МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждениевысшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению практических работ
по дисциплине «Промышленный Интернет-вещей»
Для студентов направления подготовки 09.03.02 Информационные системы и технологии, направленность (профиль) Цифровые технологии химических производств

(ЭЛЕКТРОННЫЙ ДОКУМЕНТ)

Невинномысск 2025

Содержание

Тема 1: Общие положения интернета вещей (IoT Basics)

Введение....12
Состав комплекта....14
Немного о макетной плате, резисторах и безопасности....23
Практическое занятие 1. Hello, world!....28
Практическое занятие 2. Эксперимент с мигающим светодиодом....30
Практическое занятие 3. Эксперимент с контролируемой потенциометром яркостью свечения светодиода через порт PWM....31
Практическое занятие 4. Эксперимент с внешним мигающим светодиодом....34
Практическое занятие 5. Эксперимент с рекламной расцветкой....36
Практическое занятие 6. Светофорный эксперимент....38
Практическое занятие 7. Эксперимент с пищалкой....40
Практическое занятие 8. Эксперимент с датчиком наклона....42
Практическое занятие 9. Эксперимент с чистым входным сигналом....44
Практическое занятие 10. Расширенный эксперимент с чистым сигналом....47
Практическое занятие 11. Эксперимент по чтению аналогового значения....49
Практическое занятие 12. Эксперимент по управлению звуком и светом....52
Практическое занятие 13. Эксперимент с датчиком огня....54
Практическое занятие 14. Эксперимент с вольтметром....57
Практическое занятие 16. Эксперимент с температурным сенсором....62
Практическое занятие 18. Эксперимент с одноразрядным цифровым светодиодным индикатором....66
Практическое занятие 19. Эксперимент с четырёхразрядным цифровым светодиодным индикатором....71
Практическое занятие 20. Эксперимент со светодиодной матрицей....77
Практическое занятие 21. Эксперимент с трёхцветным светодиодом....83
Практическое занятие 24. Часы реального времени DS1307....93
Практическое занятие 25. Эксперимент с датчиком уровня воды....97
Практическое занятие 27. Эксперимент с релейным модулем....102

Практическое занятие 28. Эксперимент с жидкокристаллическим монитором LCD1602A....104
Практическое занятие 29. Эксперимент с шаговым двигателем....107
Практическое занятие 30. Эксперимент с серводвигателем....110
Практическое занятие 31. Эксперимент с игровым джойстиком....113

Тема 2. Радиочастотная идентификация (RFID)

Практическое занятие 33. Эксперимент с RFID-модулем RC522....120
Практическое занятие 34. Эксперимент с системой контроля доступа....123

Тема 3. Беспроводные сенсорные сети (WSN)

Практическое занятие 15. Эксперимент с распознаванием голоса....59
Практическое занятие 32. Эксперимент с инфракрасным пультом дистанционного управления....115

Тема 4. Межмашинные коммуникации (M2M)

Практическое занятие 22. Эксперимент с модулем 74HC595....86
Практическое занятие 23. Кнопочный модуль 4×4 и библиотеки....89

Тема 4. Межмашинные коммуникации (M2M)

Практическое занятие 17. Разноцветный термостат....64
Практическое занятие 26. Эксперимент с сенсором температуры и влажности DHT11....100

Введение

Arduino – наиболее популярная платформа для разработки как простых, так и достаточно сложных проектов для интернета вещей. Популярность этой платформы обусловлена не только её низкой ценой, но и огромным количеством обучающих материалов, примеров проектов в сети, в том числе и на таких ресурсах, как YouTube, различных форумов разработчиков, а также хорошим официальным сайтом [2, 3], на котором Arduino Team постоянно предлагает ознакомиться с новыми проектами на базе одной из плат Arduino. Существует множество разновидностей плат Arduino, например: Uno, Leonardo, 101, Mega, Zero, Ethernet, Gemma, MKR FOX 1200 и т. д. Все платы на официальном сайте разделены на категории: начальный уровень (для обучения, к нему относится Uno), платы с расширенными функциями, платы для интернета вещей, платы для носимых устройств (в основном в тандеме с Lillypad, например для умной одежды). Несмотря на простоту среды Arduino IDE и её недостатки, в ней можно разрабатывать интересные и сложные проекты; кроме того, существует официальный онлайн-аналог среды и многочисленные библиотеки, способные нарастить её небольшой функционал. Именно поэтому для данного учебного пособия выбрана эта платформа, а конкретно Arduino Uno версии R3, и наиболее богатый и разнообразный в отношении количества различных модулей, сенсоров, датчиков и других компонентов комплект с этой платой – комплект интернет-вещей для начинающих на базе Arduino Uno версии R3 (Starter Learning Kit) с RFID-модулем [1].

Arduino Uno Rev. 3 – лучшая плата для начинающих. Согласно официальному сайту, эта плата является наиболее используемой и обладает наибольшим количеством документации из всех плат семейства Arduino. Arduino Uno – это плата на основе 8-битного микроконтроллера ATmega328P, см. рис. 1 – самая большая чёрная деталь на плате. Таким образом, по сути, плата Arduino Uno является платой расширения, или платой разработчика (developer board), сердце которой – ATmega328P.

Рис. 1 ❖ Лицевая сторона платы Arduino Uno

На плате есть и другой микроконтроллер – ATmega16U2, служащий для связи микроконтроллера ATmega328P и USB порта платы (на рис. 1 – чёрный квадрат слева от TX и RX). У Arduino Uno есть 14 цифровых выходов (пинов) ввода/вывода, обозначенных на плате цифрами (из них 6 ШИМ (PWM, Pulse-Width Modulation) выходных пинов широтно-импульсной модуляции, обозначенных символом ~), 6 аналоговых входов – A0-A6 (с разрешением в 10 бит, т. е. 1024 различных значения), кварцевый кристалл-резонатор на 16 Мгц, выход USB-Bf (на рис. 1 – слева вверху под кнопкой перезагрузки), выход для подключения питания от адаптера питания (7–12 В) или батарейки на 9 В (обычно – в виде прямоугольного параллелепипеда; выход находится слева внизу на рис. 1), ICSP-разъём (In Circuit Serial Programming, программирование по последовательному протоколу чипа, уже подключённого в некоторую схему, или просто – программирование контроллера внутри схемы; на рис. 1 – посередине правого края) и кнопка перезагрузки (слева вверху, см. рис. 1). Кроме этого, плата располагает тремя пинами земли (GND), одним пином на 5 В, одним – на 3.3 В, Vin-пином для подключения внешнего источника питания или для получения напряжения, если плата подключена к внешнему адаптеру питания через разъём питания (через USB-соединение питание ограничивается 5 В), IOREF-пином (Input Output Reference – информация о напряжении микроконтроллера) и встроенным светодиодом L (или 13).

Микроконтроллер ATmega328 на Arduino Uno поставляется уже с загрузчиком (bootloader), что позволяет загружать код без использования внешнего программатора. При желании можно обойти загрузчик и запрограммировать микроконтроллер через ICSP. ATmega328 обладает 32 Кб встроенной памяти (0.5 Кб из которых отведено загрузчику). В дополнение ко всему некоторые пины платы имеют несколько функций, например пины 2 и 3 могут использоваться для вызова внешних прерываний при некоторых событиях, например при событии смены высокого сигнала на пине на низкий (falling edge). Мы не будем углубляться в мельчайшие подробности характеристик платы – приведённой информации вполне достаточно для знакомства с платой и выполнения 34 практических заданий части 1 этого учебного пособия. Также здесь не будет приведена распиновка платы, так как, по сравнению с платой Raspberry Pi 3, с которой мы познакомимся в части 2 этого учебного пособия, для Arduino Uno делать распиновку нет смысла – все пины уже подписаны на плате, см. рис. 1.

Далее рассмотрим, что же входит, помимо самой платы Arduino Uno R3, в состав упомянутого комплекта интернет-вещей для начинающих на базе Arduino Uno версии R3 (Starter Learning Kit) с RFID-модулем. Этот комплект был выбран также и потому, что позволяет сделать очень много практических заданий на основе различных компонентов и устройств, которые в него входят: конечно же, число возможных проектов на его основе далеко не ограничивается 34 экспериментами, приведёнными в этой части учебного пособия, а ограничивается лишь фантазией разработчика. Практические задания предусмотрены для

всех компонентов этого комплекта, чтобы познакомить читателя со всеми возможными составными частями набора и их функциями. В следующем разделе приведены реальные фотографии компонентов комплекта, как они выглядят на самом деле, а не схематичные изображения или фотографии из интернета.

Состав комплекта

Комплект интернет-вещей Arduino Uno Starter Learning Kit с RFID-модулем [1] состоит из следующих компонентов, показанных на рис. 2–38.

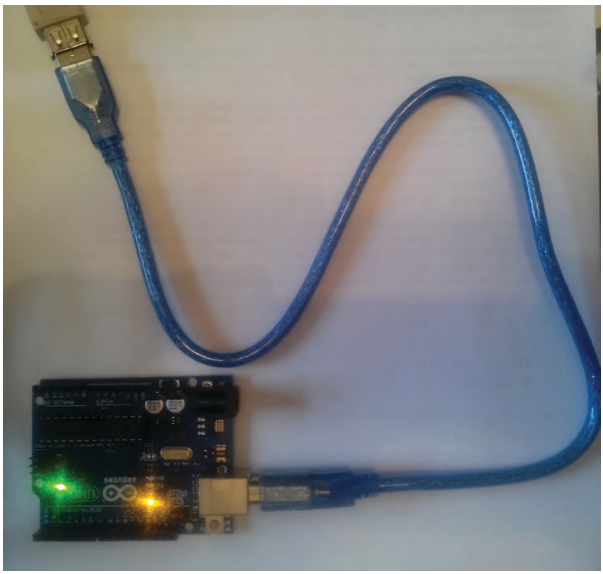


Рис. 2 ❖ Плата Arduino Uno R3 + USB-кабель (Ам-Вм)

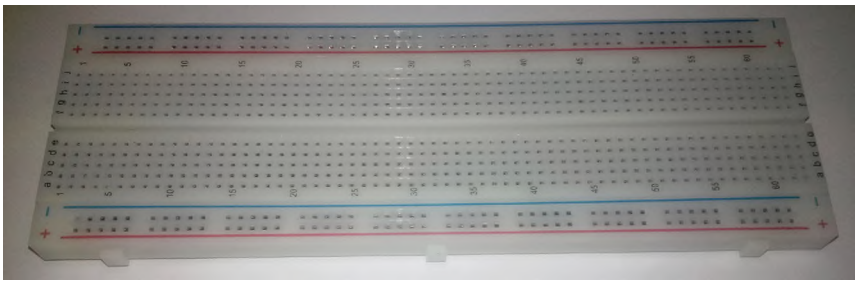


Рис. 3 ❖ Макетная плата (breadboard)

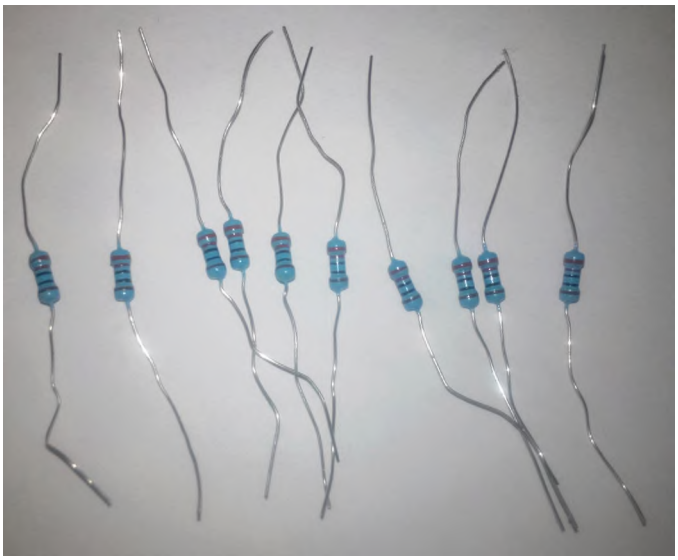


Рис. 4 ❖ Резисторы на 220 Ом, 10 шт.

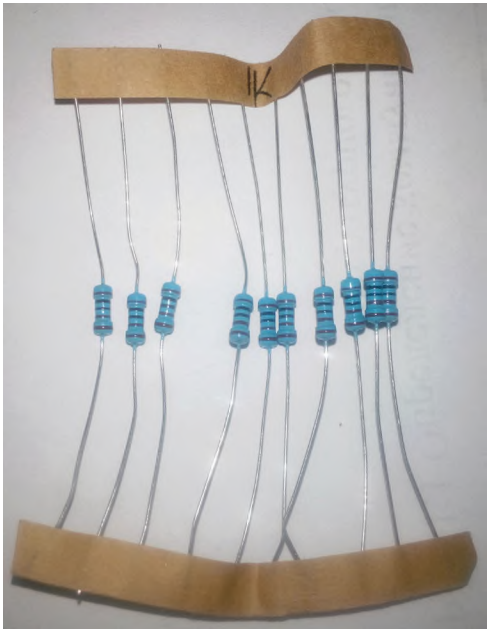


Рис. 5 ❖ Резисторы на 1 кОм, 10 шт.

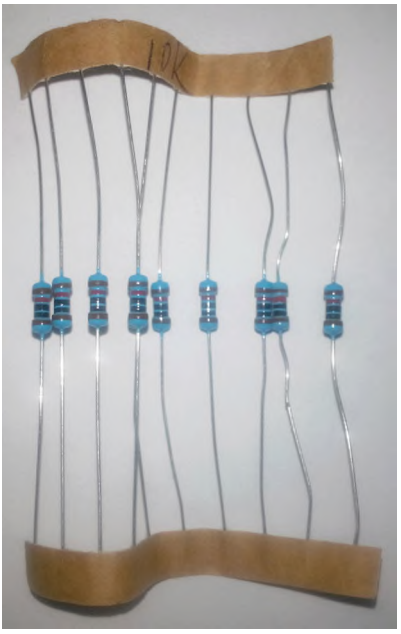


Рис. 6 ❖ Резисторы на 10 кОм, 10 шт.

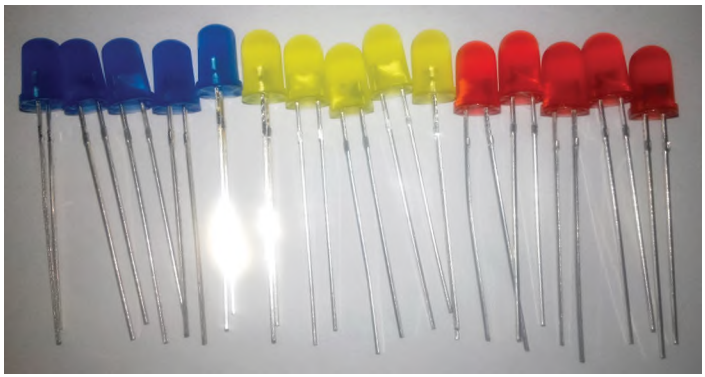


Рис. 7 ❖ Светодиоды, 15 шт. (5 синих, 5 жёлтых, 5 красных)

Рис. 8 ❖ Кнопки, 4 шт.

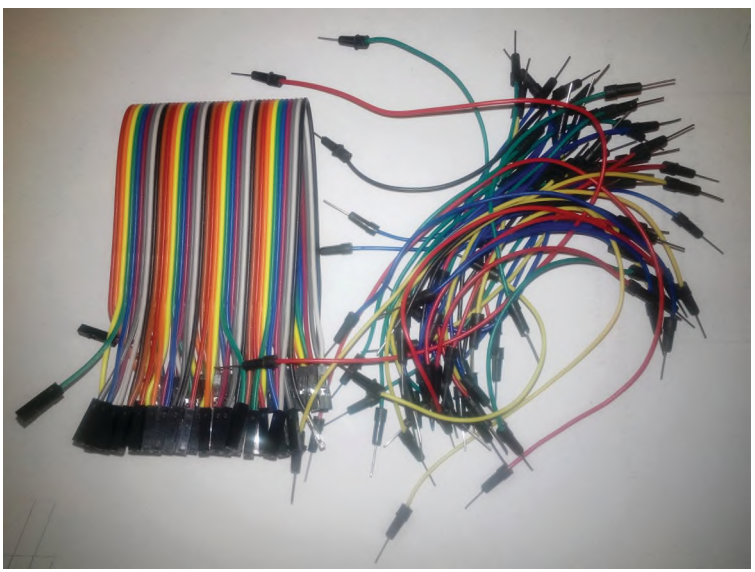


Рис. 9 ❖ Соединительные провода: m-m и f-f

Рис. 10 ❖ Шнур питания от батарейки на 9 В для Arduino Uno

Рис. 11 ❖ Динамик-пищалка, 2 шт.

Рис. 12 ❖ Датчик наклона (tilt switch), 2 шт.

Рис. 13 ❖ Потенциометр (10 КОм)

Рис. 14 ❖ Фоторезистор, 2 шт.

Рис. 15 ❖ Инфракрасный датчик огня

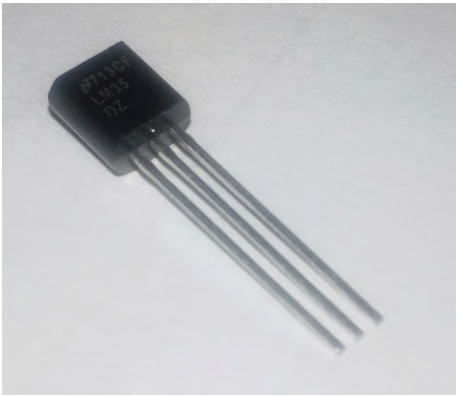


Рис. 16 ❖ Температурный сенсор LM35

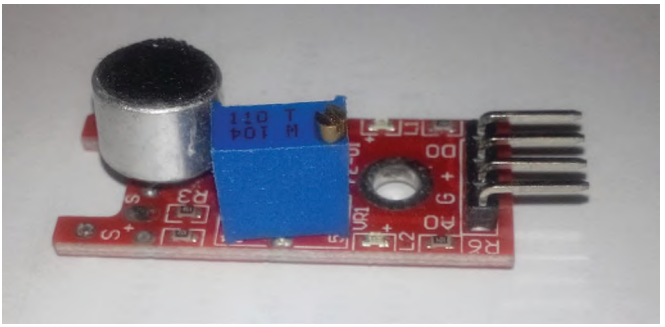


Рис. 17 ❖ Датчик уровня шума

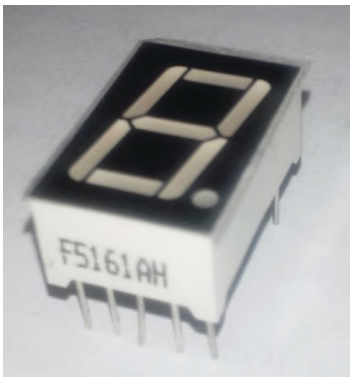


Рис. 18 ❖ Одноразрядный цифровой светодиодный индикатор

Рис. 19 ❖ Четырёхразрядный цифровой светодиодный индикатор

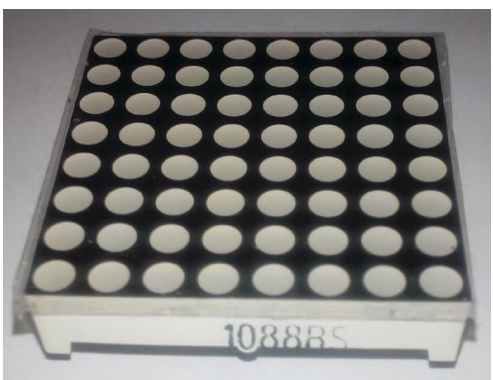


Рис. 20 ❖ Светодиодная матрица 8×8

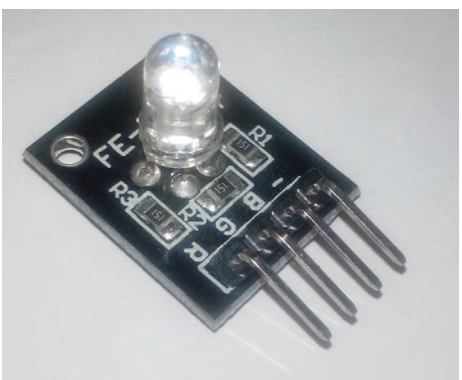


Рис. 21 ❖ Трёхцветный светодиод с общим катодом (на модуле)

Рис. 22 ❖ Модуль 74HC595

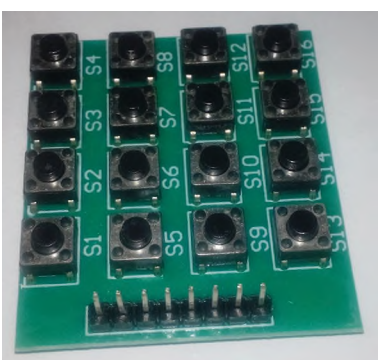


Рис. 23 ❖ Кнопочный модуль 4×4

Рис. 24 ❖ Часы реального времени RTC DS1307

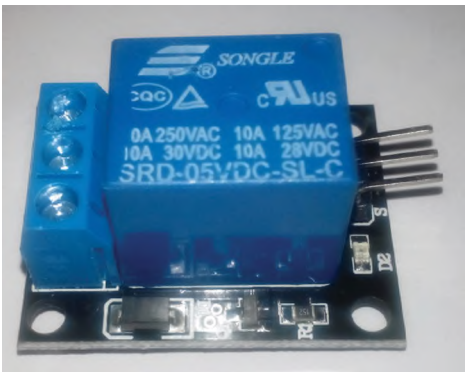


Рис. 25 ❖ Релейный модуль

Рис. 26 ❖ Датчик уровня воды (Water Sensor)

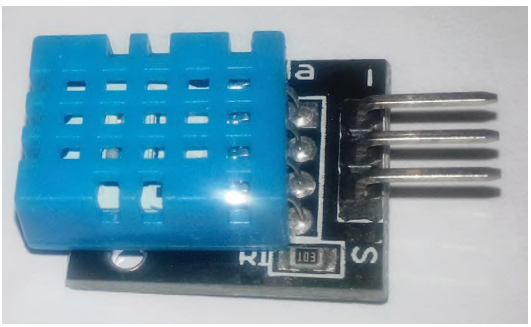


Рис. 27 ❖ Сенсор температуры и влажности DHT11

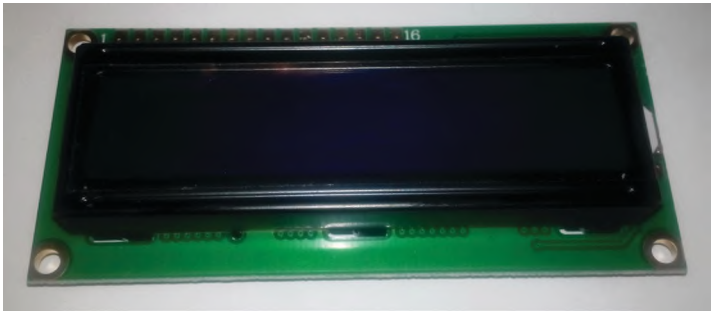


Рис. 28 ❖ Жидкокристаллический монитор (дисплей) LCD1602A

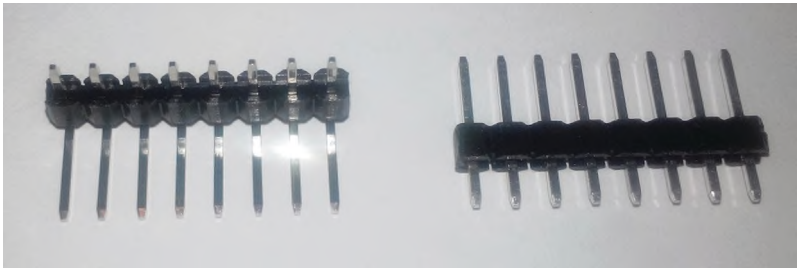


Рис. 29 ❖ Два штырьковых коннектора по 8 пинов каждый

Рис. 30 ❖ Шаговый двигатель

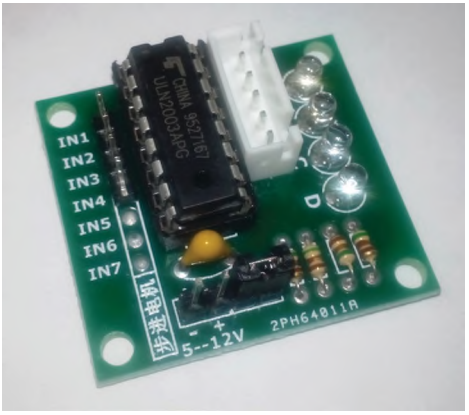


Рис. 31 ❖ Модуль для шагового двигателя

Рис. 32 ❖ Серводвигатель (сервопривод) с комплектом насадок

Рис. 33 ❖ Игровой джойстик

Рис. 34 ❖ Инфракрасный пульт дистанционного управления NEC

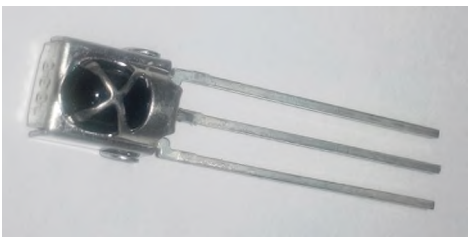


Рис. 35 ❖ Инфракрасный приёмник

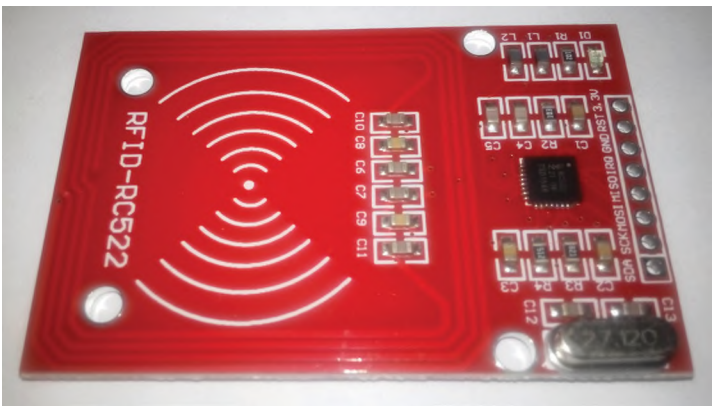


Рис. 36 ❖ RFID-модуль RC522

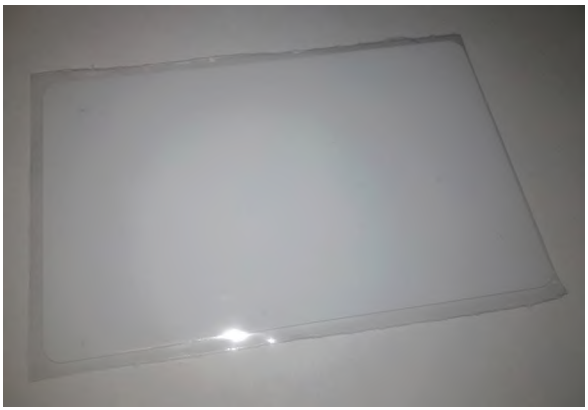


Рис. 37 ❖ RDIF-карта

Рис. 38 ❖ RFID-ключ

Немного о макетной плате, резисторах и безопасности

Как в этой, так и во второй части книги вам понадобятся знания о том, что такое макетная плата, как включать в схему светодиод и как отличить один резистор от другого. Но сначала – о безопасности.

