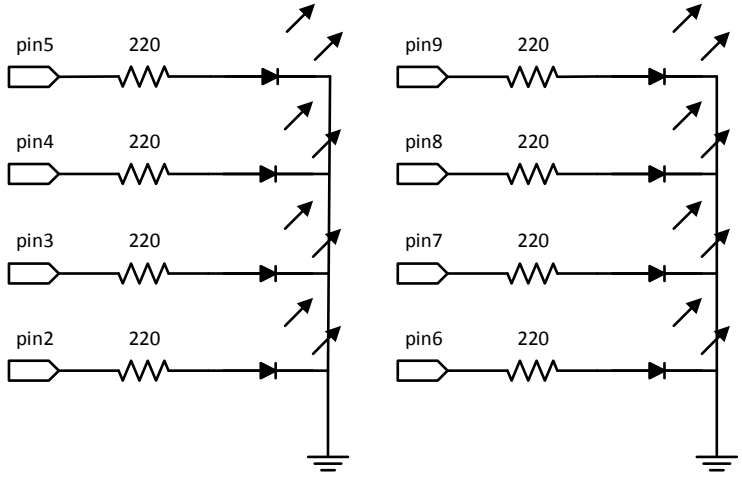


Рис. 1.10 — Схема подключения восьми светодиодов

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. 1.10). Напишите программу, которая включает и выключает светодиоды в заданной последовательности в соответствии с таблицей 1.1.

Варианты	**Задание**
1,6	Поочередное включение светодиодов (**`void oneOnAtATime()`**)
2,7	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()`**)
0,5	Случайное включение светодиодов с случайным интервалом (**`void randomLED()`**)

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. 1.11, а) и со стягивающим резистором (рис. 1.11, б).

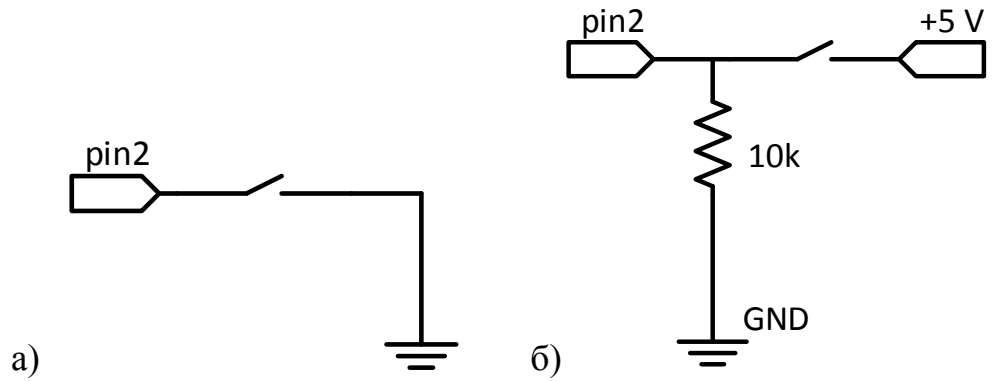


Рис. 1.11 — Схема подключения кнопки с подтягивающим резистором (а) и со стягивающим резистором (б)

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.

ПРИЛОЖЕНИЕ А

ПРИМЕР ОФОРМЛЕНИЯ ТИТУЛЬНОГО ЛИСТА ОТЧЕТА ПО ПРАКТИЧЕСКОЙ РАБОТЕ