Главное правило обращения с электричеством, компонентами и модулями гласит: помните, что как вы можете повредить технику, так и она может нанести вам вред! Перед тем как выполнять задания, нужно помнить о простых правилах работы с электронными компонентами и тем более системами на модуле (SoM, System on Module), к которым относится Arduino Uno, платами и прочими электронными изделиями:

- собирать и разбирать/менять схему можно только при выключенном питании (отсоединённом USB-кабеле) – имейте терпение;
- светодиоды и другие чувствительные компоненты подключаются строго согласно схеме – через резисторы;
- не стоит путать питание с землёй, плюс с минусом;
- никакого статического электричества! Если на вас свитер из синтетики или шерсти, если вы любите часто поправлять свои волосы, заземляйте свои руки, перед тем как дотрагиваться до электронных изделий (дотроньтесь до корпуса компьютера, железной ножки стола, батареи и т. д.)!

Макетная плата – удобное средство для соединения электрических компонентов в простые схемы и даже в схемы среднего уровня сложности. Макетные платы бывают разных типов, но в основном выделяют два типа: с разрывом горизонтальных линий земли и питания сверху и снизу посередине и без разрыва. В комплекте вам могут попасться оба типа. Макетная плата без разрыва горизонтальных линий земли и питания сверху и снизу показана на рис. 3. Если разрыва нет, это явно показывается синими и красными линиями: на

рис. 3 линии идут непрерывно, значит, разрыва нет. В случае наличия разрыва красные и синие линии прерываются посередине платы.

Соединения макетной платы без разрыва линий земли и питания показаны на рис. 39. Соединения макетной платы с разрывом этих линий, соответственно, проходят снизу и сверху по горизонтали от краёв только до середины макетной платы. Что же касается вертикальных соединений, у макетных плат обоих рассмотренных типов соединения прерываются 3 раза по вертикали: между зелёными разъёмами верхнего и нижнего рядов на рис. 39 нет соединения, так же, как и между зелёными и красными рядами.

Все принципиальные схемы и схемы с макетной платой в этой книге нарисованы с помощью наиболее распространённой и популярной открытой библиотеки + редактора электронных компонентов и схем Fritzing [10].

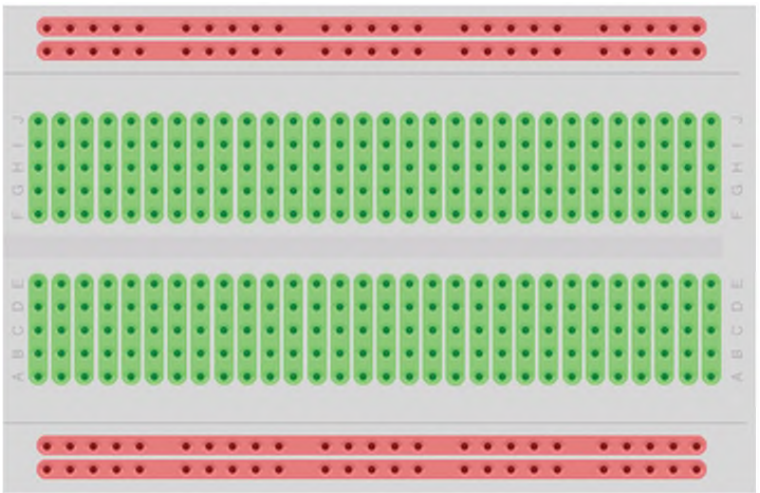


Рис. 39 ❖ Соединения макетной платы без разрывов верхних и нижних горизонтальных линий питания и земли

Теперь – светодиод. У каждого светодиода есть короткий и длинный выходы (пины) – см. рис. 7. Они представляют, соответственно, катод и анод. На схеме у светодиодов эта особенность выражена следующим способом: более длинный пин (анод) изогнут у основания цветной колбы светодиода, более короткий же (катод) входит в колбу прямо, без изгиба (см., например, рисунок с макетной платой к практическому занятию 3 в этой части учебного пособия). При подключении светодиода надо помнить, что ток по нему может протекать только в одном направлении – от анода к катоду (светодиод – вид диода, который работает только в одном направлении), таким образом, анод (длинный пин) всегда подключается к источнику питания или управляющему сигналу, тогда как катод (короткий пин) обычно подключается к земле через сопротив-

ление, ограничивающее ток, протекающий через светодиод. Светодиод нельзя подключать без сопротивления, иначе он может сгореть. Чем больше сопротивление в схеме со светодиодом (от 220 Ом до 10 кОм), тем меньше яркость свечения светодиода (меньший ток проходит через него). Подробнее о светодиодах можно почитать, например, в источнике [12].

И наконец, резисторы. На схемах резисторы обозначаются следующим образом (слева – в англоязычных источниках, справа – в русскоязычных источниках):

Рис. 40 ❖ Обозначение резисторов на принципиальных схемах [11]

У каждого резистора есть номинал: 220 Ом, 1 кОм и т. д. Резисторы, входящие в комплект с Arduino Uno и в другие комплекты с иными платами, обладают цветовыми насечками, см., например, рис. 4–6. Каждый цвет обозначает цифру, от 0 до 9, и цветовых насечек на резисторе несколько: таким образом можно определить номинал резистора, пользуясь правилами, изображёнными на рис. 41 [11]. Подобные резисторы, несмотря на их размер, всё же встречаются в реальных схемах, используемых в промышленности: например, управляющая плата холодильника Whirlpool собрана с помощью таких резисторов, поскольку холодильник большой и делать миниатюрную плату с применением сверхточных технологий, которые используются при производстве материнской платы для настольного компьютера, не имеет смысла.

Обладая знаниями из таблицы, взятой с сайта [11] и изображённой на рис. 41, вы можете подсчитать номиналы и точность резисторов, входящих в комплект с Arduino Uno и показанных на рис. 4, 5 и 6. На рис. 42 даны примеры резисторов и их номиналов.

Резисторы для схем бывают двух типов (здесь мы для простоты не говорим о специфических типах резисторов – потенциометрах, термисторах, варисторах, фоторезисторах и т. д., хотя некоторые из них встретятся нам в этой части книги): включённые последовательно в электрическую цепь (series resistors) по отношению к пинам платы, и – параллельно, которые, в свою очередь, подразделяются на стягивающие и подтягивающие резисторы (pull-down и pull-up resistors). Значения последовательных резисторов обычно варьируются от 100 до 300 Ом, стягивающих и подтягивающих – от 1 кОм до 10 кОм. Последовательные резисторы подключаются в электрическую цепь, чтобы защитить оборудование и, главное, пины платы от больших значений тока: например, так подключается светодиод через резистор номиналом 220 Ом, чтобы светодиод не перегорел. Каждый пин платы обладает ограниченной способностью быть источником (высокое значение, логическая 1) или приёмником (низкое значение, логический 0) электрического тока через электрическую схему, под-

ключённую к нему. Периферия, которая в схеме потребляет большие значения тока – больше, чем может позволить себе пин платы, даже если это происходит очень короткое время, может повредить пин на плате – именно поэтому в схему последовательно включается ограничивающий ток резистор, например как показано на рис. 43 в схеме со светодиодом [19], где OUT – это пины платы, а Vcc – источник питания.

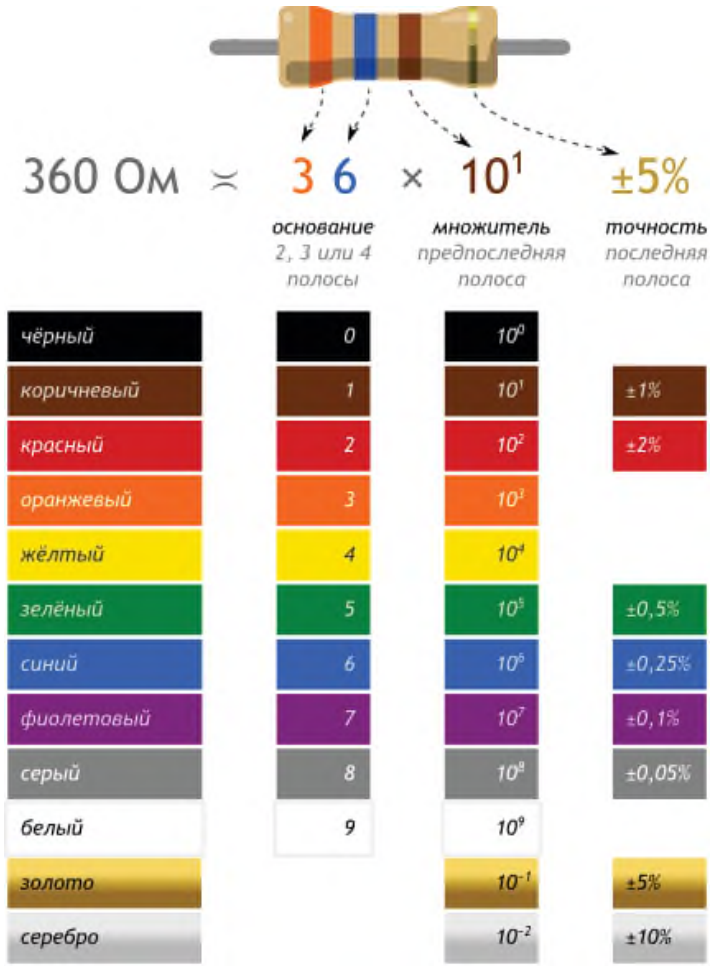


Рис. 41 ❖ Правила подсчёта номинала резистора и его точности [11]

Рис. 42 ❖ Примеры резисторов и их номиналов [11]

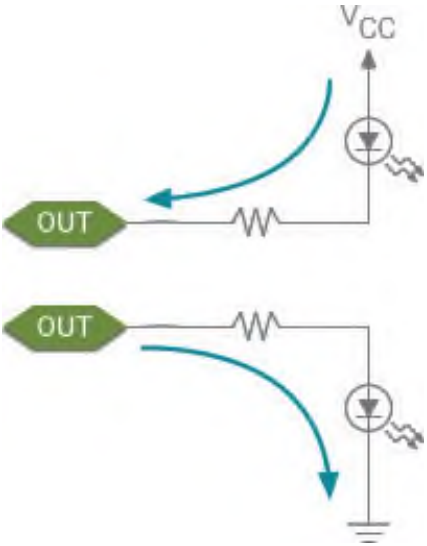


Рис. 43 ❖ Пример подключения последовательных резисторов [19]

Стягивающие и подтягивающие резисторы подключаются в схему параллельно по отношению к пинам и используются для того, чтобы, соответственно, «стягивать» значение напряжения на пине к низкому (обеспечивать стабильный сигнал логического 0) или «подтягивать» значение напряжения на пине к высокому (стабильная логическая 1). Делается это потому, что цифровые входы, не подключённые ни к какой нагрузке, являются «плавающими» (см. рис. 44, левую часть – представьте её без включённых туда резисторов; I/O – пины платы): они подвержены различного рода помехам и искажениям, появляющимся из-за электромагнитных возмущений, которые влияют на значения, читаемые с пинов платы приложениями, и могут способствовать непредсказуемым изменениям этих значений. Стягивающие и подтягивающие резисторы заставляют пины показывать правильные значения 0 или 1, даже если к ним ничего не подключено [19]. В правой части рис. 44 изображён случай с переключателем: если в схеме не будет подтягивающего резистора, значение при открытом переключателе на входном пине IN платы, читающем значения, будет плавающим; если в схеме есть такой резистор, изображённый на рисунке, значение на пине IN будет точно соответствовать логической единице. Для более подробного ознакомления с резисторами рекомендуются источники [11] или [19].

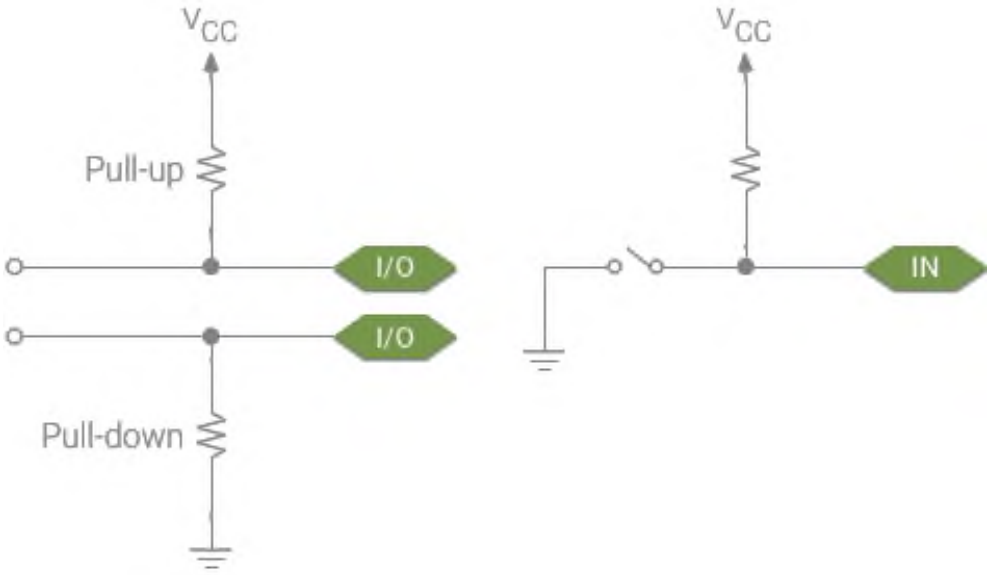


Рис. 44 ❖ Пример подключения стягивающего (pull-down) и подтягивающего (pull-up) резисторов [19]

Ещё один момент, о котором следует упомянуть, – это тот факт, что при подключении различных модулей и сенсоров надо также обращать внимание не только на схему, но и на надписи рядом с выходами (пинами) этих сенсоров. Сенсоры в наборах могут отличаться, хоть эта вероятность и мала, поэтому всегда проверяйте наличие информации/меток рядом с пинами устройств, используемых в практических заданиях, – это первично, а схема вторична. Простой пример – задание 24 из этой части книги, где к конкретным пинам платы подключаются конкретные выходы модуля часов реального времени, обозначения которых можно увидеть на самом модуле – см. рис. 24. Некоторые обозначения: – (минус), GND, G – земля; + (плюс), VCC, VIN, +5V, 3.3V – питание; CLK, SCK – clock (время, частота), DAT, SDA – date/data (дата, данные), RST – reset (сброс настроек), R,G,B – цвета на трёхцветном светодиоде; A0 (аналоговый), D0 (цифровой), SIG, S, VRx,VRy, SW – пин для передачи сигнала (данных).

Теперь, получив базовые знания об основных элементах, использующихся в практических экспериментах, и о технике безопасности, можно приступать к выполнению заданий.

Практическое занятие 1. Hello, world!

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm).

Первое, что нужно сделать, – это выбрать, где вы будете работать: в среде Arduino IDE или с помощью онлайн-системы Arduino Create (Arduino Web Editor) через веб-браузер. В первом случае необходимо скачать Arduino IDE ([2] -> Windows Installer) и установить её, включая установку драйвером для COM- и USB-портов. Во втором случае необходимо зарегистрироваться на сайте ([3] -> sign up).

Далее требуется пройти урок-инструкцию по следующему адресу: [4]. Это нужно для того, чтобы установить драйверы для платы, если они правильно не установились или нет прав администратора на компьютере, а также для того, чтобы правильно настроить среду и выбрать плату Arduino Uno и COM-порт.

После этого можно приступать к занятию. Подсоедините плату Arduino Uno к USB-порту компьютера (если вы ещё этого не сделали). Если драйверы установлены правильно, плата должна определиться, её название появится в панели уведомлений операционной системы. На плате есть встроенный мини-светодиод (miniLED), подключённый к 13-му цифровому порту. В этом занятии мы напишем код, который будет ожидать ввода через консоль буквы R, при её вводе заставлять miniLED 13 загораться на полсекунды, гаснуть на полсекунды и писать в консоль фразу Hello, World! Так как miniLED является встроенным, никаких дополнительных схем создавать не надо. Для того чтобы открыть консоль, надо выбрать в пункте меню **Инструменты** Монитор порта, или нажать комбинацию **Ctrl**+**Shift**+**M** (Arduino IDE), или выбрать пункт меню **Монитор порта** слева (Arduino Web Editor). Когда всё готово, можно скопировать код ниже в среду и загрузить программу на плату с помощью кнопки ->.

В коде используется команда Serial.begin(9600), означающая, что скорость/частота обмена данными платы с компьютером по UBS-соединению составляет 9600 bps (bits per second, бит в секунду). В консоли можно увидеть, что есть и другие частоты, но для выполнения задания в консоли должна быть выставлена такая же частота (справа внизу).

Схема представлена на рис. 45.

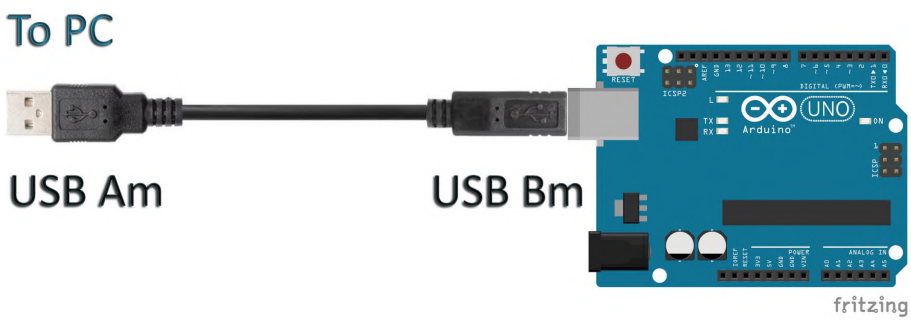


Рис. 45 ❖ Схема подключения для практического занятия 1

Код программы

```
int val ;// define a variable val
int ledpin = 13 ;// define the digital interface 13
void setup ()
{
    Serial.begin (9600) ;// set the baud rate to 9600, where the software settings keep
consistent.
    pinMode (ledpin, OUTPUT) ;// set the digital output interface 13 is, Arduino, we use
the I / O port should be carried out like this definition.
}
void loop ()
{
    val = Serial.read () ;// read the PC sends a command to the Arduino or characters, and
the
    instruction or character assigned val
    if (val == 'R') // determine the received command or character is «R».
    {// If you receive a «R» character
        digitalWrite (ledpin, HIGH) ;// lit Digital 13 LED.
        delay (500);
        digitalWrite (ledpin, LOW) ;// Off Digital 13 LED
        delay (500);
        Serial.println ("Hello World!") ;// Displays «Hello World!» String
    }
}
```

Практическое занятие 2. Эксперимент с мигающим светодиодом

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm).

В этом занятии мы всё ещё работаем со встроенным мини-светодиодом на плате (miniLED), подключённым к 13-му цифровому порту. Напишем код, который будет заставлять miniLED 13 загораться на секунду и гаснуть на секунду. Так как miniLED является встроенным, никаких дополнительных схем создавать не надо.

Схема представлена на рис. 46.

Код программы

```
int ledpin = 13 ;// define the digital interface 13
void setup ()
{
        pinMode (ledpin, OUTPUT) ;// set the digital output interface 13 is, Arduino, we
use the I / O port should be carried out like this definition.
}
void loop ()
{
        digitalWrite (ledpin, HIGH) ;// lit Digital 13 LED.
```

```
        delay (1000);
        digitalWrite (ledpin, LOW) ;// Off Digital 13 LED
        delay (1000);
}
```

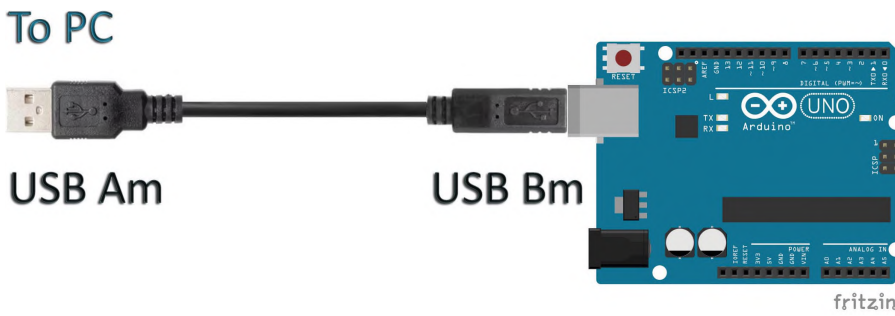


Рис. 46 ❖ Схема подключения для практического занятия 2

Практическое занятие 3. Эксперимент с контролируемой потенциометром яркостью свечения светодиода через порт PWM

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- потенциометр;
- светодиод;
- резистор на 220 Ом;
- макетная плата;
- соединительные провода.

В этом занятии мы познакомимся с потенциометром и портом PWM. PWM (Pulse Width Modulation) – широтно-импульсная модуляция, или процесс управления мощностью, подводимой к нагрузке, путём изменения отношения периода импульса к длительности импульса при неизменной частоте. На плате Arduino Uno можно подавать значения от 0 до 255 на PWM-пин, что заставит плату выдавать PWM-сигнал в определённые моменты времени, соответствующие поданному входному значению. Другими словами, в терминах напряжения, подавая разные значения от 0 до 255 на PWM-пин, мы заставляем плату менять напряжение от 0 до 5 В. В коде используется функция analogWrite(pin, value), с помощью которой можно менять напряжение, подаваемое на PWM-пин, задавая value от 0 до 255. Эта функция используется для регулировки скорости вращения мотора, яркости светодиода и т. д.

Arduino Uno располагает несколькими PWM-пинами, обозначенными символом ~ рядом с номером пина: 3, 5, 6, 9, 10, 11 (см. рис. 1). С помощью потенци-

ометра, подключённого к одному из аналоговых портов платы (A0–A6), и светодиода, катод (короткий пин) которого подключён через резистор на 220 ОМ к земле (всегда! к минусу) и анод (длинный пин) которого подключён к одному из цифровых пинов (0–13) платы (всегда! к плюсу), мы будем регулировать значение переменной val (крутя ручку потенциометра) и, соответственно, напряжение, которое выдаёт плата из этого цифрового пина.

Схема представлена на рис. 47–48.

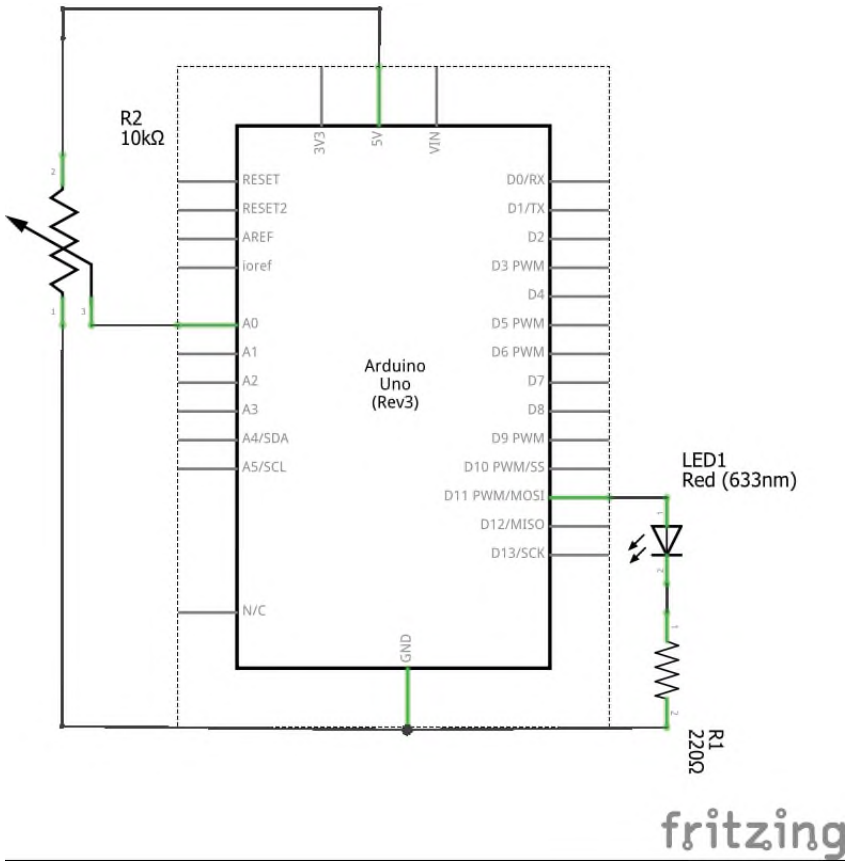


Рис. 47 ❖ Принципиальная схема подключения для практического занятия 3

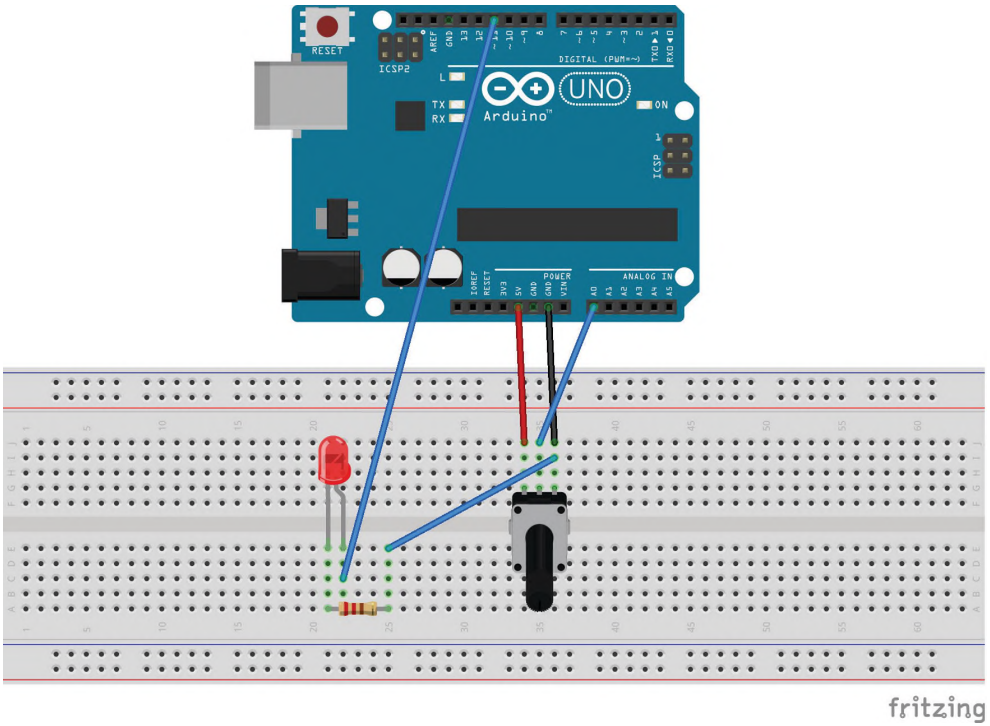


Рис. 48 ❖ Схема подключения с макетной платой для практического занятия 3

Код программы

```
    int potpin = 0 ;// define analog interface 0
    int ledpin = 11 ;// define the digital interface 11 (PWM output)
    int val = 0 ;// temporary values of the variables from the sensor
    void setup ()
    {
        pinMode (ledpin, OUTPUT) ;// define the digital interface 11 as output
        Serial.begin (9600) ;// set the baud rate to 9600
// NOTE: analog interface is automatically set to the input
    }
    void loop ()
    {
        val = analogRead (potpin) ;// read sensor analog values and assigned to val
        Serial.println (val) ;// display val variable
        analogWrite (ledpin, val / 4) ;// turn on the LED and set the brightness (PWM
output max 255)
        delay (10) ;// delay of 0.01 seconds
    }
```

Практическое занятие 4. Эксперимент с внешним мигающим светодиодом

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- светодиод;
- резистор на 220 Ом;
- макетная плата;
- соединительные провода.

В этом занятии мы подсоединим внешний светодиод к цифровому порту 10 платы. Напишем код, который будет заставлять светодиод загораться на секунду и гаснуть на секунду. В этот раз, по сравнению с занятием 2, светодиод у нас внешний, поэтому надо собрать схему, показанную ниже. В схеме с внешним светодиодом всегда должен быть резистор с малым сопротивлением (220 Ом), иначе светодиод может перегореть.

Схема представлена на рис. 49–50.

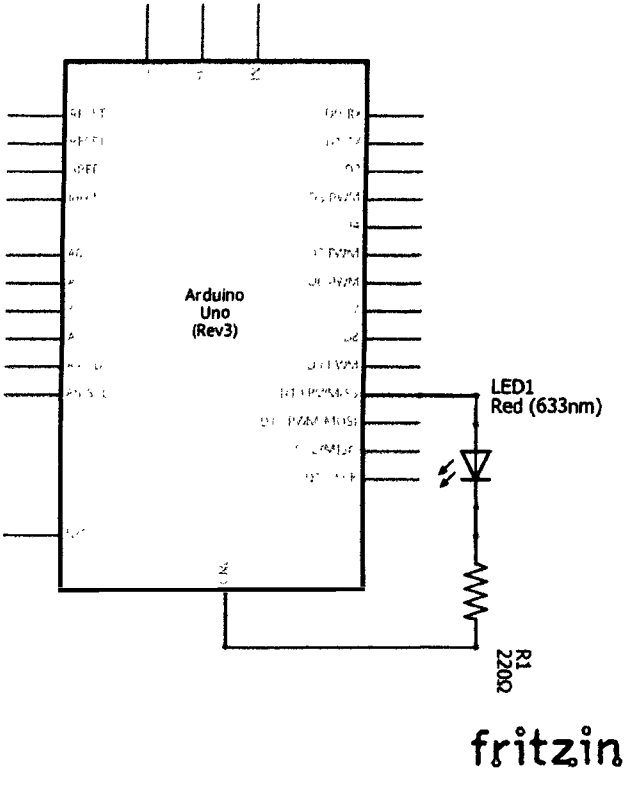


Рис. 49 ❖ Принципиальная схема подключения для практического занятия 4

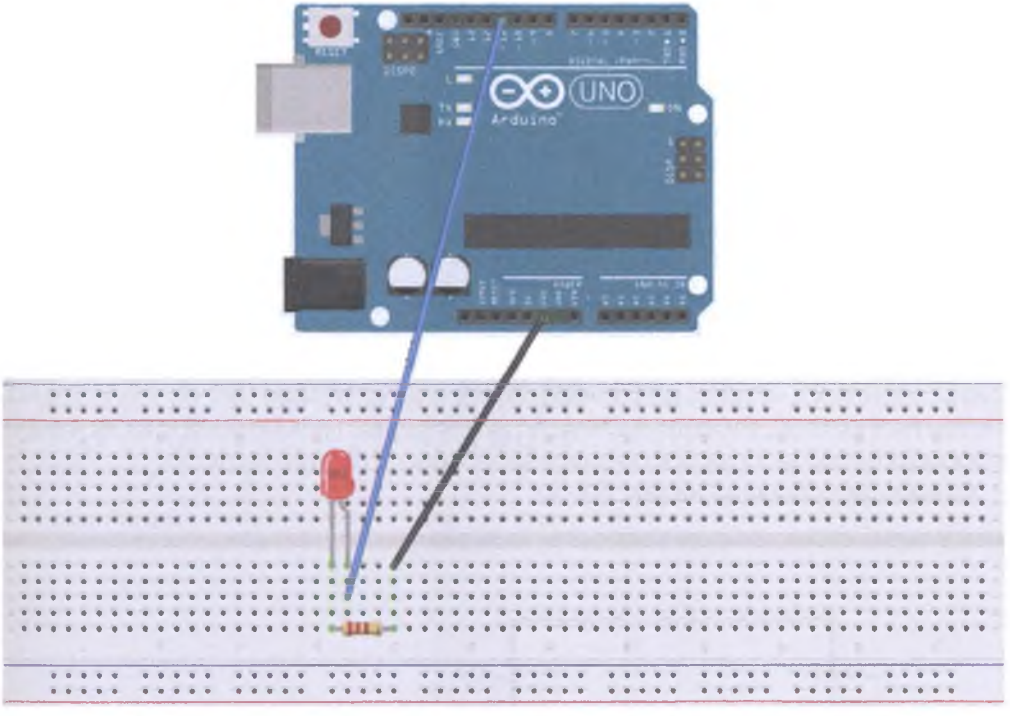


Рис. 50 ❖ Схема подключения с макетной платой для практического занятия 4

Код программы

```
int ledPin = 10; // define the interface number 10
void setup ()
{
  pinMode (ledPin, OUTPUT) ;// define a small lamp interface output interface
}
void loop ()
{
  digitalWrite (ledPin, HIGH); // lit a small lamp
  delay (1000); // Delay 1 second
  digitalWrite (ledPin, LOW); // extinguish small lights
  delay (1000); // Delay 1 second
}
```

Практическое занятие 5. Эксперимент с рекламной расцветкой

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- светодиоды – 6 шт., по 2 каждого цвета;
- резистор на 220 Ом – 6 шт.;
- макетная плата;
- соединительные провода.

В этом занятии мы напишем код, который будет заставлять 6 светодиодов, подключённых к чётным цифровым пинам платы Arduino Uno (2, 4, 6, 8, 10, 12), загораться волной, имитируя ёлочную гирлянду или рекламный щит – сначала все светодиоды загораются по очереди, потом гаснут по очереди.

Схема представлена на рис. 51–52.

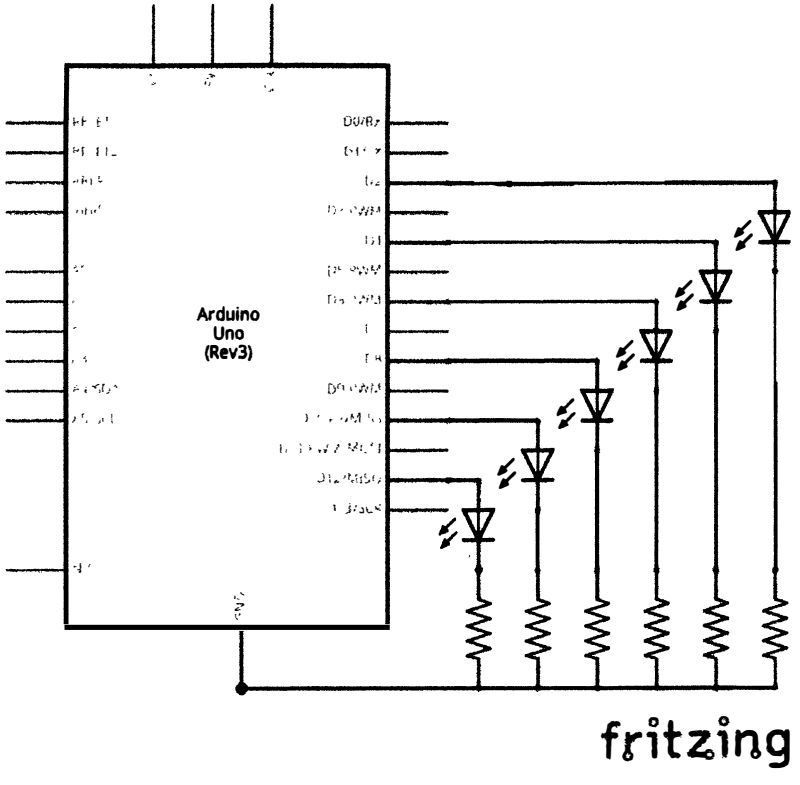


Рис. 51 ❖ Принципиальная схема подключения для практического занятия 5

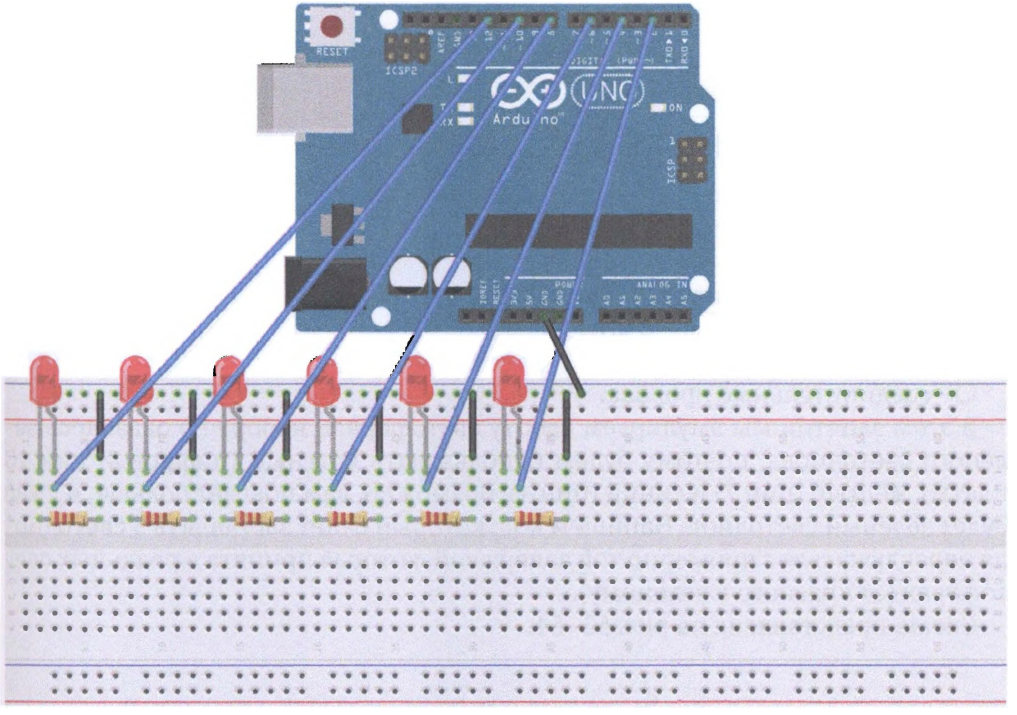


Рис. 52 ❖ Схема подключения с макетной платой для практического занятия 5

Код программы

```
int BASE = 2; // the first one LED connected to the I / O pins
int NUM = 6; // LED's total
void setup ()
{
for (int i = BASE; i <=BASE*NUM; i+=2)
{
    pinMode (i, OUTPUT); // set the digital I / O pin as an output
}
}
void loop ()
{
  for (int i = BASE; i <=BASE*NUM; i+=2)
  {
    digitalWrite (i, LOW); // set the digital I / O pin output is «low», that gradually
turn off the
    lights
    delay (200); // delay
  }
  for (int i = BASE; i <=BASE*NUM; i+=2)
  {
    digitalWrite (i, HIGH); // set the digital I / O pin output is «low», that gradually
lights
```

```
    delay (200); // delay
  }
}
```

Практическое занятие 6. Светофорный эксперимент

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- светодиоды – 3 шт., разных цветов;
- резистор на 220 Ом – 3 шт.;
- макетная плата;
- соединительные провода.

В этом занятии мы эмулируем работу светофора. В наборе не оказалось зелёного светодиода, поэтому будем использовать красный (цифровой пин 10 платы), жёлтый (пин 7) и синий (пин 4). Возможно, вам повезёт больше, и у вас будет зелёный светодиод. Напишем код, который будет заставлять 3 светодиода переключаться с примерно той задержкой, которая принята в реальных светофорах (от красного к зелёному).

Схема представлена на рис. 53–54.

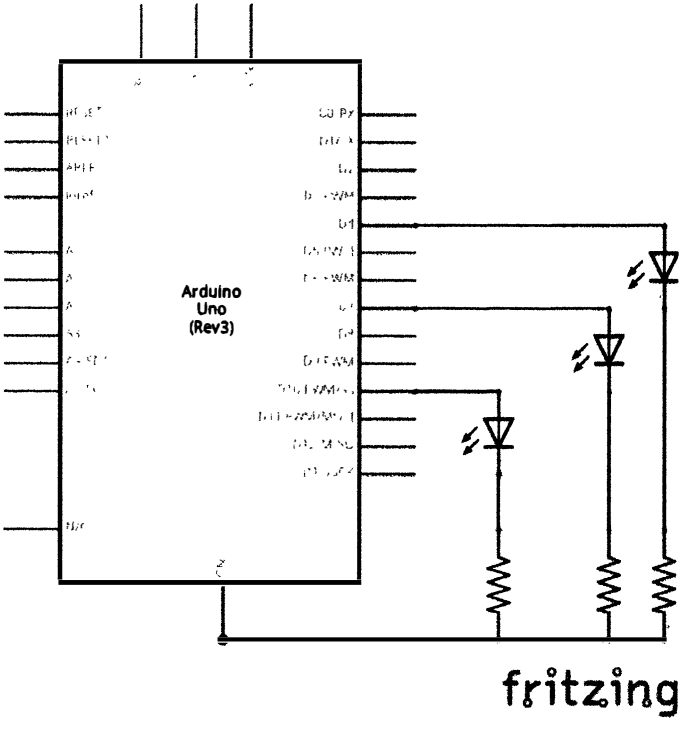


Рис. 53 ❖ Принципиальная схема подключения для практического занятия 6

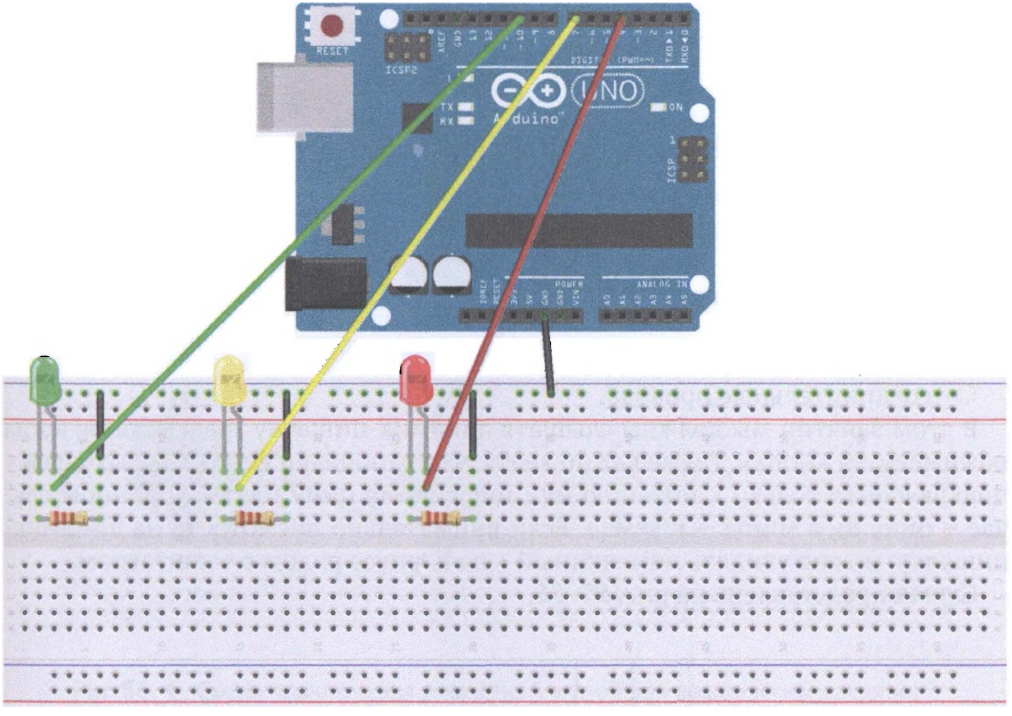


Рис. 54 ❖ Схема подключения с макетной платой для практического занятия 6

Код программы

```
int redled = 10; // define the interface number 10
        int yellowled = 7; // define the number 7 Interface
        int greenled = 4; // define the number 4 Interface
        void setup ()
        {
            pinMode (redled, OUTPUT) ;// define a small red light interface output
interface
            pinMode (yellowled, OUTPUT); // define the yellow light interface output
interface
            pinMode (greenled, OUTPUT); // define the small green light interface output
interface
}
        void loop ()
        {
            digitalWrite (redled, HIGH) ;// lit red lights
            delay (1000) ;// delay of 1 second
            digitalWrite (redled, LOW); // off red light
            digitalWrite (yellowled, HIGH) ;// light yellow light
            delay (200) ;// delay of 0.2 seconds
            digitalWrite (yellowled, LOW) ;// off yellow light
            digitalWrite (greenled, HIGH) ;// flashes the green LED
```

```
    delay (1000) ;// delay of 1 second
    digitalWrite (greenled, LOW) ;// green LED off
}
```

Практическое занятие 7. Эксперимент с пищалкой

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- динамик-пищалка;
- кнопка;
- макетная плата;
- соединительные провода.

В этом занятии мы должны слышать динамик-пищалку только тогда, когда нажимаем на кнопку. Также важно знать, что у пищалки есть + и – выходы и + подключается только к питанию или цифровому пину платы (в данном случае – пину 8), а минус – к земле. Если внимательно посмотреть на динамик-пищалку, можно увидеть помеченный + на корпусе рядом с + контактом.

Схема представлена на рис. 55–56.

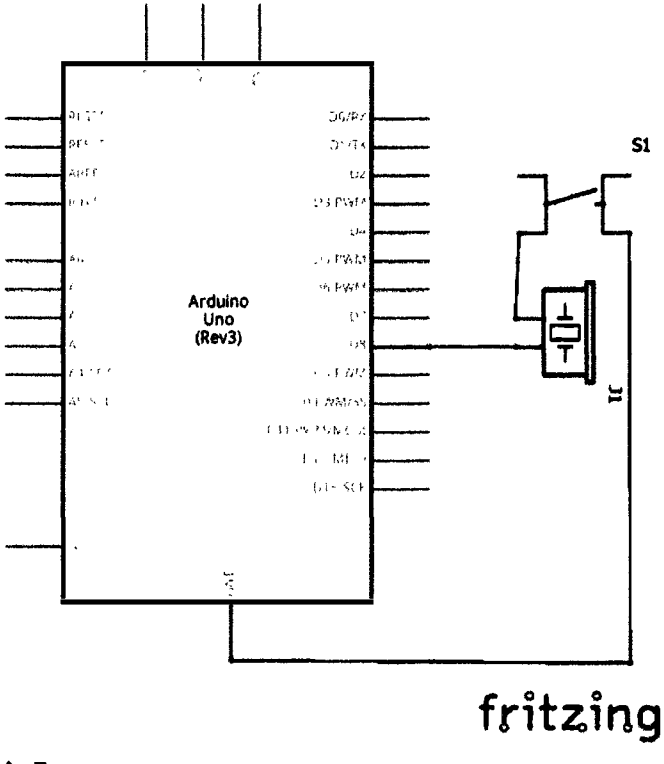


Рис. 55 ❖ Принципиальная схема подключения для практического занятия 7

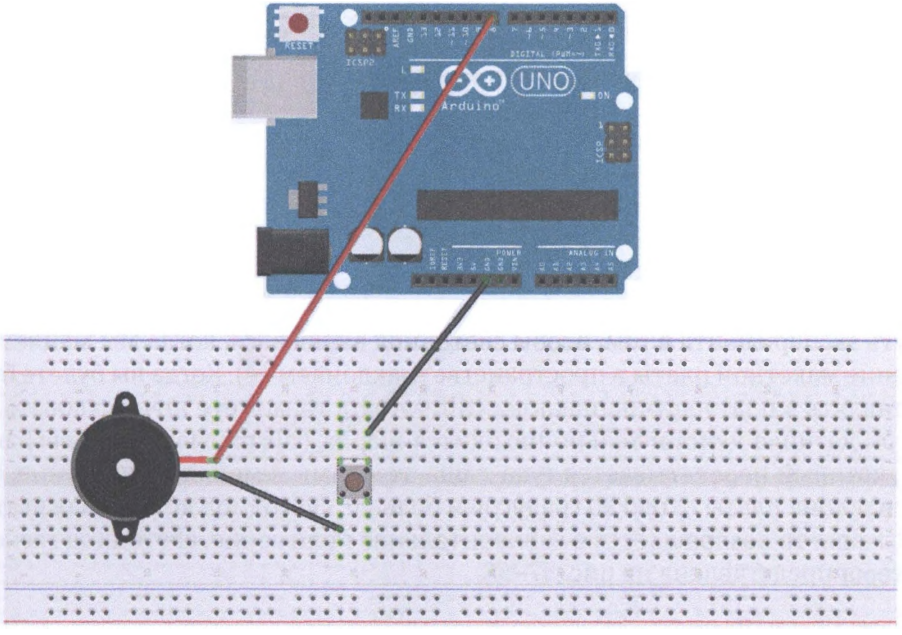


Рис. 56 ❖ Схема подключения с макетной платой для практического занятия 7

Код программы

```
int buzzer = 8 ;// setting controls the digital IO foot buzzer
void setup ()
{
    pinMode (buzzer, OUTPUT) ;// set the digital IO pin mode, OUTPUT out of Wen
}
void loop ()
{
    unsigned char i, j ;// define variables
    while (1)
    {
    for (i = 0; i <80; i++) // Wen a frequency sound
    {
        digitalWrite (buzzer, HIGH) ;// send voice
        delay (1) ;// Delay 1ms
        digitalWrite (buzzer, LOW) ;// do not send voice
        delay (1) ;// delay ms
    }
    for (i = 0; i <100; i++) // Wen Qie out another frequency sound
    {
        digitalWrite (buzzer, HIGH) ;// send voice
        delay (2) ;// delay 2ms
        digitalWrite (buzzer, LOW) ;// do not send voice
        delay (2) ;// delay 2ms
    }
    }
}
```

Практическое занятие 8. Эксперимент с датчиком наклона

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- датчик наклона (tilt switch);
- светодиод;
- резистор на 220 Ом;
- макетная плата;
- соединительные провода.

Суть эксперимента в том, чтобы светодиод включался, когда мы меняем положение макетной платы в пространстве (наклоняем её). Когда вы будете брать и вставлять в плату датчик наклона (tilt switch), то можете почувствовать, что внутри датчика действительно находится шарик (tilt mechanism – mechanical ball), который перекатывается туда-сюда, тем самым меняя напряжение. Если угол наклона платы около 90 градусов и больше, светодиод должен зажигаться, если же плата возвращается в горизонтальное положение, светодиод гаснет.

Схема представлена на рис. 57–58.

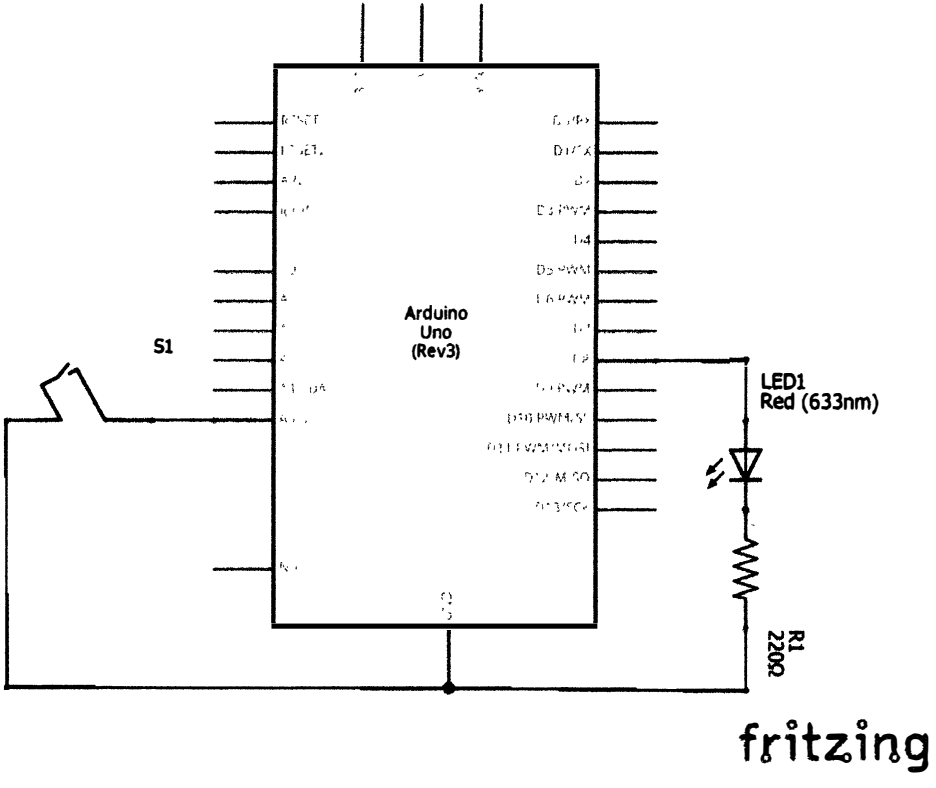


Рис. 57 ❖ Принципиальная схема подключения для практического занятия 8

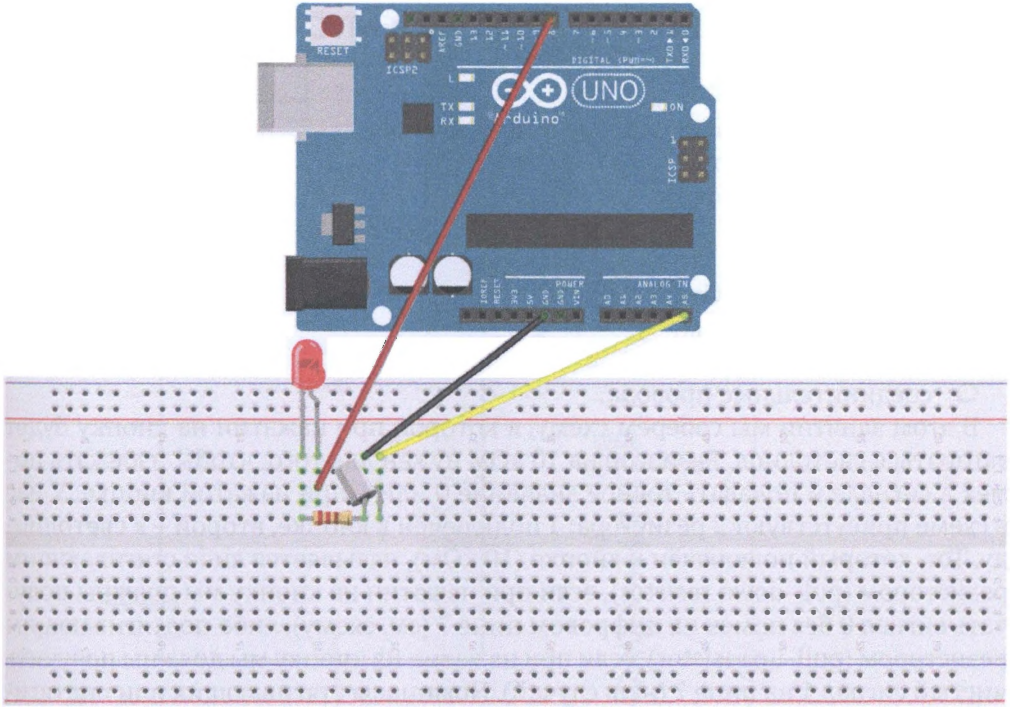


Рис. 58 ❖ Схема подключения с макетной платой для практического занятия 8

Код программы

```
void setup ()
{
    pinMode (8, OUTPUT) ;// set the digital 8 pin to out mode
}
void loop ()
{
    int i ;// define the amount of i
    while (1)
    {
    i = analogRead (5) ;// read the analog voltage value 5
    if (i> 200) // if greater than 512 (2.5V)
    {
        digitalWrite (8, HIGH) ;// led light is lit
    }
    else // Otherwise
    {
        digitalWrite (8, LOW) ;// off led light
 }
 }
}
```

Практическое занятие 9. Эксперимент с чистым входным сигналом

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- кнопка;
- светодиод;
- резистор на 220 Ом;
- резистор на 10 кОм;
- макетная плата;
- соединительные провода.

В этом занятии мы соберём схему, в которой при нажатии на кнопку будет загораться светодиод. Резистор на 10 кОм нужен для того, чтобы избежать помех в сигнале и подавать точное значение 0 или 1 при нажатой кнопке. У нас в схеме присутствуют 2 резистора: 1 подключён к кнопке, второй – к светодиоду. Тот, который подключён к кнопке (10 кОм), называется либо стягивающим резистором (pull-down resistor), если при нажатии на кнопку мы должны получать сигнал 0 без помех на цифровом пине 7 (см. схему), либо подтягивающим резистором (pull-up resistor), если при нажатии на кнопку мы должны получить чистый сигнал 1 на пине 7 (наш случай). Номиналы стягивающих или подтягивающих резисторов обычно варьируются от 1 кОм до 10 кОм. Тот же резистор, который подключён к катоду светодиода, называется серийным, или последовательным (series resistor), и нужен для того, чтобы брать на себя часть тока во избежание перегорания светодиода. Значения таких резисторов обычно варьируются от 100 Ом до 300 Ом. Почему нам важен чистый сигнал в этом занятии? Потому что мы читаем его с пина 7 с помощью функции digitalRead(7).

Схема представлена на рис. 59–60.

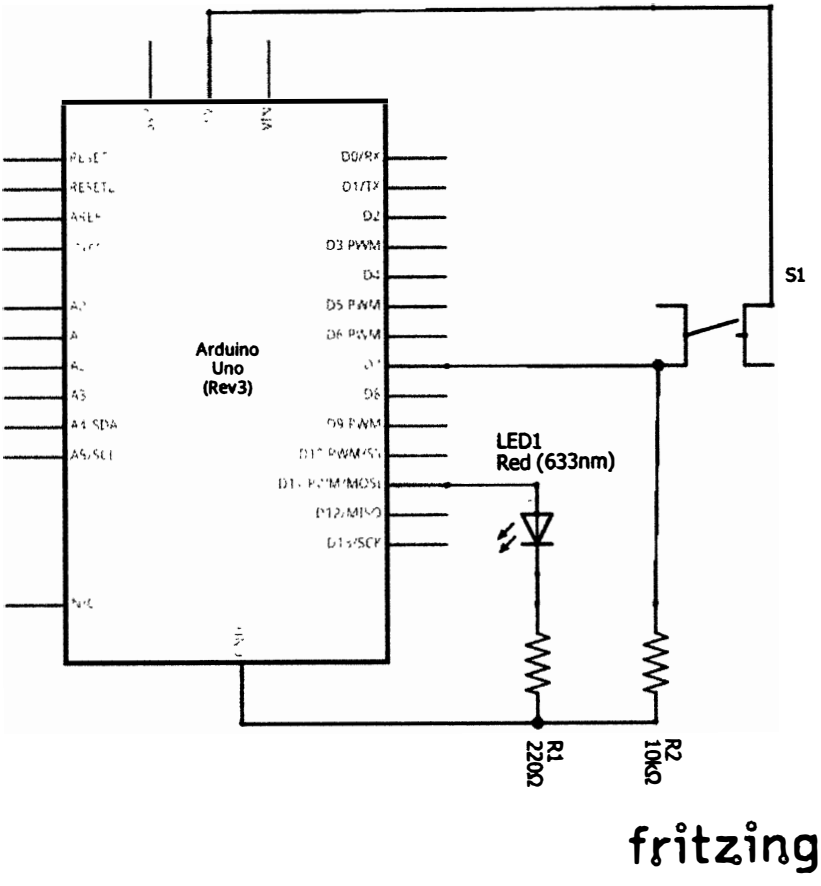


Рис. 59 ❖ Принципиальная схема подключения для практического занятия 9

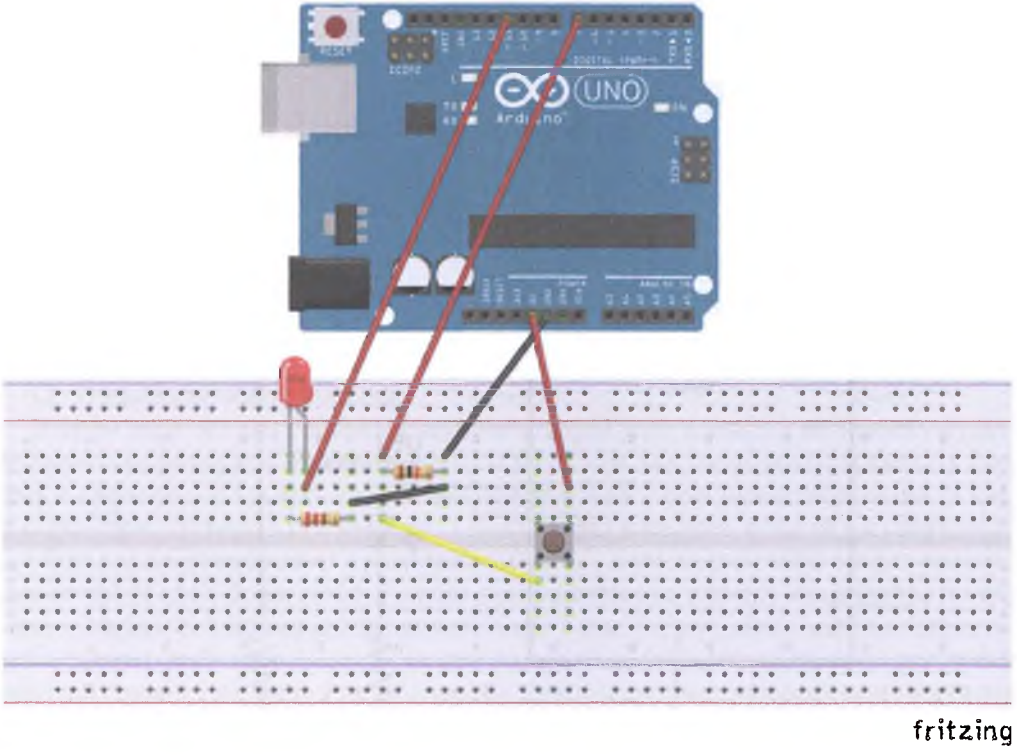


Рис. 60 ❖ Схема подключения с макетной платой для практического занятия 9

Код программы

```
int ledpin = 11 ;// define the interface number 11
int inpin = 7 ;// define the number 7 Interface
int val ;// define a variable val
void setup ()
{
    pinMode (ledpin, OUTPUT) ;// define a small lamp interface output interface
    pinMode (inpin, INPUT) ;// define the key interface for the input interface
}
void loop ()
{
    val = digitalRead (inpin) ;// read digital 7-level value assigned to val
    if (val == LOW) // test button is pressed, the button lights up when pressed small
    {digitalWrite (ledpin, LOW);}
    else
    {digitalWrite (ledpin, HIGH);}
}
```

Практическое занятие 10. Расширенный эксперимент с чистым сигналом

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- кнопка – 3 шт.;
- светодиоды разных цветов – 3 шт.;
- резистор на 220 Ом – 3 шт.;
- резистор на 10 кОм – 3 шт.;
- макетная плата;
- соединительные провода.

В этом занятии мы соберём схему, в которой при нажатии на каждую из кнопок будет загораться соответствующий светодиод. В сложных схемах, подобных этой, лучше всего выводить питание и землю на отдельные рельсы (+ и –, или красный и синий горизонтальный рельс) макетной платы.

Схема представлена на рис. 61–62.

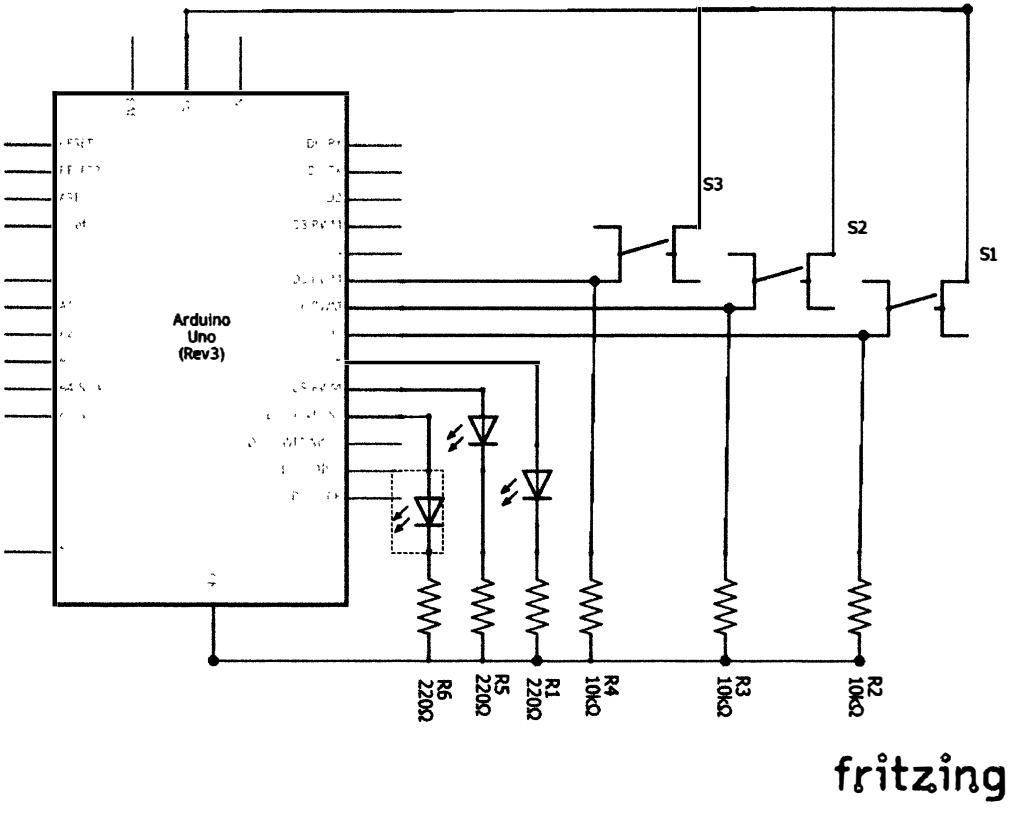


Рис. 61 ❖ Принципиальная схема подключения для практического занятия 10

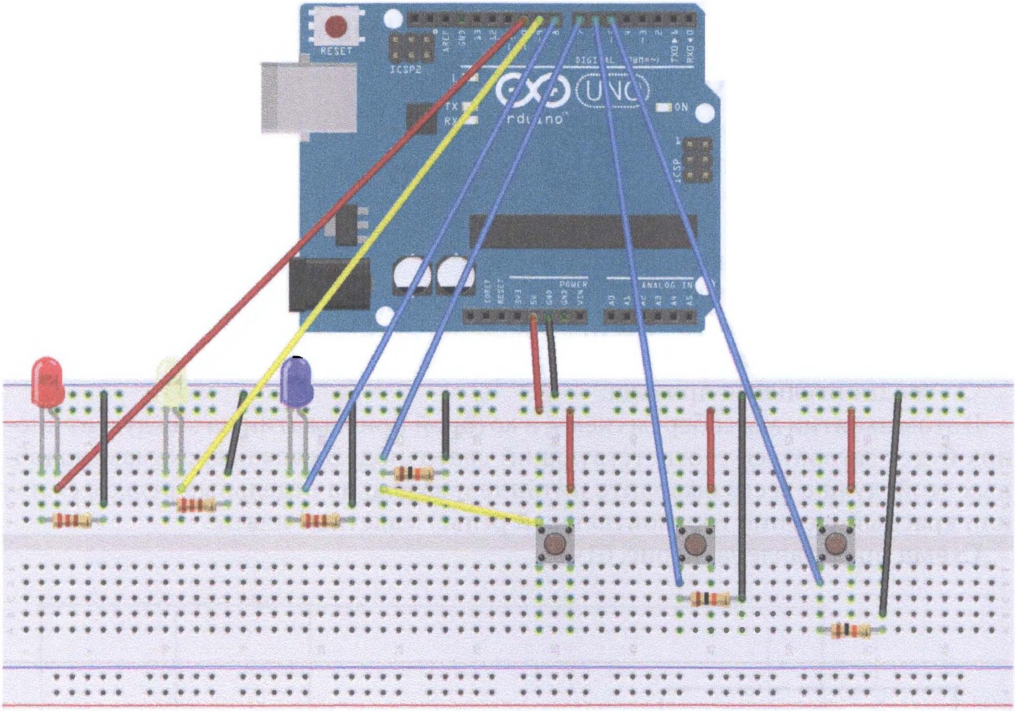


Рис. 62 ❖ Схема подключения с макетной платой для практического занятия 10

Код программы

```
int redled = 10;
int yellowled = 9;
int greenled = 8;
int redpin = 7;
int yellowpin = 6;
int greenpin = 5;
int red;
int yellow;
int green;
void setup ()
{
    pinMode (redled, OUTPUT);
    pinMode (yellowled, OUTPUT);
    pinMode (greenled, OUTPUT);
    pinMode (redpin, INPUT);
```

```
    pinMode (yellowpin, INPUT);
    pinMode (greenpin, INPUT);
}
void loop ()
{
    red = digitalRead (redpin);
    if (red == LOW)
    {digitalWrite (redled, LOW);}
    else
    {digitalWrite (redled, HIGH);}
    yellow = digitalRead (yellowpin);
    if (yellow == LOW)
    {digitalWrite (yellowled, LOW);}
    else
    {digitalWrite (yellowled, HIGH);}
    green = digitalRead (greenpin);
    if (green == LOW)
    {digitalWrite (greenled, LOW);}
    else
    {digitalWrite (greenled, HIGH);}
}
```

Практическое занятие 11. Эксперимент по чтению аналогового значения

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- потенциометр;
- макетная плата;
- соединительные провода.

У Arduino Uno есть аналоговые входы A0–A5. Воспользуемся входом A0 и функцией analogRead(), чтобы считывать данные о текущем сопротивлении потенциометра (значения от 0 до 1023, т. к. Arduino Uno располагает 10-битными аналоговыми или цифровыми портами, 2^10 = 1024 значения) и выводить их на экран. В коде используется команда Serial.begin(9600), означающая, что скорость/частота обмена данными платы с компьютером по UBS-соединению составляет 9600 bps (bits per second, битов в секунду). В консоли должна быть выставлена такая же частота (справа внизу). Не забудьте открыть консоль, чтобы видеть сообщения (**Ctrl**+**Shift**+**M**)!

Схема представлена на рис. 63–64.

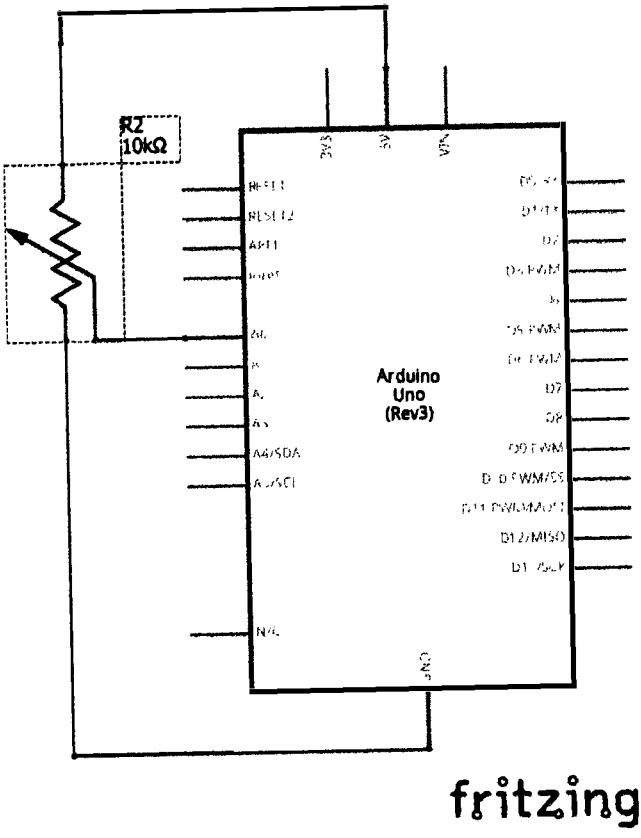


Рис. 63 ❖ Принципиальная схема подключения для практического занятия 11

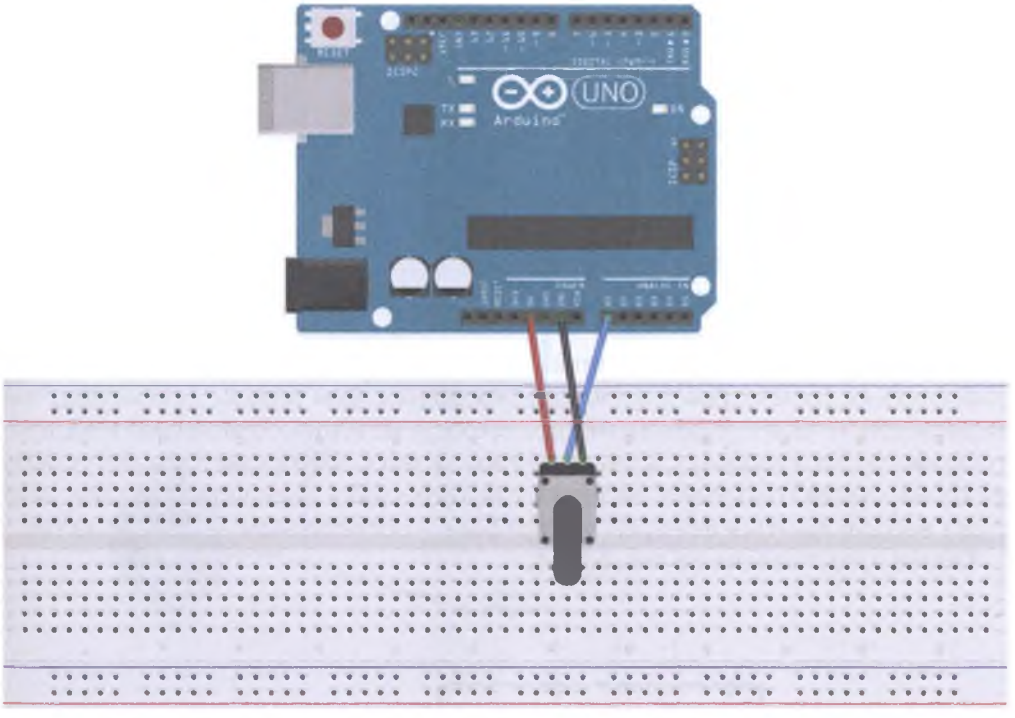


Рис. 64 ❖ Схема подключения с макетной платой для практического занятия 11

Код программы

```
int potpin = 0 ;// define analog interface 0
int ledpin = 13 ;// define the digital interface 13
int val = 0 ;// will define the variable val, and the initial value 0
void setup ()
{
    pinMode (ledpin, OUTPUT) ;// output interface defines the digital interface
    Serial.begin (9600) ;// set the baud rate to 9600
}
void loop ()
{
    digitalWrite (ledpin, HIGH) ;// digital interface 13 of the LED lights
    delay (50) ;// delay of 0.05 seconds
    digitalWrite (ledpin, LOW) ;// off LED digital interface 13
    delay (50) ;// delay of 0.05 seconds
    val = analogRead (potpin) ;// read the value of analog interface 0, and assign val
    Serial.println (val) ; // shows the value of val
}
```

Практическое занятие 12. Эксперимент по управлению звуком и светом

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- фоторезистор;
- динамик-пищалка;
- макетная плата;
- соединительные провода.

В данном занятии мы подсоединяем фоторезистор к пищалке, для того чтобы управлять её громкостью с помощью освещения. Чем больше освещения, тем меньше сопротивление фоторезистора и, соответственно, больше сила тока, протекающего через пищалку, а значит – громче писк. Фоторезисторы могут быть не очень чувствительными, поэтому, для того чтобы услышать пищалку, надо будет воспользоваться фонарём в смартфоне – посветить им на фоторезистор.

Схема представлена на рис. 65–66.

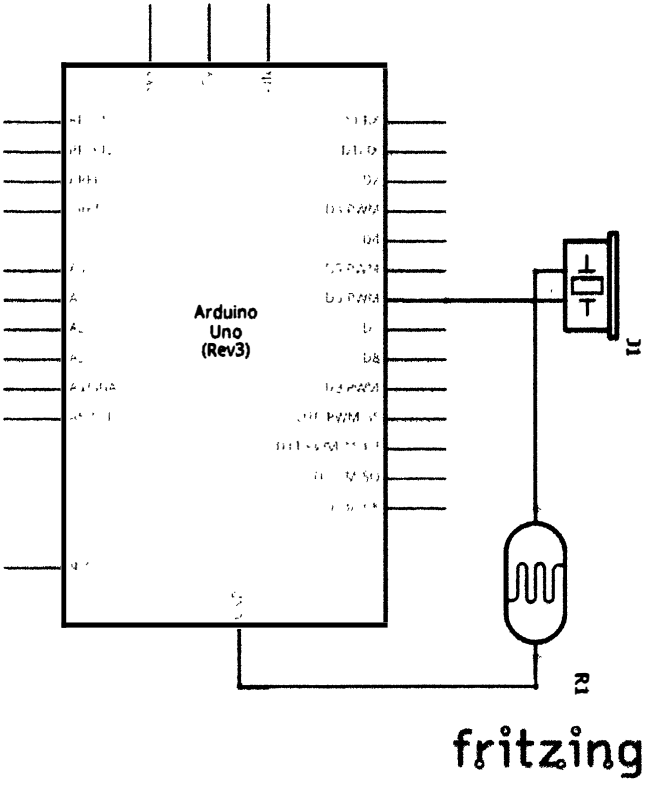


Рис. 65 ❖ Принципиальная схема подключения для практического занятия 12

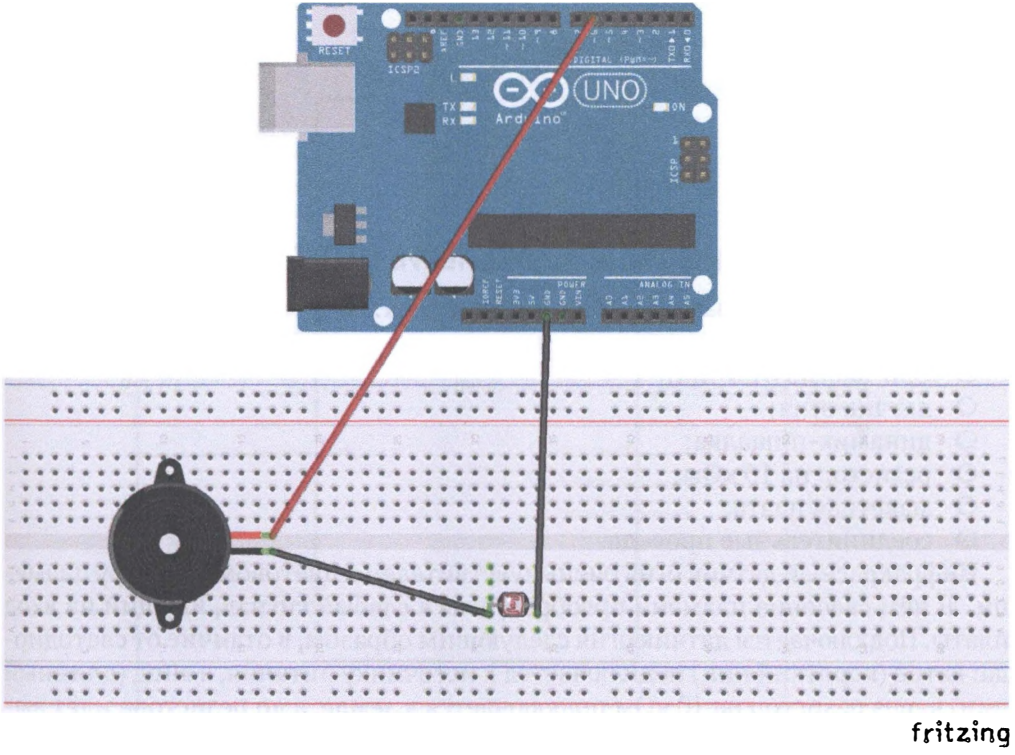


Рис. 66 ❖ Схема подключения с макетной платой для практического занятия 12

Код программы

```
void setup ()
{
    pinMode (6, OUTPUT);
}
void loop ()
{
    while (1)
    {
        char i, j;
        while (1)
        {
            for (i = 0; i <80; i++) // Wen Qie one frequency sound
            {
                digitalWrite (6, HIGH);
                delay (1);
                digitalWrite (6, LOW);
                delay (1);
            }
            for (i = 0; i <100; i++) // Wen Qie out another frequency sound
            {
                digitalWrite (6, HIGH);
```

```
                delay (2);
                digitalWrite (6, LOW);
                delay (2);
            }
        }
    }
}
```

Практическое занятие 13. Эксперимент с датчиком огня

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- датчик огня;
- динамик-пищалка;
- резистор на 10 кОм;
- макетная плата;
- соединительные провода.

Инфракрасный датчик огня очень чувствителен к цветовому спектру пламени. В нём свечение пламени преобразуется в слабый сигнал, идущий на вход платы. Подключается датчик огня следующим образом, в отличие от светодиода: катод (короткий пин) подключается к источнику питания, а анод (длинный пин) через резистор на 10 кОм подключается к земле, а до резистора идёт выход на аналоговый пин платы Arduino Uno. Датчик огня работает и без огня, выдавая на аналоговый пин платы около 0.3 В. Когда пламя находится слишком близко, датчик выдаёт уже 1.0 В и больше. Поэтому в коде было бы целесообразно использовать пороговое значение 0.6 В. Однако практика показала, что на самом деле датчик очень чувствительный, и даже в комнате с верхним освещением из трёх обыкновенных ламп накаливания выдаёт значение 984–985, что соответствует 4.8 В, хотя никакого огня в комнате нет. Поэтому в коде пороговое значение выставлено в 1000. В этом задании нам также понадобится консоль (**Ctrl+Shift+M**) и зажигалка/спички (зажигалка, конечно, получше спичек).

Схема представлена на рис. 67–68.

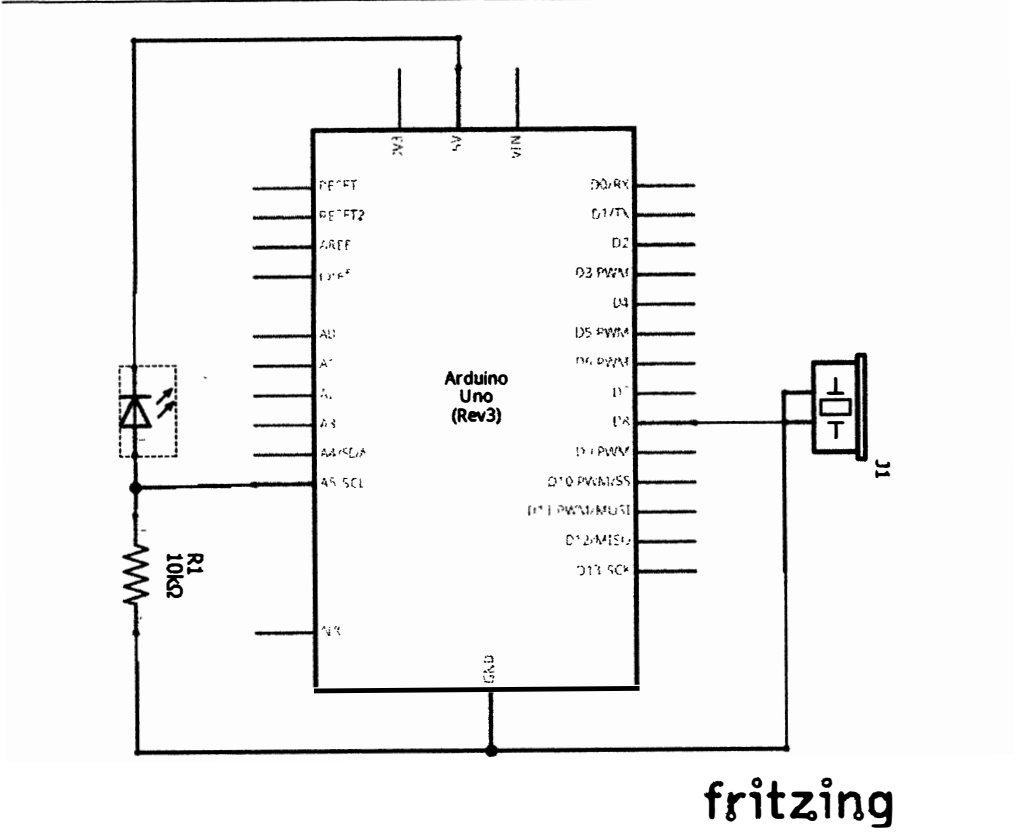


Рис. 67 ❖ Принципиальная схема подключения для практического занятия 13

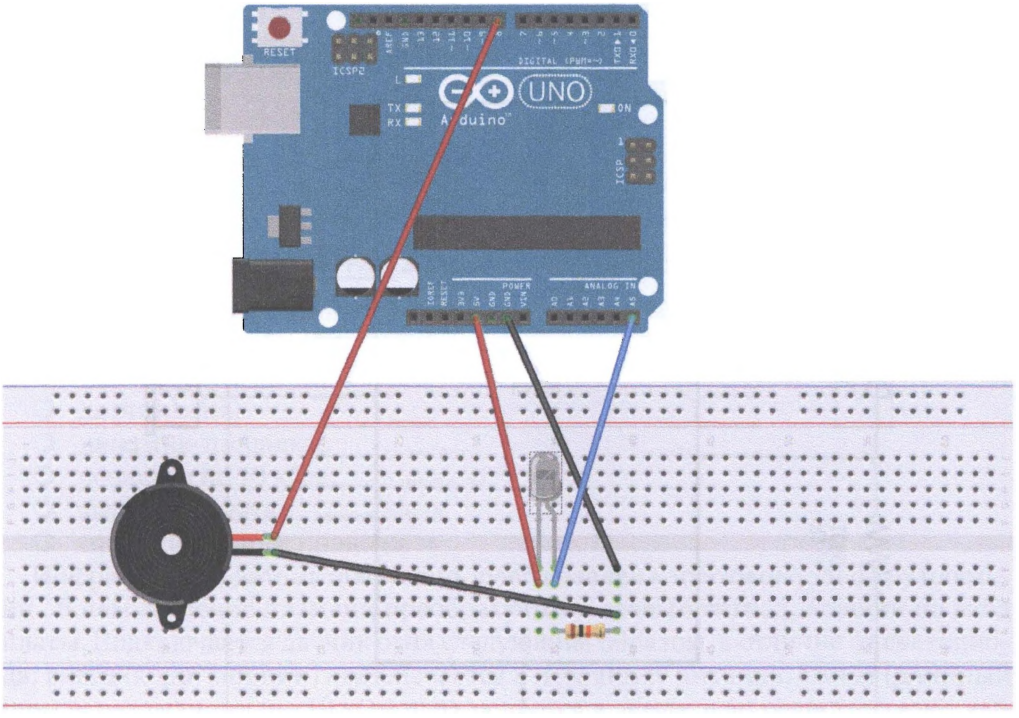


Рис. 68 ❖ Схема подключения с макетной платой для практического занятия 13

Код программы

```
int flame = A5;// define the flame interface analog 0 interface
int Beep = 8;// buzzer interface defines the interface number 7
int val = 0;// define numeric variables val
void setup ()
    {
        pinMode (Beep, OUTPUT) ;// define LED as output interface
        pinMode (flame, INPUT) ;// define the buzzer as the input interface
        Serial.begin (9600) ;// set the baud rate to 9600
    }
void loop () {
        val = analogRead (flame) ;// read the analog value flame sensor
        Serial.println (val) ;// output analog values, and print them out
        if (val >= 1000) // When the analog value is greater than 600 when the buzzer
sounds
            {
            digitalWrite (Beep, HIGH);
            } else {
            digitalWrite (Beep, LOW);
            }
    }
```

Практическое занятие 14. Эксперимент с вольтметром

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- резистор на 1 кОм;
- макетная плата;
- батарейка/аккумулятор на 1.5 В (**не больше**);
- соединительные провода.

В этом занятии будем строить схему с вольтметром с диапазоном 0–5 В. Нам также понадобится консоль (**Ctrl+Shift+M**), чтобы смотреть, какие значения мы считываем с аналогового порта 0 платы. При подключении схемы в консоли мы будем видеть, какой заряд у батарейки/аккумулятора.

Схема представлена на рис. 69–70.

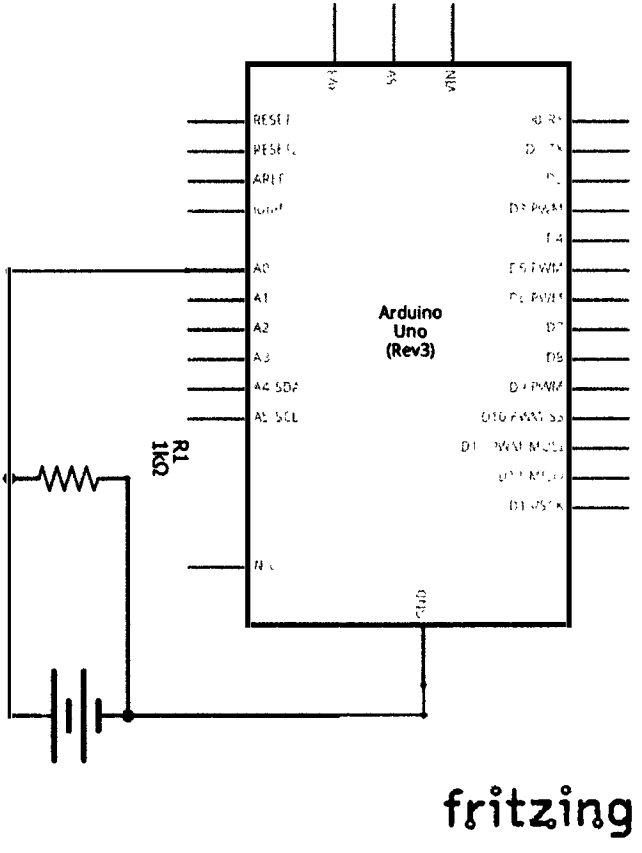


Рис. 69 ❖ Принципиальная схема подключения для практического занятия 14

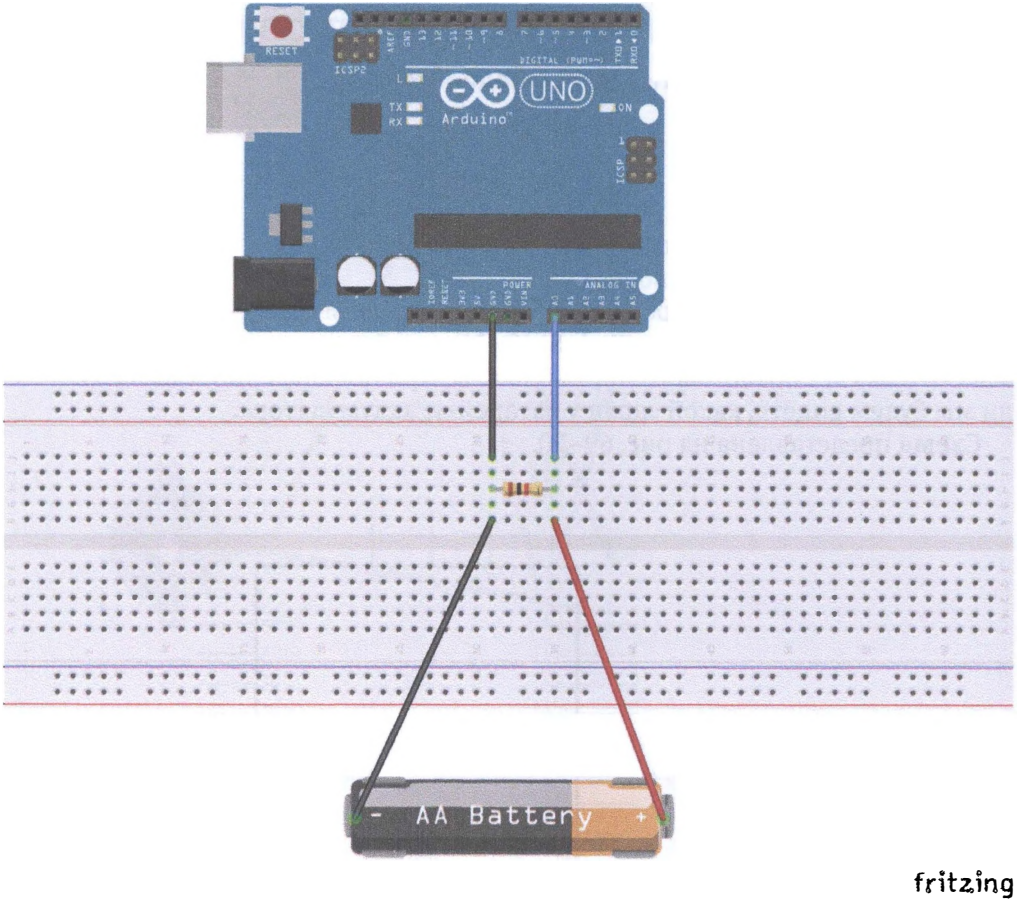


Рис. 70 ❖ Схема подключения с макетной платой для практического занятия 14

Код программы

```
float temp; // create a float variable temp as a storage space to store data preparation
void setup ()
{
Serial.begin (9600); // use 9600 baud rate serial communication
}
void loop ()
{
int V1 = analogRead (A0);
// Read voltage from A0 mouth integer type data into the newly created variable V1, analog port voltage measurement range of 0-5V returns a value of 0-1024
float vol = V1 * (5.0 / 1023.0);
// We will be converted into the actual value of V1 voltage value into a float variable vol
if (vol == temp)
// This part of the judgment is used to filter duplicate data, only the second voltage value when the output and the last mixed
```

```
{
temp = vol; // After the completion of the comparison, the ratio of this value into a
variable temp for comparison
}
else
{
Serial.print (vol); // serial output voltage value, and do not wrap
Serial.println ("V"); // serial output character V, and line breaks
temp = vol;
delay (1000); // Wait a second output is completed, the data used to control the refresh
rate.
}
}
```

Практическое занятие 15. Эксперимент с распознаванием голоса

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- модуль распознавания шумов;
- светодиод;
- резистор на 220 Ом;
- макетная плата;
- соединительные провода.

У модуля распознавания шумов есть 4 выхода. A0 – аналоговый выход, который мы подключаем к аналоговому входу A0 платы. G – земля, а + выход надо подключить к 5 В на плате. Далее собираем схему со светодиодом, как обычно, и открываем консоль (**Ctrl+Shift+M**), чтобы смотреть значение шума, создаваемого в аудитории. На схеме ниже представлен другой элемент с 4 выходами (не модуль распознавания шумов), так как в библиотеке Fritzing не оказалось подобного модуля или микрофона с 3 выходами; но главное – суть подключения.

Схема представлена на рис. 71–72.

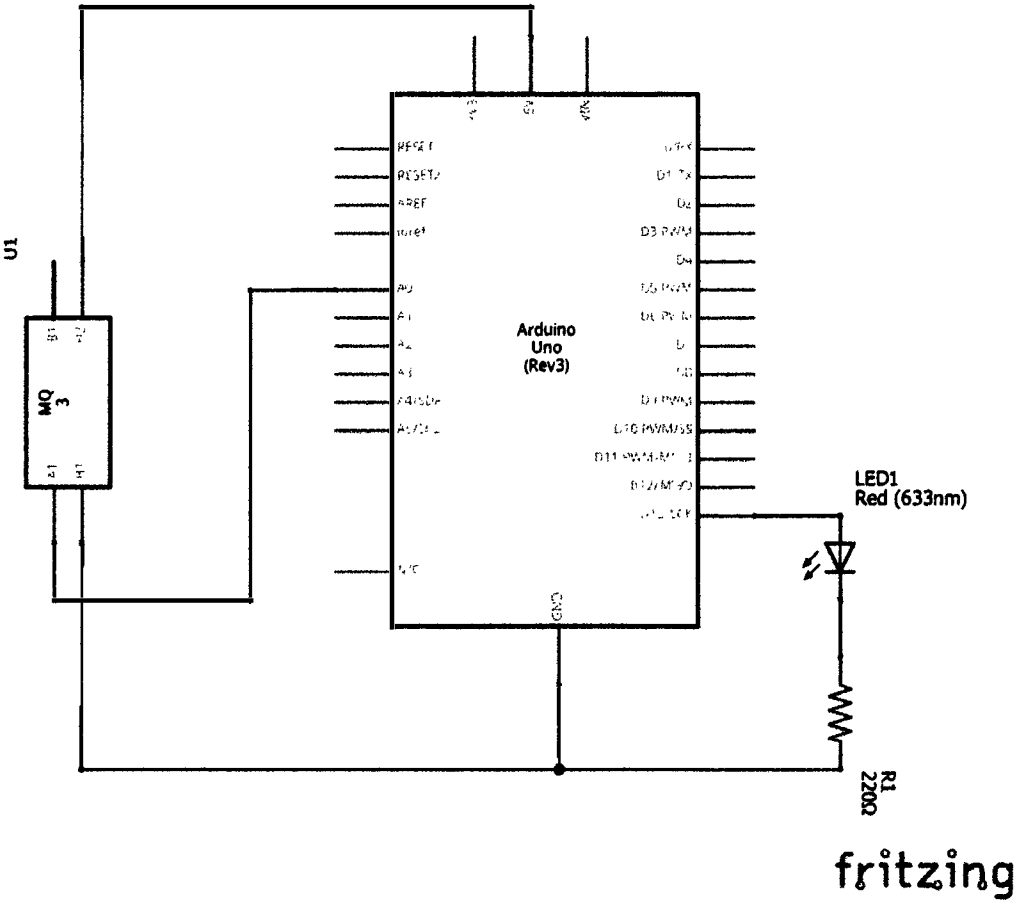


Рис. 71 ❖ Принципиальная схема подключения для практического занятия 15

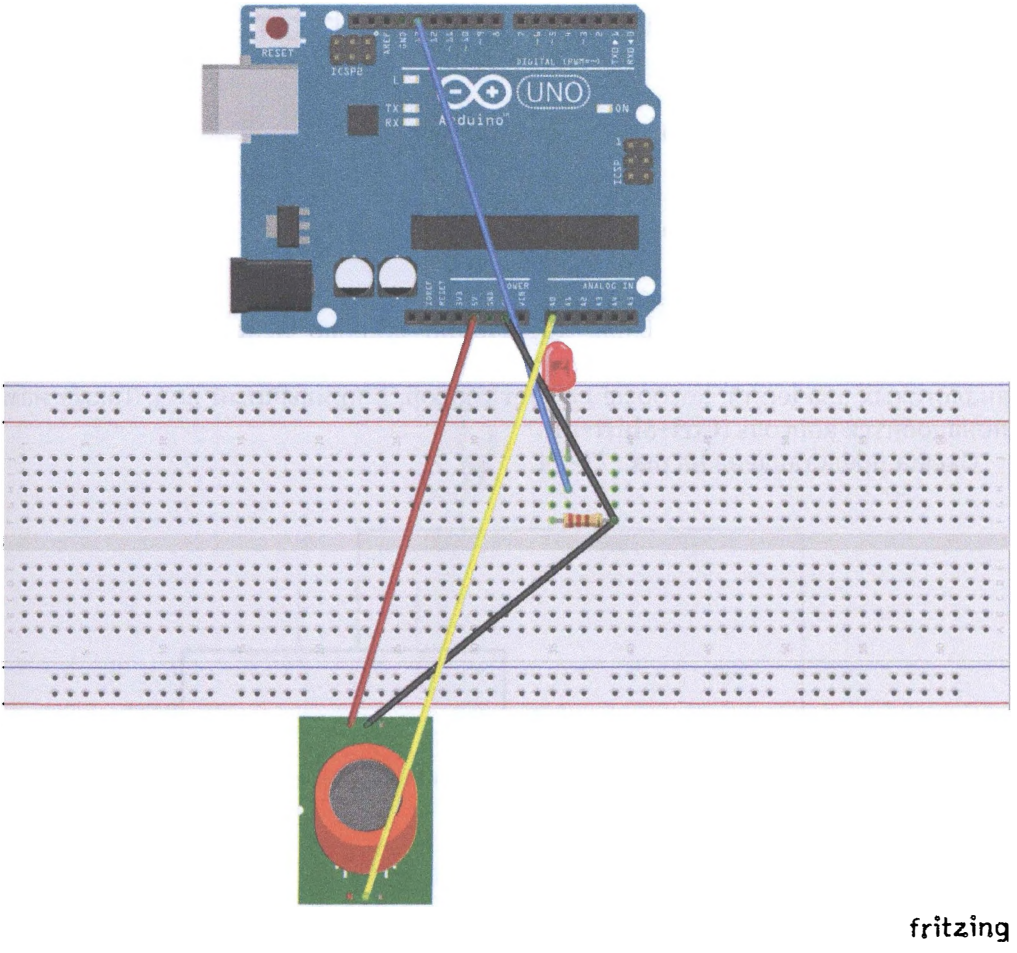


Рис. 72 ❖ Схема подключения с макетной платой для практического занятия 15

Код программы

```
int sensorPin = A0; // select the input pin for the potentiometer
int ledPin = 13; // select the pin for the LED
int sensorValue = 0; // variable to store the value coming from the sensor
void setup () {
pinMode (ledPin, OUTPUT);
Serial.begin (9600);
}
void loop () {
sensorValue = analogRead (sensorPin);
digitalWrite (ledPin, HIGH);
delay (sensorValue);
digitalWrite (ledPin, LOW);
delay (sensorValue);
Serial.println (sensorValue, DEC);
}
```

Практическое занятие 16. Эксперимент с температурным сенсором

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- температурный сенсор LM35;
- макетная плата;
- соединительные провода.

Температурный сенсор LM35 достаточно широко используется и прост в эксплуатации. Единственная нетривиальная часть этого задания – перевод аналоговых значений, которые выдаёт сенсор, в привычный вид. Также нам понадобится консоль (**Ctrl+Shift+M**).

Схема представлена на рис. 73–74.

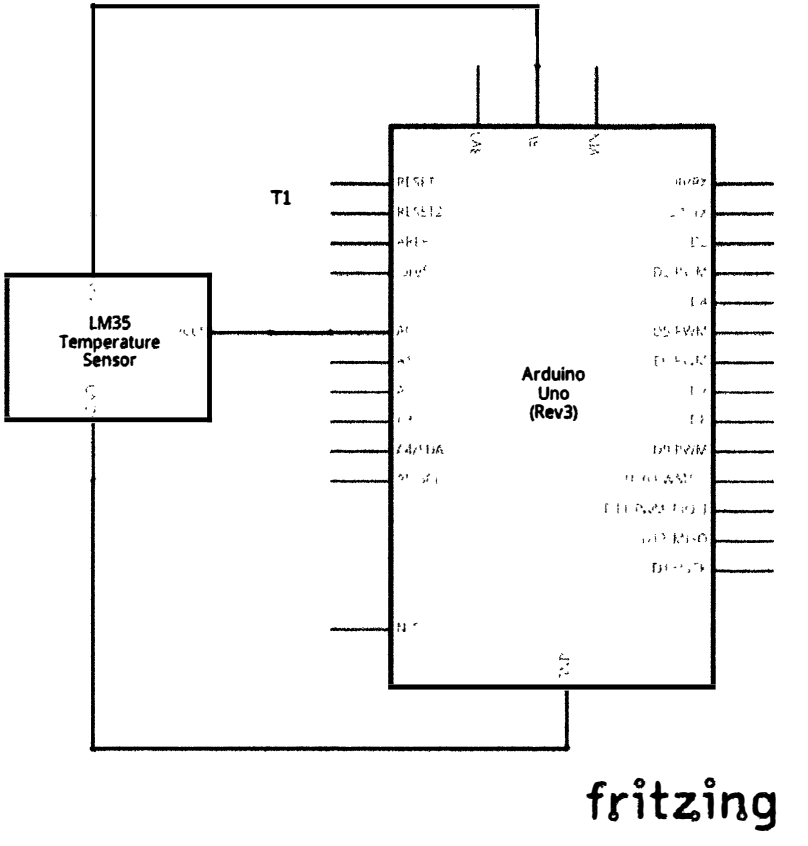


Рис. 73 ❖ Принципиальная схема подключения для практического занятия 16

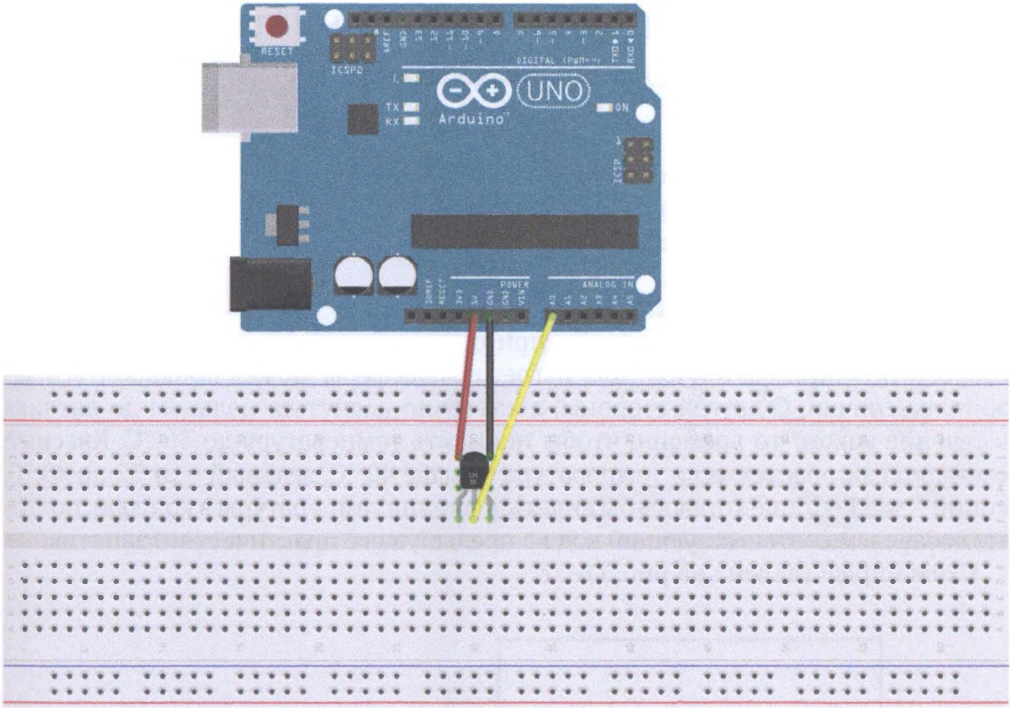


Рис. 74 ❖ Схема подключения с макетной платой для практического занятия 16

Код программы

```
int potPin = A0; // define the analog interface 0 LM35 temperature sensor connection
void setup ()
{
Serial.begin (9600) ;// set the baud rate
}
void loop ()
{
int val ;// define variables
int dat ;// define variables
val = analogRead (potPin) ;// read sensor analog values and assigned to val
dat = (125 * val) >> 8 ;// temperature formula
Serial.print ("Temp:") ;// output as a string represents the temperature display Tep
Serial.print (dat) ;// output shows the value of dat
Serial.println ("C") ;// output display as a C string
delay (500) ;// delay of 0.5 seconds
}
```

Практическое занятие 17. Разноцветный термостат

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- температурный сенсор LM35;
- 3 разноцветных светодиода;
- 3 резистора на 220 Ом;
- макетная плата;
- соединительные провода.

Суть этого задания в том, чтобы при определённой температуре зажигался соответствующий светодиод. Для нагрева атмосферы может понадобиться зажигалка/спички. С другой стороны, достаточно коснуться пальцем до датчика в течение какого-то времени, чтобы повысить температуру до 36 °C. Красный светодиод загорается, если температура больше 40 °C, зелёный – от 32 до 40 °C, синий – до 31 °C. Ещё хотелось бы видеть значения температуры в консоли, поэтому добавляем соответствующий код из предыдущего практического занятия.

Схема представлена на рис. 77–78.

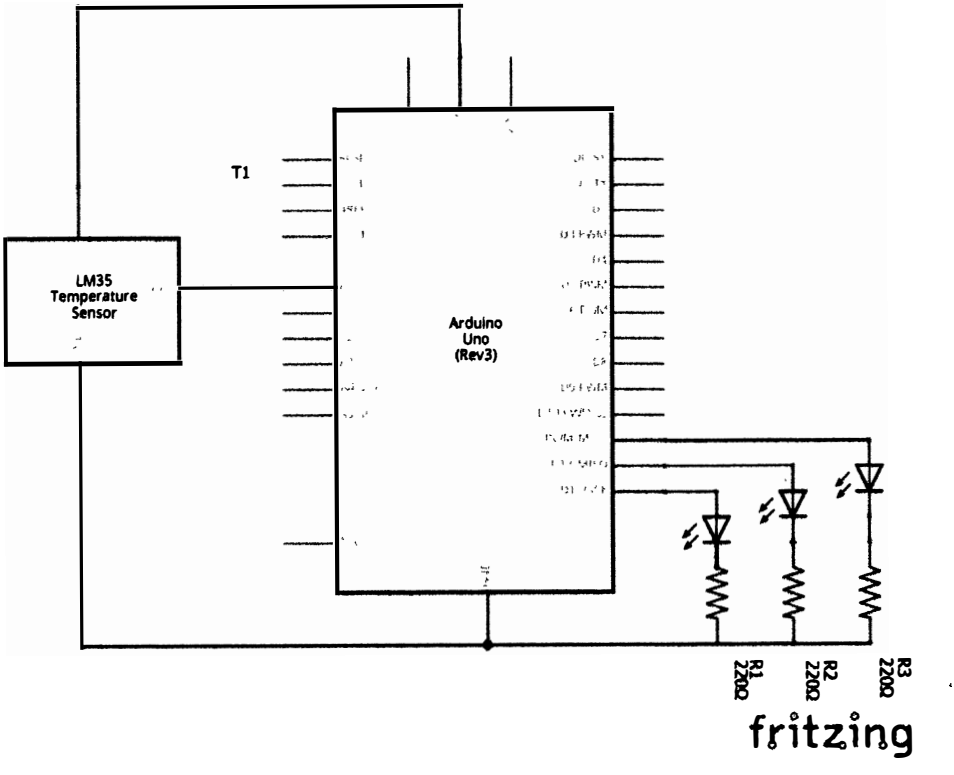


Рис. 75 ❖ Принципиальная схема подключения для практического занятия 17

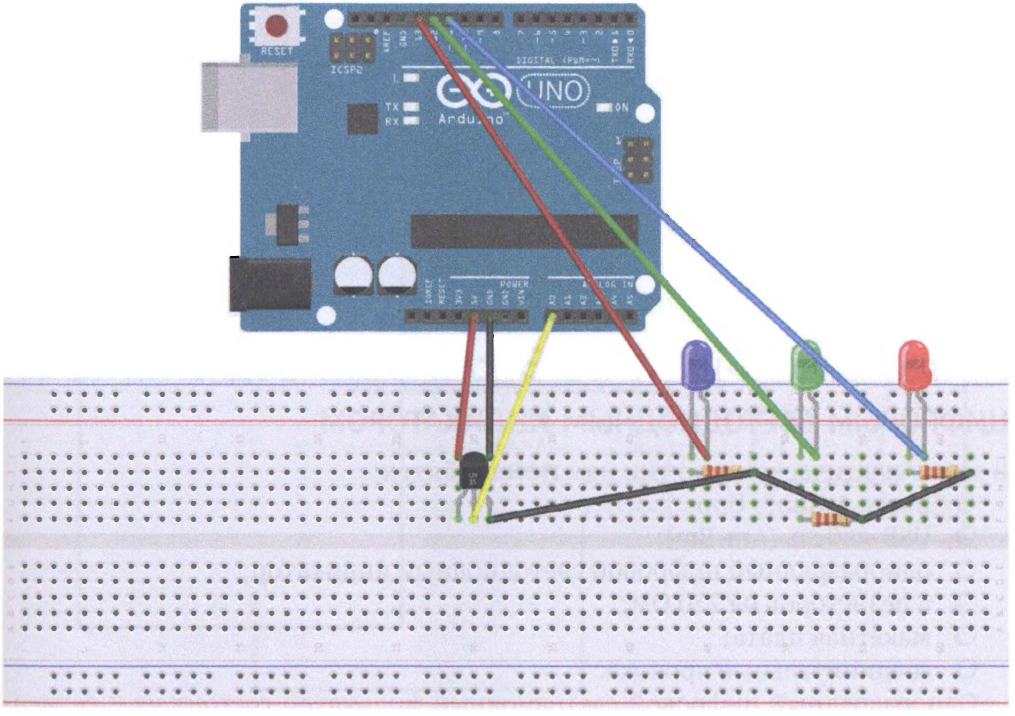


Рис. 76 ❖ Схема подключения с макетной платой для практического занятия 17

Код программы

```
   void setup () {
    pinMode (13, OUTPUT);
    pinMode (12, OUTPUT);
    pinMode (11, OUTPUT);
    Serial.begin (9600);
}
void loop () {
    int vol = analogRead (A0) * (5.0 / 1023.0 * 100); // read the LM35 temperature
    Serial.print ("Temp:") ;// output as a string represents the temperature display Tep
    Serial.print (vol) ;// output shows the value of dat
    Serial.println ("C") ;// output display as a C string
    if (vol <= 31) // set the value of the temperature of the low temperature region, and
led display
    {
        digitalWrite (13, HIGH);
        digitalWrite (12, LOW);
        digitalWrite (11, LOW);
    }
    else if (vol >= 32 && vol <= 40)
    {
        digitalWrite (13, LOW);
```

```
        digitalWrite (12, HIGH);
        digitalWrite (11, LOW);
    }
    else if (vol >= 41) // hot zone temperature setting
    {
        digitalWrite (13, LOW);
        digitalWrite (12, LOW);
        digitalWrite (11, HIGH);
    }
    delay (500);
}
```

Практическое занятие 18. Эксперимент с одноразрядным цифровым светодиодным индикатором

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- одноразрядный цифровой светодиодный индикатор;
- 8 резисторов на 220 Ом;
- макетная плата;
- соединительные провода.

Одноразрядный цифровой светодиодный индикатор состоит из 8 сегментов, каждый из которых представляет собой маленький светодиод, отвечающий за соответствующий сегмент цифры на индикаторе (или точки). Бывают два типа разноцветных светодиодов: с общим катодом (когда катодный выход один, а анодных несколько), который подключается к земле, и с общим анодом (анод один, катодов несколько), который подключается к 3.3 или 5 В выходу платы. В этом задании мы построим схему и напишем код так, чтобы индикатор показывал цифры от 1 до 8.

Схема представлена на рис. 77–78.

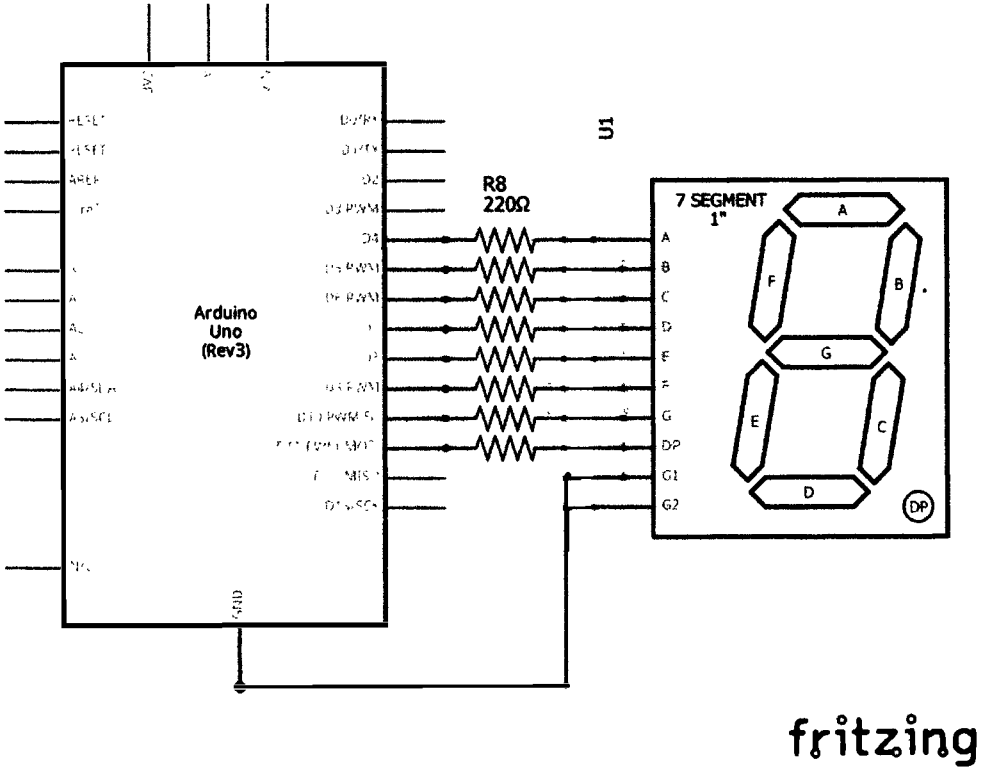


Рис. 77 ❖ Принципиальная схема подключения для практического занятия 18

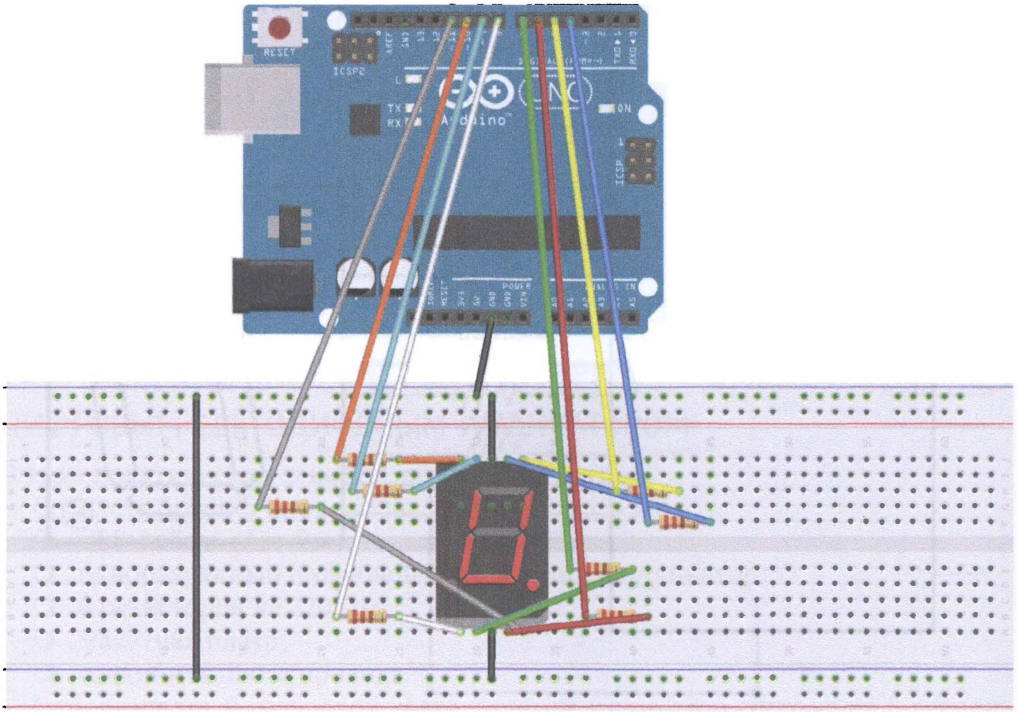


Рис. 78 ❖ Схема подключения с макетной платой для практического занятия 18

Код программы

```
// Set control each segment digital IO pin
    int a = 4 ;// define the digital interface to connect a seven segment LED
    int b = 5 ;// define the connection b Digital Interface 6-segment LED
    int c = 6 ;// define paragraph (c) Digital Interface 5 digital connection
    int d = 7 ;// define the digital interface 11 is connected to d-segment digital tube
    int e = 8 ;// define the digital interface 10 is connected to e-segment digital tube
    int f = 9 ;// define the digital interface 8 digital tube connection f
    int g = 10 ;// define the digital interface 9 g of the digital control connection
    int dp = 11 ;// define the digital interface 4 digital tube connecting dp
    void digital_1 (void) // display the number 1
    {
        digitalWrite (a, LOW);
        digitalWrite (b, HIGH);
        digitalWrite (c, HIGH);
        digitalWrite (d, LOW);
        digitalWrite (e, LOW);
        digitalWrite (f, LOW);
        digitalWrite (g, LOW);
        digitalWrite (dp, LOW);
    }
    void digital_2 (void) // display number 2
```

```
{
    digitalWrite (a, HIGH);
    digitalWrite (b, HIGH);
    digitalWrite (c, LOW);
    digitalWrite (d, HIGH);
    digitalWrite (e, HIGH);
    digitalWrite (f, LOW);
    digitalWrite (g, HIGH);
    digitalWrite (dp, LOW);
}
void digital_3 (void) // display the number 3
{
    digitalWrite (a, HIGH);
    digitalWrite (b, HIGH);
    digitalWrite (c, HIGH);
    digitalWrite (d, HIGH);
    digitalWrite (e, LOW);
    digitalWrite (f, LOW);
    digitalWrite (g, HIGH);
    digitalWrite (dp, LOW);
}
void digital_4 (void) // show 4
{
    digitalWrite (a, LOW);
    digitalWrite (b, HIGH);
    digitalWrite (c, HIGH);
    digitalWrite (d, LOW);
    digitalWrite (e, LOW);
    digitalWrite (f, HIGH);
    digitalWrite (g, HIGH);
    digitalWrite (dp, LOW);
}
void digital_5 (void) // display the number 5
{
    digitalWrite (a, HIGH);
    digitalWrite (b, LOW);
    digitalWrite (c, HIGH);
    digitalWrite (d, HIGH);
    digitalWrite (e, LOW);
    digitalWrite (f, HIGH);
    digitalWrite (g, HIGH);
    digitalWrite (dp, LOW);
}
void digital_6 (void) // display the number 6
{
    digitalWrite (a, HIGH);
    digitalWrite (b, LOW);
    digitalWrite (c, HIGH);
    digitalWrite (d, HIGH);
    digitalWrite (e, HIGH);
    digitalWrite (f, HIGH);
    digitalWrite (g, HIGH);
```

```
    digitalWrite (dp, LOW);
}
void digital_7 (void) // display the number 7
{
    digitalWrite (a, HIGH);
    digitalWrite (b, HIGH);
    digitalWrite (c, HIGH);
    digitalWrite (d, LOW);
    digitalWrite (e, LOW);
    digitalWrite (f, LOW);
    digitalWrite (g, LOW);
    digitalWrite (dp, LOW);
}
void digital_8 (void) // display the number 8
{
    digitalWrite (a, HIGH);
    digitalWrite (b, HIGH);
    digitalWrite (c, HIGH);
    digitalWrite (d, HIGH);
    digitalWrite (e, HIGH);
    digitalWrite (f, HIGH);
    digitalWrite (g, HIGH);
    digitalWrite (dp, LOW);
}
void setup ()
{
int i ;// define variables
for (i = 4; i <= 11; i++)
pinMode (i, OUTPUT) ;// set 4 to 11 pin to output mode
}
void loop ()
{
while (1)
{
digital_1 () ;// display numbers 1
delay (1000) ;// delay 2s
digital_2 () ;// display number 2
delay (1000); // delay 1s
digital_3 () ;// display the number 3
delay (1000); // delay 1s
digital_4 () ;// show 4
delay (1000); // delay 1s
digital_5 () ;// display the number 5
delay (1000); // delay 1s
digital_6 () ;// display the number 6
delay (1000); // delay 1s
digital_7 () ;// display the number 7
delay (1000); // delay 1s
digital_8 () ;// display the number 8
delay (1000); // delay 1s
}
}
```

Думаю, вам не составит особого труда **добавить ещё 2 цифры – 0 и 9!**

Практическое занятие 19. Эксперимент с четырёхразрядным цифровым светодиодным индикатором

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- четырёхразрядный цифровой светодиодный индикатор;
- 8 резисторов на 220 Ом;
- макетная плата;
- соединительные провода.

У четырёхразрядного цифрового светодиодного индикатора 12 разъёмов, по сравнению с одноразрядным, у которого 10. В принципе, не намного больше. Нумеруются они против часовой стрелки: если положить перед собой индикатор, внизу слева направо будут пины 1–6, а вверху справа налево – 7–12. Соберём схему и напишем программу, которая будет показывать значения от 0:00 до 9:00 в цикле.

Схема представлена на рис. 79–80.

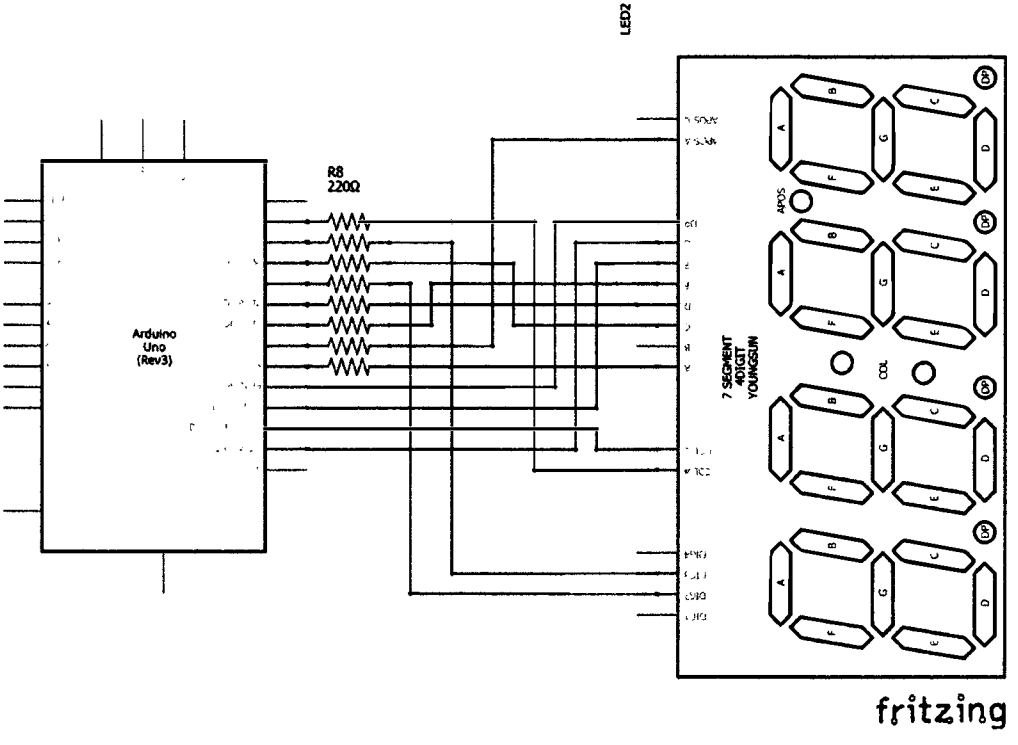


Рис. 79 ❖ Принципиальная схема подключения для практического занятия 19

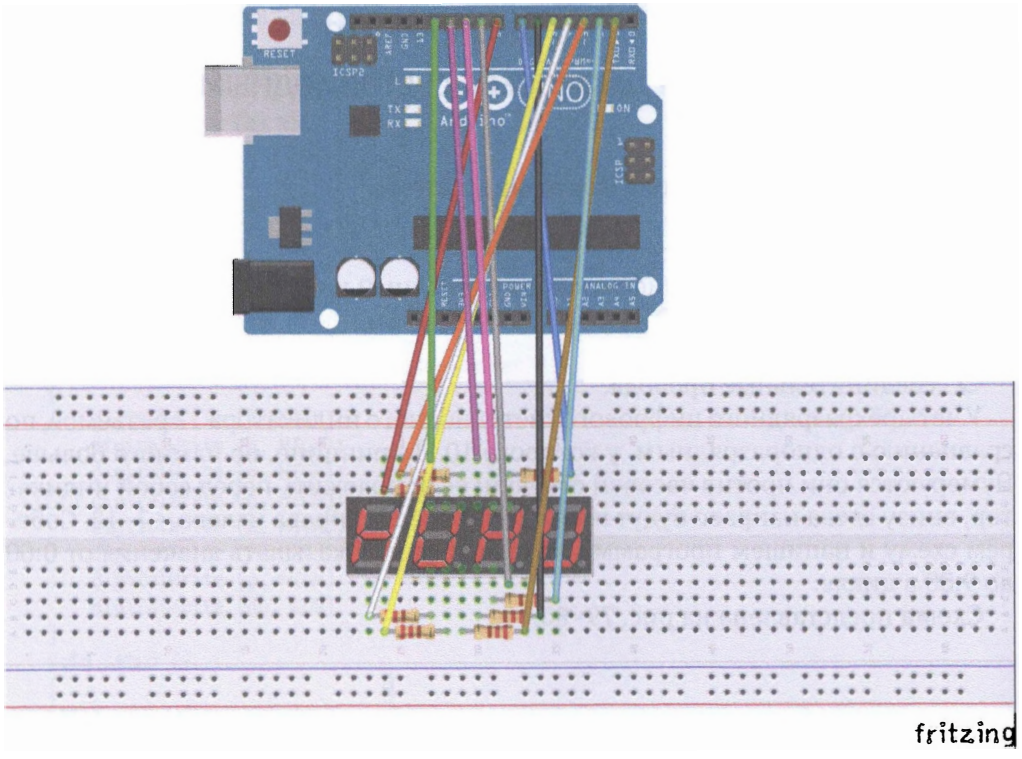


Рис. 80 ❖ Схема подключения с макетной платой для практического занятия 19

Код программы

```
int a = 8;
int b = 7;
int c = 6;
int d = 5;
int e = 4;
int f = 3;
int g = 2;
int p = 1;

int d4 = 9;
int d3 = 10;
int d2 = 11;
int d1 = 12;

long n = 0;
int x = 100;
int del = 55;

void setup()
{
    pinMode(d1, OUTPUT);
    pinMode(d2, OUTPUT);
```

```
    pinMode(d3, OUTPUT);
    pinMode(d4, OUTPUT);
    pinMode(a, OUTPUT);
    pinMode(b, OUTPUT);
    pinMode(c, OUTPUT);
    pinMode(d, OUTPUT);
    pinMode(e, OUTPUT);
    pinMode(f, OUTPUT);
    pinMode(g, OUTPUT);
    pinMode(p, OUTPUT);
}

void loop()
{
    clearLEDs();
    pickDigit(1);
    pickNumber((n/x/1000)%10);
    delayMicroseconds(del);

    clearLEDs();
    pickDigit(2);
    pickNumber((n/x/100)%10);
    delayMicroseconds(del);

    clearLEDs();
    pickDigit(3);
    dispDec(3);
    pickNumber((n/x/10)%10);
    delayMicroseconds(del);

    clearLEDs();
    pickDigit(4);
    pickNumber(n/x%10);
    delayMicroseconds(del);

    n++;

    if (digitalRead(13) == LOW)
    {
        n = 0;
    }
}

void pickDigit(int x)
{
    digitalWrite(d1, HIGH);
    digitalWrite(d2, HIGH);
    digitalWrite(d3, HIGH);
    digitalWrite(d4, HIGH);

    switch(x)
    {
        case 1:
            digitalWrite(d1, LOW);
            break;
```

```
        case 2:
            digitalWrite(d2, LOW);
            break;
        case 3:
            digitalWrite(d3, LOW);
            break;
        default:
            digitalWrite(d4, LOW);
            break;
    }
}

void pickNumber(int x)
{
    switch(x)
    {
        default:
            zero();
            break;
        case 1:
            one();
            break;
        case 2:
            two();
            break;
        case 3:
            three();
            break;
        case 4:
            four();
            break;
        case 5:
            five();
            break;
        case 6:
            six();
            break;
        case 7:
            seven();
            break;
        case 8:
            eight();
            break;
        case 9:
            nine();
            break;
    }
}

void dispDec(int x)
{
    digitalWrite(p, LOW);
}
```

```
void clearLEDs()
{
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(p, LOW);
}

void zero()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, LOW);
}

void one()
{
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
}

void two()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, LOW);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, LOW);
    digitalWrite(g, HIGH);
}

void three()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, HIGH);
```

```
}
void four()
{
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
}
void five()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
}
void six()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
}
void seven()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
}
void eight()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
```

```
    digitalWrite(g, HIGH);
}
void nine()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
}
```

Практическое занятие 20. Эксперимент со светодиодной матрицей

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- светодиодная матрица 8×8;
- 8 резисторов на 220 Ом;
- макетная плата;
- соединительные провода.

Светодиодная матрица 8×8 содержит 64 светодиода, соединённых так, чтобы можно было зажечь соответствующий светодиод, пользуясь его *X*- и *У*-координатой и подавая, соответственно, на одну координату землю через резистор, а на другую – управляющий сигнал с одного из цифровых пинов-выходов платы Arduino Uno. К сожалению, в parts-библиотеке Fritzing не нашлось соответствующего элемента, поэтому пришлось пририсовывать светодиодной матрице 8×8 с 12 выходами ещё 4 на схеме с макетной платой и рисовать с нуля принципиальную схему. Напишем два кода: первый будет просто зажигать и гасить верхний левый светодиод на матрице, второй – выводить буквы английского алфавита от A до I.

Схема представлена на рис. 81–82.

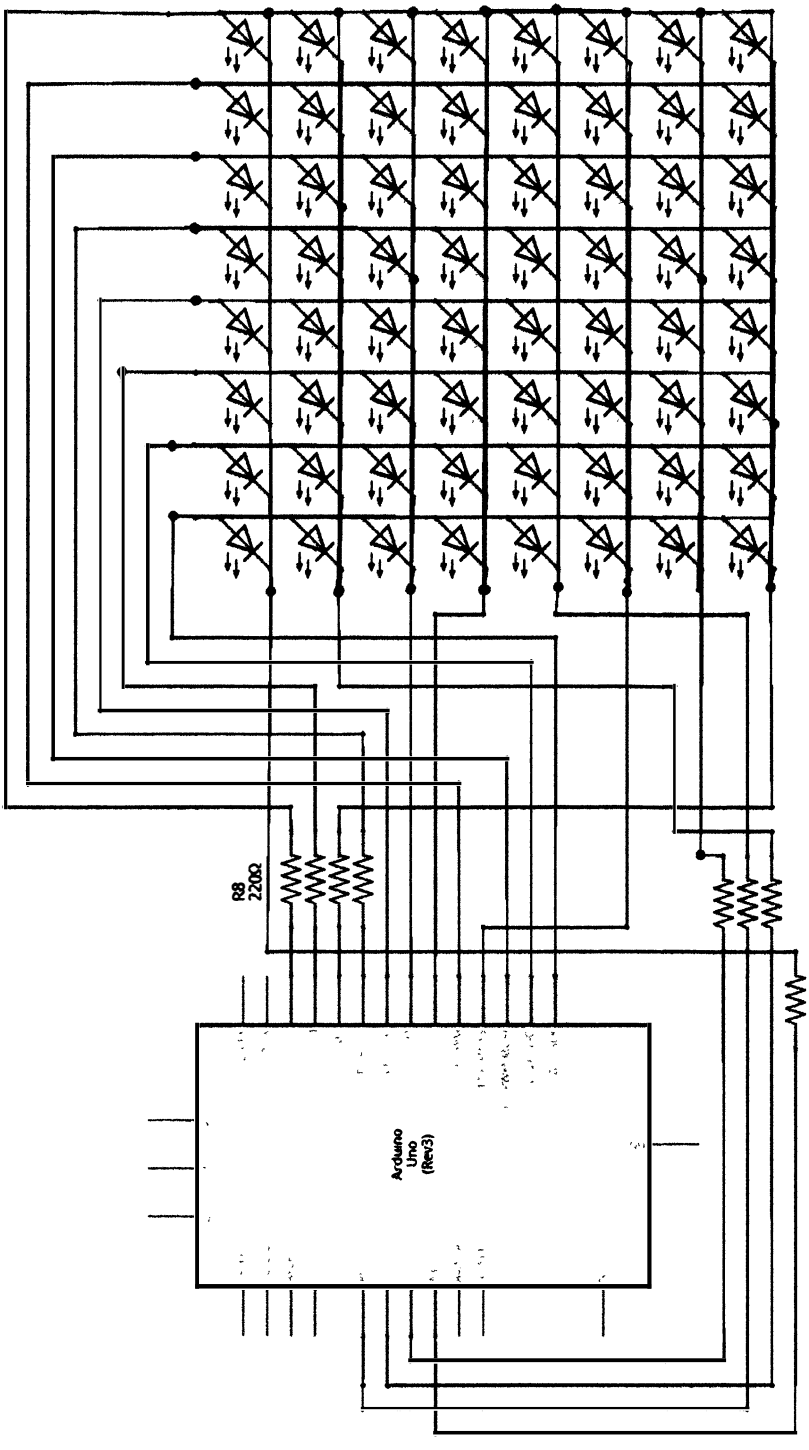


Рис. 81 ❖ Принципиальная схема подключения для практического занятия 20

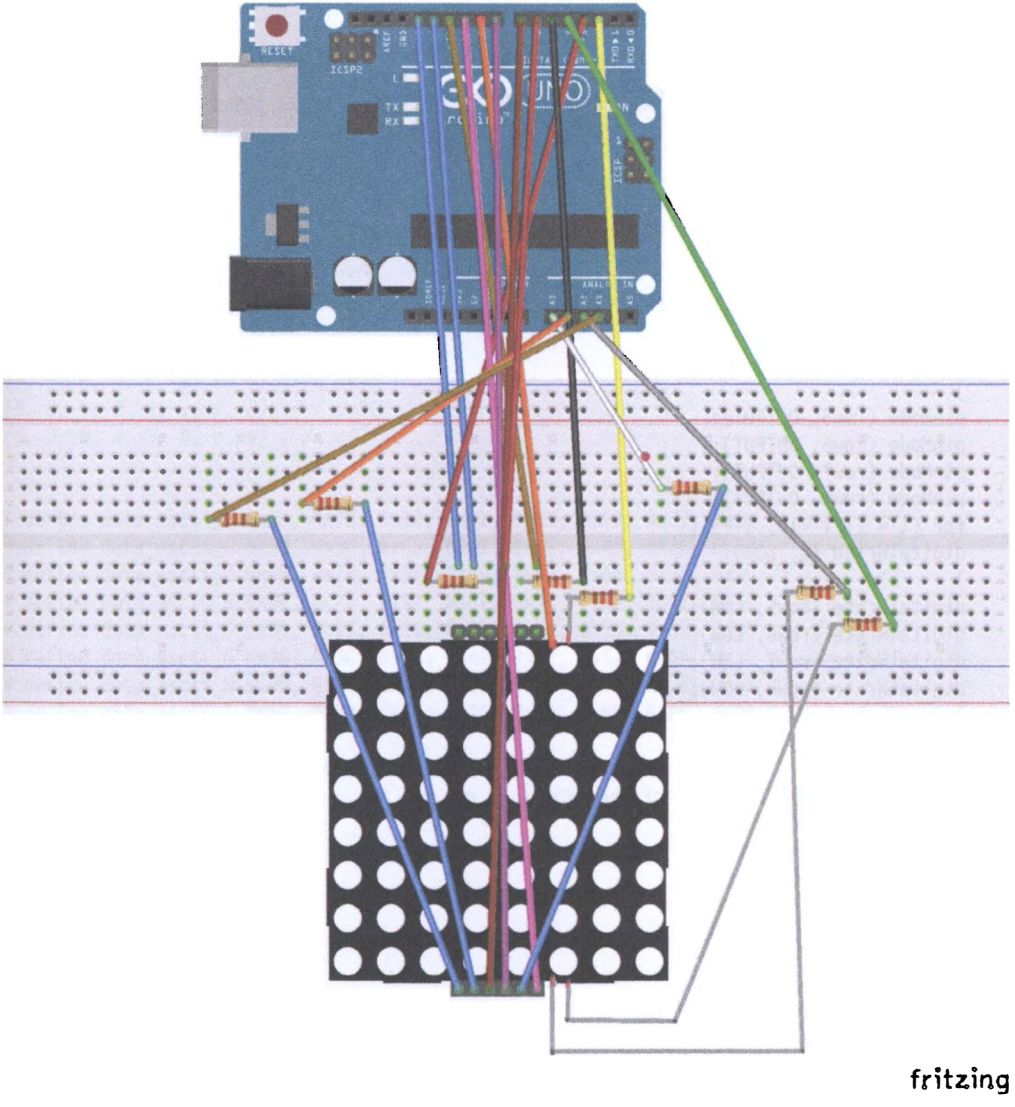


Рис. 82 ❖ Схема подключения с макетной платой для практического занятия 20

Код программы 1

```
const int row1 = 2; // the number of the row pin 9
const int row2 = 3; // the number of the row pin 14
const int row3 = 4; // the number of the row pin
const int row4 = 5; // the number of the row pin 12
const int row5 = 17; // the number of the row pin 1
const int row6 = 16; // the number of the row pin 7
const int row7 = 15; // the number of the row pin 2
const int row8 = 14; // the number of the row pin 5
```

```
// The pin to control COl
const int col1 = 6; // the number of the col pin 13
const int col2 = 7; // the number of the col pin 3
const int col3 = 8; // the number of the col pin 4
const int col4 = 9; // the number of the col pin 10
const int col5 = 10; // the number of the col pin 6
const int col6 = 11; // the number of the col pin 11
const int col7 = 12; // the number of the col pin 15
const int col8 = 13; // the number of the col pin 16
void setup () {
int i = 0;
for (i = 2; i<18; i++)
    {
    pinMode (i, OUTPUT);
    }
    pinMode (row5, OUTPUT);
    pinMode (row6, OUTPUT);
    pinMode (row7, OUTPUT);
    pinMode (row8, OUTPUT);
    for (i = 2; i <18; i++) {
    digitalWrite (i, LOW);
    }
    digitalWrite (row5, LOW);
    digitalWrite (row6, LOW);
    digitalWrite (row7, LOW);
    digitalWrite (row8, LOW);
    }
    void loop () {
    int i;
    // The row # 1 and col # 1 of the LEDs turn on
    digitalWrite (row1, HIGH);
    digitalWrite (row2, LOW);
    digitalWrite (row3, LOW);
    digitalWrite (row4, LOW);
    digitalWrite (row5, LOW);
    digitalWrite (row6, LOW);
    digitalWrite (row7, LOW);
    digitalWrite (row8, LOW);
    digitalWrite (col1, LOW);
    digitalWrite (col2, HIGH);
    digitalWrite (col3, HIGH);
    digitalWrite (col4, HIGH);
    digitalWrite (col5, HIGH);
    digitalWrite (col6, HIGH);
    digitalWrite (col7, HIGH);
    digitalWrite (col8, HIGH);
    delay (1000);
    // Turn off all
    for (i = 2; i <18; i++) {
    digitalWrite (i, LOW);
    }
    delay (1000);
    }
```

Код программы 2

```
# define display_array_size 8
// Ascii 8x8 dot font
# define data_null 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 // null char
# define data_ascii_A 0x02, 0x0C, 0x18, 0x68, 0x68, 0x18, 0x0C, 0x02 /* «A», 0 */
/**
** «A»
# Define A {//
{0, 0, 0, 0, 0, 0, 1, 0}, / / 0x02
{0, 0, 0, 0, 1, 1, 0, 0}, / / 0x0C
{0, 0, 0, 1, 1, 0, 0, 0}, / / 0x18
{0, 1, 1, 0, 1, 0, 0, 0}, / / 0x68
{0, 1, 1, 0, 1, 0, 0, 0}, / / 0x68
{0, 0, 0, 1, 1, 0, 0, 0}, / / 0x18
{0, 0, 0, 0, 1, 1, 0, 0}, / / 0x0C
{0, 0, 0, 0, 0, 0, 1, 0} / / 0x02
}
**/
# define data_ascii_B 0x00, 0x7E, 0x52, 0x52, 0x52, 0x52, 0x2C, 0x00 /* «B», 1 */
# define data_ascii_C 0x00, 0x3C, 0x66, 0x42, 0x42, 0x42, 0x2C, 0x00 /* «C», 2 */
# define data_ascii_D 0x00, 0x7E, 0x42, 0x42, 0x42, 0x66, 0x3C, 0x00 /* «D», 3 */
# define data_ascii_E 0x00, 0x7E, 0x52, 0x52, 0x52, 0x52, 0x52, 0x42 /* «E», 4 */
# define data_ascii_F 0x00, 0x7E, 0x50, 0x50, 0x50, 0x50, 0x50, 0x40 /* «F», 5 */
# define data_ascii_G 0x00, 0x3C, 0x66, 0x42, 0x42, 0x52, 0x16, 0x1E /* «G», 6 */
# define data_ascii_H 0x00, 0x7E, 0x10, 0x10, 0x10, 0x10, 0x7E, 0x00 /* «H», 7 */
# define data_ascii_I 0x00, 0x00, 0x00, 0x7E, 0x00, 0x00, 0x00, 0x00 /* «I», 8 */
// Display array
byte data_ascii [] [display_array_size] = {
data_null,
data_ascii_A, data_ascii_B,
data_ascii_C,
data_ascii_D,
data_ascii_E,
data_ascii_F,
data_ascii_G,
data_ascii_H,
data_ascii_I,
};
// The pin to control ROW
const int row1 = 2; // the number of the row pin 24
const int row2 = 3; // the number of the row pin 23
const int row3 = 4; // the number of the row pin 22
const int row4 = 5; // the number of the row pin 21
const int row5 = 17; // the number of the row pin 4
const int row6 = 16; // the number of the row pin 3
const int row7 = 15; // the number of the row pin 2
const int row8 = 14; // the number of the row pin 1
// The pin to control COl
const int col1 = 6; // the number of the col pin 20
const int col2 = 7; // the number of the col pin 19
const int col3 = 8; // the number of the col pin 18
const int col4 = 9; // the number of the col pin 17
```

```
const int col5 = 10; // the number of the col pin 16
const int col6 = 11; // the number of the col pin 15
const int col7 = 12; // the number of the col pin 14
const int col8 = 13; // the number of the col pin 13
void displayNum (byte rowNum, int colNum)
  {
     int j;
     byte temp = rowNum;
     for (j = 2; j <6; j++)
       {
          digitalWrite (j, LOW);
       }
    digitalWrite (row5, LOW);
    digitalWrite (row6, LOW);
    digitalWrite (row7, LOW);
    digitalWrite (row8, LOW);
    for (j = 6; j <14; j++)
      {
        digitalWrite (j, HIGH);}
        switch (colNum)
      {
    case 1: digitalWrite (col1, LOW); break;
    case 2: digitalWrite (col2, LOW); break;
    case 3: digitalWrite (col3, LOW); break;
    case 4: digitalWrite (col4, LOW); break;
    case 5: digitalWrite (col5, LOW); break;
    case 6: digitalWrite (col6, LOW); break;
    case 7: digitalWrite (col7, LOW); break;
    case 8: digitalWrite (col8, LOW); break;
    default: break;
    }
    for (j = 1; j <9; j++)
      {
       temp = (0x80) & (temp);
       if (temp == 0)
         {
           temp = rowNum << j;
           continue;
         }
        switch (j)
        {
         case 1: digitalWrite (row1, HIGH); break;
         case 2: digitalWrite (row2, HIGH); break;
         case 3: digitalWrite (row3, HIGH); break;
         case 4: digitalWrite (row4, HIGH); break;
         case 5: digitalWrite (row5, HIGH); break;
         case 6: digitalWrite (row6, HIGH); break;
         case 7: digitalWrite (row7, HIGH); break;
         case 8: digitalWrite (row8, HIGH); break;
         default: break;
        }
      temp = rowNum << j;
```

```
  }
}

void setup () {
int i = 0;
for (i = 2; i <18; i++)
{
pinMode (i, OUTPUT);
}
for (i = 2; i <18; i++) {
digitalWrite (i, LOW);
}
}
void loop () {
int t1;
int l;
int arrage;
for (arrage = 0; arrage <10; arrage++)
{
for (l = 0; l <512; l++)
  {
    for (t1 = 0; t1 <8; t1++)
      {
        displayNum (data_ascii [arrage] [t1], (t1 +1));
      }
   }
}
}
```

Практическое занятие 21. Эксперимент с трёхцветным светодиодом

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- трёхцветный светодиод;
- 3 резистора на 220 Ом;
- макетная плата;
- соединительные провода.

В этом занятии мы сначала выведем красный, белый и синий цвета с помощью трёхцветного (RGB) светодиода, а затем выведем смешанные цвета. У этого светодиода 4 выхода: 3 анода (красный, жёлтый, синий цвета) и 1 катод (на землю). Такой светодиод называется светодиодом с общим катодом. Есть также светодиод с общим анодом, когда трём цветам соответствуют 3 катодных выхода. В комплекте Arduino Uno Starter Learning Kit с RFID-модулем нет отдельного трёхцветного светодиода, но есть небольшой модуль со встроенным трёхцветным светодиодом, который мы и будем использовать. Вообще,

на модуле уже есть встроенные сопротивления, но на всякий случай лучше всё-таки использовать дополнительные внешние резисторы на 220 Ом. Обратите внимание на маркировку пинов на модуле с трёхцветным светодиодом – она может отличаться от приведённой ниже схемы подключения (например: R, G, B подключаются, соответственно, к 11, 9, 10 и GND пинам платы)!

Схема представлена на рис. 83–84.

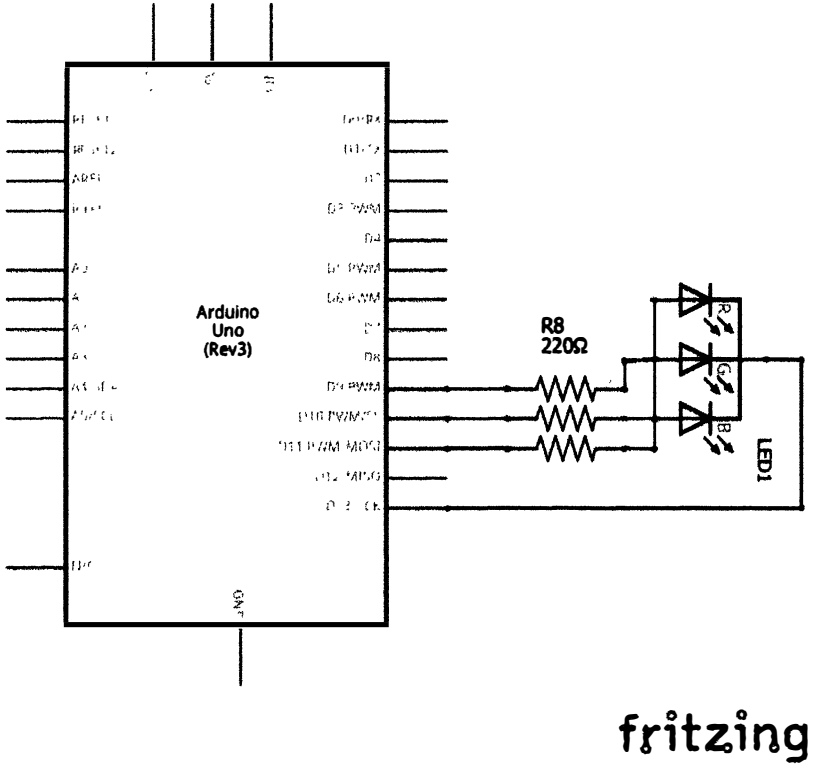


Рис. 83 ❖ Принципиальная схема подключения для практического занятия 21

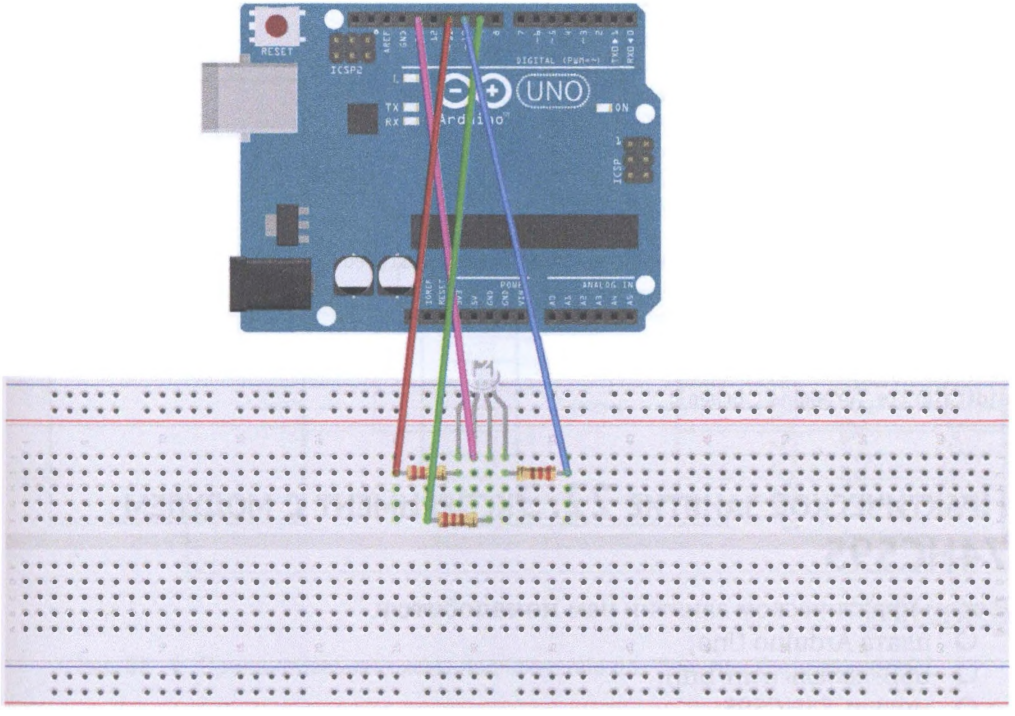


Рис. 84 ❖ Схема подключения с макетной платой для практического занятия 21

Код программы

```
int ledPin = 13; // LED is connected to digital pin 13
int redPin = 11; // R petal on RGB LED module connected to digital pin 11
int greenPin = 9; // G petal on RGB LED module connected to digital pin 9
int bluePin = 10; // B petal on RGB LED module connected to digital pin 10
void setup ()
{
pinMode (ledPin, OUTPUT); // sets the ledPin to be an output
pinMode (redPin, OUTPUT); // sets the redPin to be an output
pinMode (greenPin, OUTPUT); // sets the greenPin to be an output
pinMode (bluePin, OUTPUT); // sets the bluePin to be an output
}
void loop () // run over and over again
{
// Basic colors:
color (255, 0, 0); // turn the RGB LED red
delay (1000); // delay for 1 second
color (0,255, 0); // turn the RGB LED green
delay (1000); // delay for 1 second
color (0, 0, 255); // turn the RGB LED blue
delay (1000); // delay for 1 second
// Example blended colors:
```

```
color (255,255,0); // turn the RGB LED yellow
delay (1000); // delay for 1 second
color (255,255,255); // turn the RGB LED white
delay (1000); // delay for 1 second
color (128,0,255); // turn the RGB LED purple
delay (1000); // delay for 1 second
color (0,0,0); // turn the RGB LED off
delay (1000); // delay for 1 second
}
void color (unsigned char red, unsigned char green, unsigned char blue) // the color
generating function
{
digitalWrite (redPin, red);
digitalWrite (bluePin, blue);
digitalWrite (greenPin, green);
}
```

Практическое занятие 22. Эксперимент с модулем 74HC595

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- модуль 74HC595;
- 8 светодиодов;
- 8 резисторов на 220 Ом;
- макетная плата;
- соединительные провода.

Зачем использовать модуль 74HC595, у которого есть сдвиговые регистры и память, для работы 8 светодиодов? Дело в том, что на любой плате есть ограниченное количество входов и выходов (пинов). Для работы 8 светодиодов нам надо занять 9 пинов платы (вместе с землёй). Рассматриваемый модуль сократит количество задействованных пинов до 5. **Будьте внимательны, подключайте модуль правильно, иначе плата будет перегреваться и выключаться.**

Схема представлена на рис. 85–86.

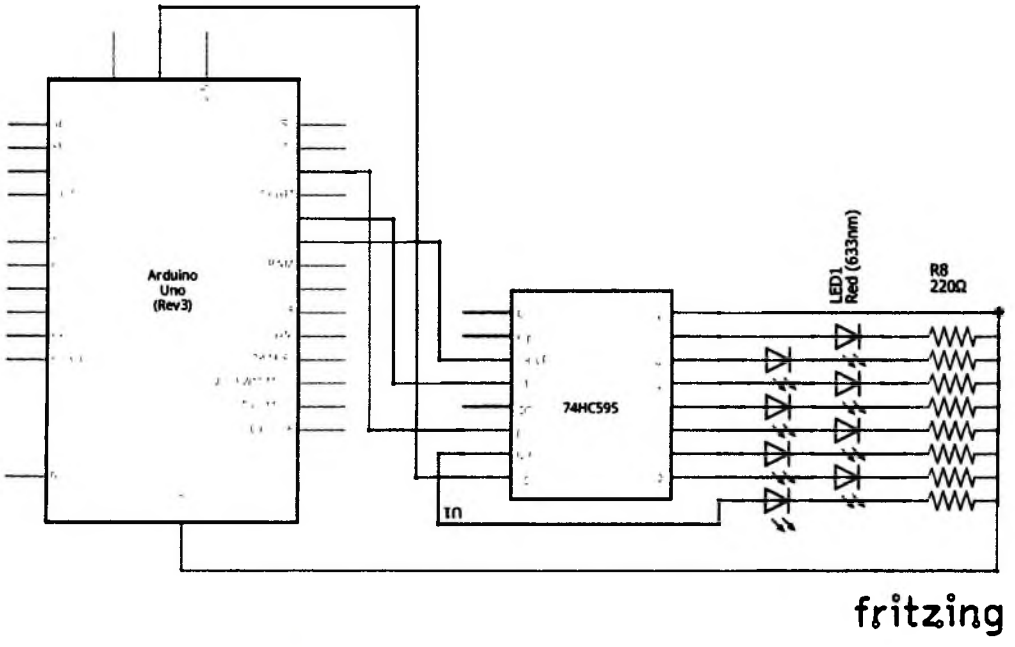


Рис. 85 ❖ Принципиальная схема подключения для практического занятия 22

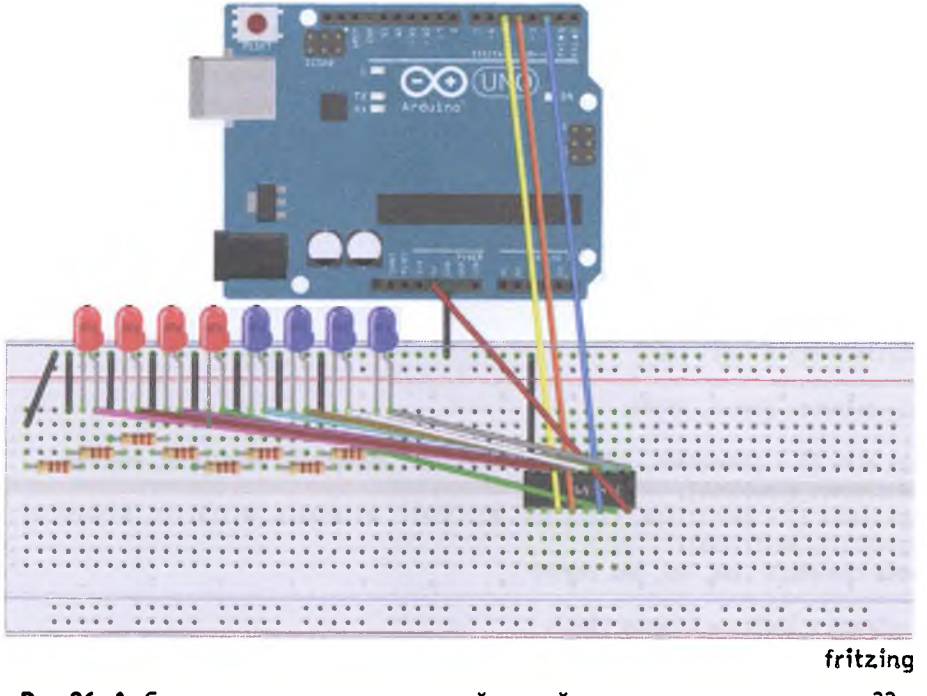


Рис. 86 ❖ Схема подключения с макетной платой для практического занятия 22

Код программы

```
int data = 2;
int clock = 4;
int latch = 5;
int ledState = 0;
const int ON = HIGH;
const int OFF = LOW;
void setup ()
{
    pinMode (data, OUTPUT);
    pinMode (clock, OUTPUT);
    pinMode (latch, OUTPUT);
}
void loop ()
    {
    int delayTime = 100;
    for (int i = 0; i <256; i++)
    {
    updateLEDs (i);
    delay (delayTime);
    }
    }
    void updateLEDs (int value)
    {
    digitalWrite (latch, LOW);
    shiftOut (data, clock, MSBFIRST, value);
    digitalWrite (latch, HIGH);
    }
    void updateLEDsLong (int value)
    {
    digitalWrite (latch, LOW);
    for (int i = 0; i <8; i++)
    {
    int bit = value & B10000000;
    value = value << 1;
    if (bit == 128) {digitalWrite (data, HIGH);}
    else {digitalWrite (data, LOW);}
    digitalWrite (clock, HIGH);
    delay (1);
    digitalWrite (clock, LOW);
    }
    digitalWrite (latch, HIGH);
    }
    int bits [] = {B00000001, B00000010, B00000100, B00001000, B00010000, B00100000,
    B01000000, B10000000};
    int masks [] = {B11111110, B11111101, B11111011, B11110111, B11101111, B11011111,
    B10111111, B01111111};
    void changeLED (int led, int state)
    {
    ledState = ledState & masks [led];
    if (state == ON) {ledState = ledState | bits [led];}
    updateLEDs (ledState);
    }
```

Практическое занятие 23. Кнопочный модуль 4×4 и библиотеки

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- кнопочный модуль 4×4;
- соединительные провода.

В этом занятии мы будем использовать кнопочный модуль 4×4 с 16 кнопками. В библиотеке компонентов Fritzing есть похожий компонент Buttonpad-4×4 в разделе Sparkfun-Electromechanical, однако внутри кнопки никак не соединены между собой и присутствуют светодиоды, которые так же висят в воздухе, как и кнопки, и вообще нам не нужны; поэтому для принципиальной схемы пришлось соединять кнопки между собой и выводить необходимые в занятии 8 выходов компонента. Также нам понадобится консоль, или монитор порта (**Ctrl+Shift+M**), чтобы смотреть результаты нажатия кнопок. Кроме того, нам понадобится библиотека Keypad.h, которую можно скачать с официального сайта Arduino по адресу [5]. Чтобы добавить библиотеку в проект Arduino IDE, нужно выбрать пункт меню **Скетч – Подключить библиотеку – Добавить .ZIP библиотеку...**, выбрать скачанный архив Keypad.zip и затем снова нажать **Скетч – Подключить библиотеку**, после чего выбрать библиотеку Keypad – она будет внизу списка. При этом в начало кода добавляется строка `#include <Keypad.h>`. Первая программа выводит символы в соответствии с нажатой кнопкой, вторая – задействует встроенный мини-светодиод на плате, который горит, пока нажата кнопка, ответственная за знак `'*'`, и загорается или гаснет по нажатии кнопки, ответственной за символ `'#'`.

Схема представлена на рис. 87–88.

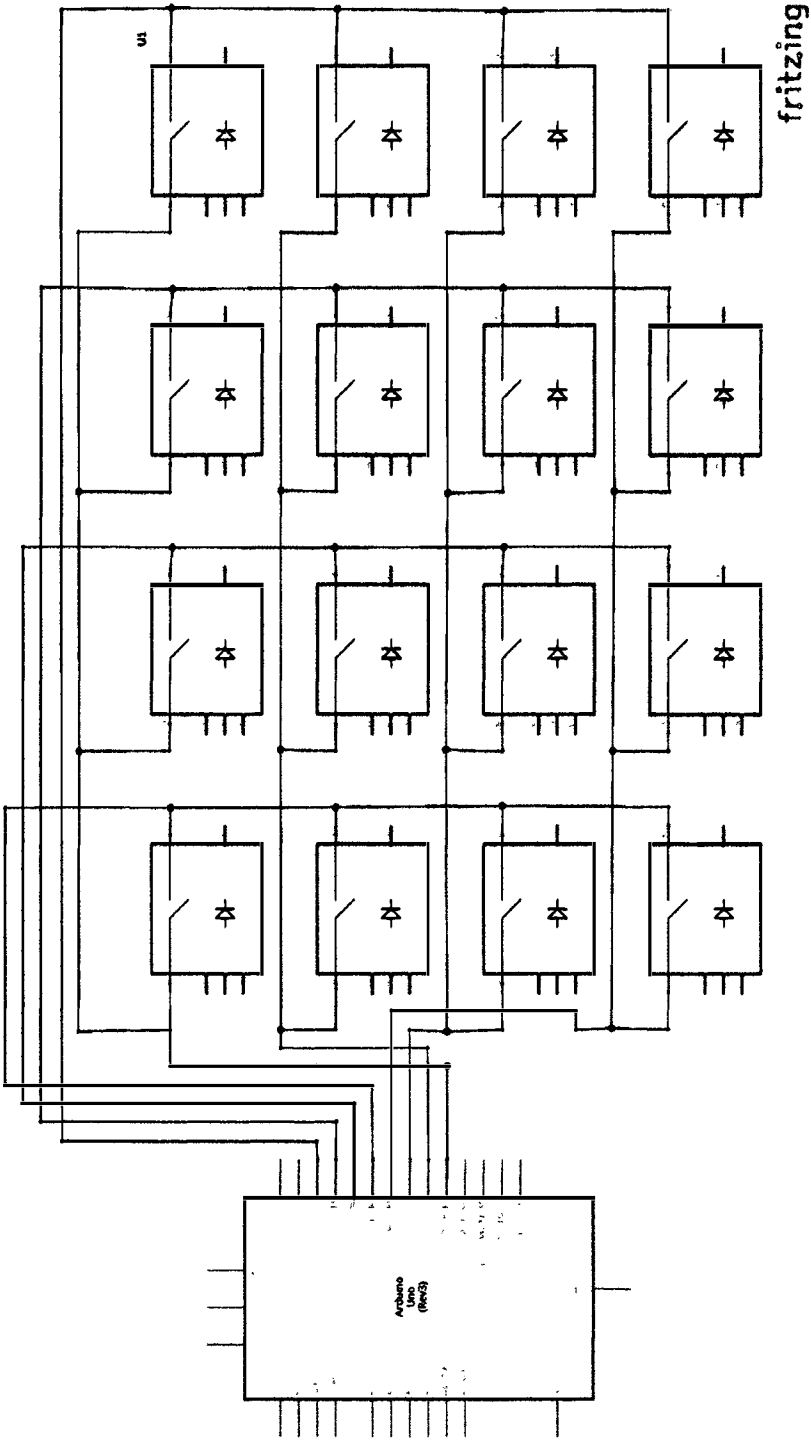


Рис. 87 ❖ Принципиальная схема подключения для практического занятия 23

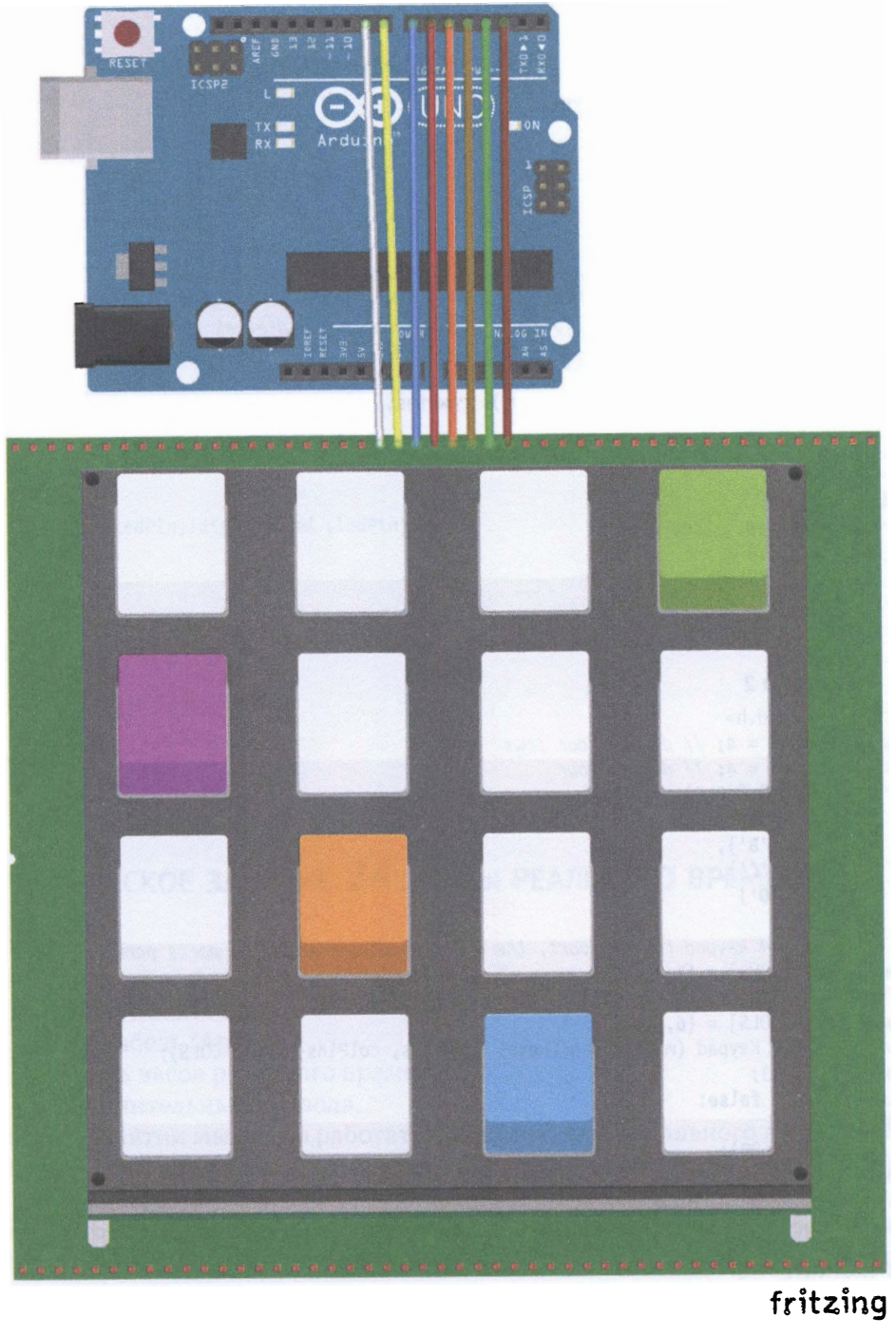


Рис. 88 ❖ Схема подключения с макетной платой для практического занятия 23

Код программы 1

```
#include <Keypad.h>
const byte ROWS = 4; // define four rows
const byte COLS = 4; // define four
char keys [ROWS] [COLS] = {
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};
// Connect 4 * 4 keypad row-bit port, the corresponding digital IO ports panel
byte rowPins [ROWS] = {2,3,4,5};
// Connect 4 * 4 buttons faithfully port, the corresponding digital IO ports panel
byte colPins [COLS] = {6,7,8,9};
// Call the function library function Keypad
Keypad keypad = Keypad (makeKeymap (keys), rowPins, colPins, ROWS, COLS);
void setup () {
Serial.begin (9600);
}
void loop () {
char key = keypad.getKey ();
if (key!= NO_KEY) {
Serial.println (key);
}
}
```

Код программы 2

```
#include <Keypad.h>
const byte ROWS = 4; // define four rows
const byte COLS = 4; // define four
char keys [ROWS] [COLS] = {
{'1', '2', '3','A'},
{'4', '5', '6','B'},
{'7', '8', '9','C'},
{'*', '0','#','D'}
};
// Connect 4 * 4 keypad row-bit port, the corresponding digital IO ports panel
byte rowPins [ROWS] = {2,3,4,5};
// Connect 4 * 4 buttons faithfully port, the corresponding digital IO ports panel
byte colPins [COLS] = {6,7,8,9};
Keypad keypad = Keypad (makeKeymap (keys), rowPins, colPins, ROWS, COLS);
byte ledPin = 13;
boolean blink = false;
void setup () {
Serial.begin (9600);
pinMode (ledPin, OUTPUT); // sets the digital pin as output
digitalWrite (ledPin, HIGH); // sets the LED on
keypad.addEventListener (keypadEvent); // add an event listener for this keypad
}
void loop () {
char key = keypad.getKey ();
if (key!= NO_KEY) {
Serial.println (key);
```

```
}
if (blink) {
digitalWrite (ledPin,! digitalRead (ledPin));
delay (100);
}
}
// Take care of some special events
void keypadEvent (KeypadEvent key) {
switch (keypad.getState ()) {
case PRESSED:
switch (key) {
case '#': digitalWrite (ledPin,!digitalRead (ledPin)); break;
case '*':
digitalWrite (ledPin,!digitalRead (ledPin));
break;
}
break;
case RELEASED:
switch (key) {
case '*':
digitalWrite (ledPin,!digitalRead (ledPin));
blink = false;
break;
}
break;
case HOLD:
switch (key) {
case '*': blink = true; break;
}
break;
}
}
```

Практическое занятие 24. Часы реального времени DS1307

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- модуль часов реального времени;
- соединительные провода.

В этом занятии мы будем работать с модулем часов реального времени (RTC module DS1307). Он поддерживает отображение года, месяца, дня, дня недели, часа, минуты и секунды. Нам понадобятся: консоль, или монитор порта (**Ctrl+Shift+M**), чтобы видеть время, выдаваемое модулем, и следующие библиотеки: stdio.h, string.h (они установлены по умолчанию) и DS1302.h, которую можно найти в [6]. Она вполне подходит для работы с модулем DS1307, хотя для него можно отдельно скачать библиотеку через **Скетч – Подключить библиотеку – Управлять библиотеками...** главного меню Adruino IDE и применить

код из примеров именно для него. Когда вы запустите программу, в консоль наверняка будет выводиться что-то типа Sunday 2000-1-1 1:11:17, т. е. часы немного отстали. Однако это можно исправить, отправив в текстовое поле корректные дату, время и день недели в следующем формате: год,месяц,день,час, минута,секунда,день недели, при этом день недели задаётся в виде числа, где воскресенье (Sunday) = 1, понедельник (Monday) = 2 и т. д. Например, можно ввести так: 2017,11,29,0,17,59,4 (откуда вы узнаете, когда я делал это задание). При этом после ввода вы увидите запись в консоли: You inputed:2017,11,29,0,17,59,4, после которой пойдёт отображение правильной даты и времени в соответствии с тем, что было введено: Wednesday 2017-11-29 0:17:59. Вы можете отключить на время модуль, а потом включить его снова и убедиться, что часы работают (потому что в них есть батарейка, рассчитанная на 9 лет работы). Такие же часы установлены на материнской плате любого компьютера. Обратите внимание на маркировку пинов на модуле с часов – она может отличаться от приведённой ниже схемы подключения!

Схема представлена на рис. 89–90.

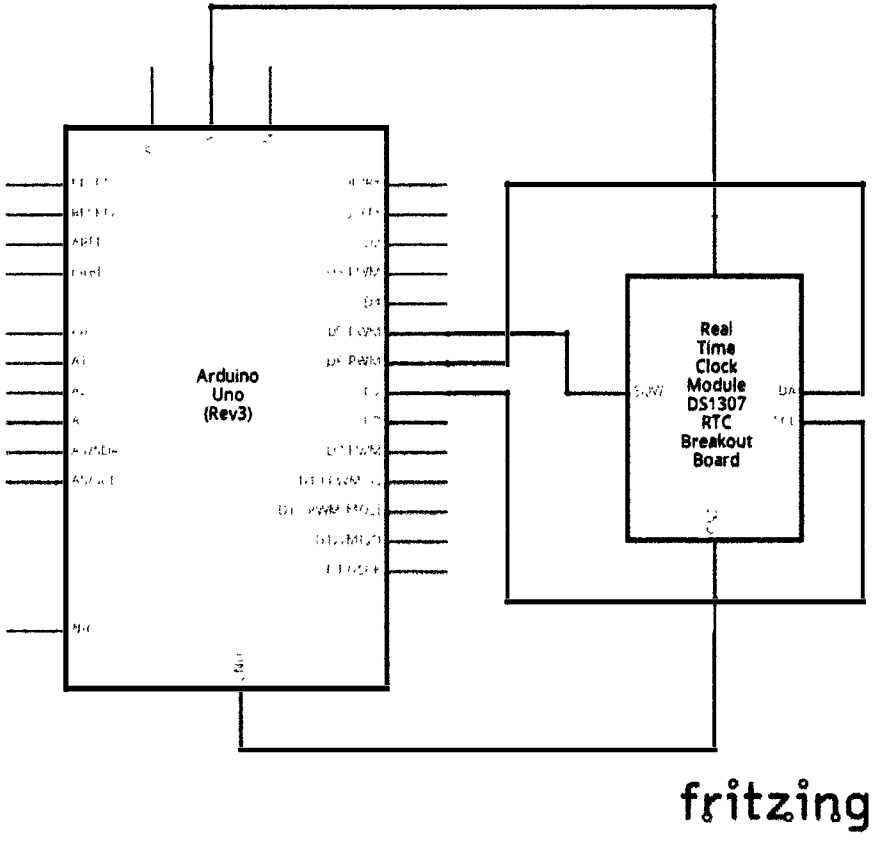


Рис. 89 ❖ Принципиальная схема подключения для практического занятия 24

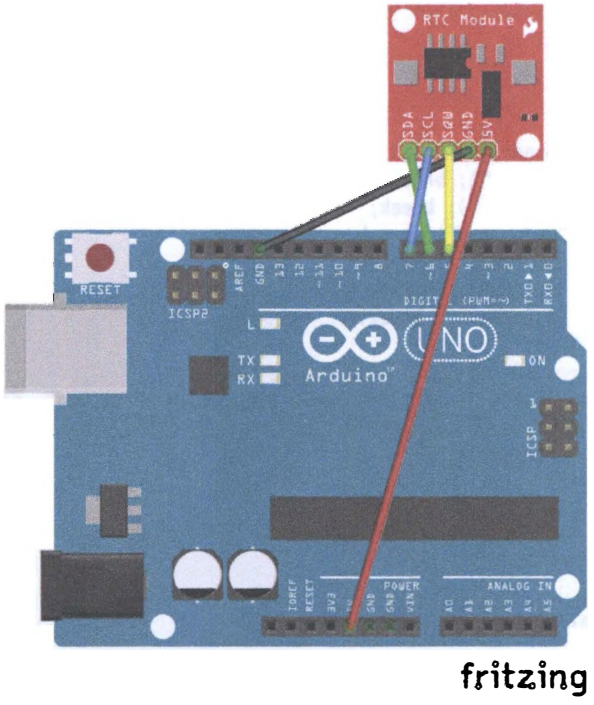


Рис. 90 ❖ Схема подключения для практического занятия 24

Код программы

```
#include <DS1302.h>
#include <stdio.h>
#include <string.h>

/* Interface Definition
CE (DS1302 pin5) -> Arduino D5
IO (DS1302 pin6) -> Arduino D6
SCLK (DS1302 pin7) -> Arduino D7
*/
uint8_t CE_PIN = 5;
uint8_t IO_PIN = 6;
uint8_t SCLK_PIN = 7;
/* Date variable buffer */
char buf [50];
char day [10];
/* Serial data cache */
String comdata = "";
int numdata [7] = {0}, j = 0, mark = 0;
/* Create DS1302 object */
DS1302 rtc (CE_PIN, IO_PIN, SCLK_PIN);
void print_time ()
{
/* Get the current time from the DS1302 */
```

```
Time t = rtc.time ();
/* The week from digital to name */
memset (day, 0, sizeof (day));
switch (t.day)
{
case 1: strcpy (day, "Sunday"); break;
case 2: strcpy (day, "Monday"); break;
case 3: strcpy (day, "Tuesday"); break;
case 4: strcpy (day, "Wednesday"); break;
case 5: strcpy (day, "Thursday"); break;
case 6: strcpy (day, "Friday"); break;
case 7: strcpy (day, "Saturday"); break;
}
/* The date code format to make up for output buf */
snprintf (buf, sizeof (buf), "% s% 04d-% 02d-% 02d% 02d:% 02d:% 02d", day, t.yr, t.mon,
t.date, t.hr, t. min, t.sec);
/* Output the date to the serial port */
Serial.println (buf);
}
void setup ()
{
Serial.begin (9600);
rtc.writeProtect (false);
rtc.halt (false);
}
void loop ()
{
/* When the serial port has data, the data is spliced into a variable comdata */
while(Serial.available()>0)
{
comdata+=char(Serial.read());
delay(2);
mark=1;
}
/* A comma-separated string comdata decomposition, the decomposition results become
converted into digital to numdata [] array */
if(mark==1)
{
Serial.print("You inputed:");
Serial.println(comdata);
for(int i=0;i<comdata.length();i++)
{
if(comdata[i]==','||comdata[i]==0x10||comdata[i]==0x13)
{
j++;
}
else
{
numdata[j]=numdata[j]*10+(comdata[i]-'0');
}
}
/* Make up the converted numdata time format, write DS1302 */
```

```
Time t(numdata[0],numdata[1],numdata[2],numdata[3],numdata[4],numdata[5],
numdata[6]);
rtc.time(t);
mark=0;j=0;
/* Empty comdata variable to wait for the next input */
comdata=String("");
/* Empty numdata */
for(int i=0;i<7;i++)numdata[i]=0;
}
/* Print the current time */
print_time();
delay(1000);
}
```

Практическое занятие 25. Эксперимент с датчиком уровня воды

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- датчик уровня воды;
- красный светодиод;
- резистор на 220 Ом;
- макетная плата;
- соединительные провода.

Датчик уровня воды (Water Sensor) определяет количество (точнее, уровень) воды, гибок в применении, высокочувствителен и потребляет мало энергии, а также поддерживается многими контроллерами и платами расширения/разработчика. Однако в библиотеке Fritzing его почему-то нет, поэтому в схемах пришлось воспользоваться датчиком влажности почвы (Soil Moisture Sensor) из набора элементов (parts) SparkFun-Sensors, который обладает такими же тремя выходами и, в принципе, решает те же задачи. В этом задании мы будем зажигать светодиод, подключённый к цифровому выходу 12 платы Arduino Uno, если уровень воды превышает определённое пороговое значение (550), выдаваемое на аналоговый порт A1 платы. Кроме того, нам понадобится ёмкость с водой, куда мы будем погружать (**только не до конца!**) датчик уровня воды. Не бойтесь погрузить датчик в воду на несколько секунд, бойтесь погрузить его не тем концом, или тем, но – по уровень надписи Water Sensor и глубже, или бойтесь оставить его в воде на долгое время. И ещё нам надо открыть консоль (монитор порта), чтобы видеть показания датчика. Не забудьте вытереть датчик после выполнения задания!

Схема представлена на рис. 91–92.

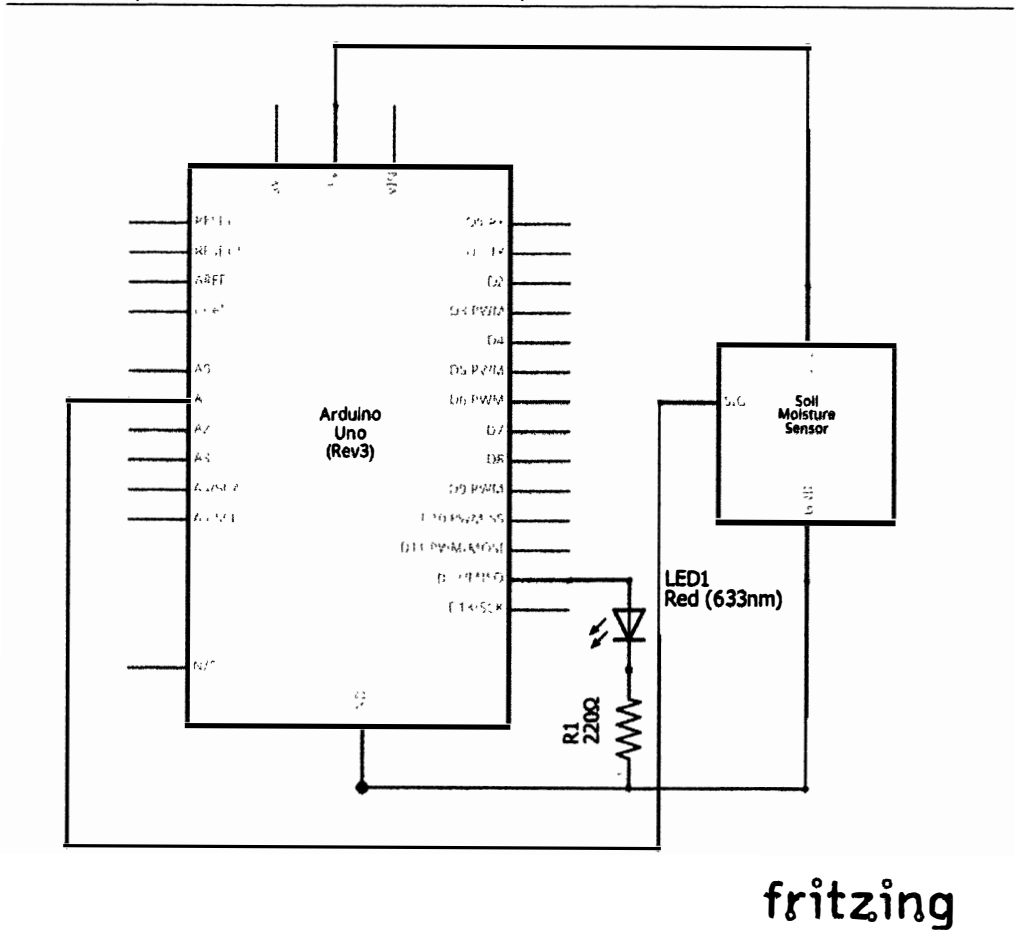


Рис. 91 ❖ Принципиальная схема подключения для практического занятия 25

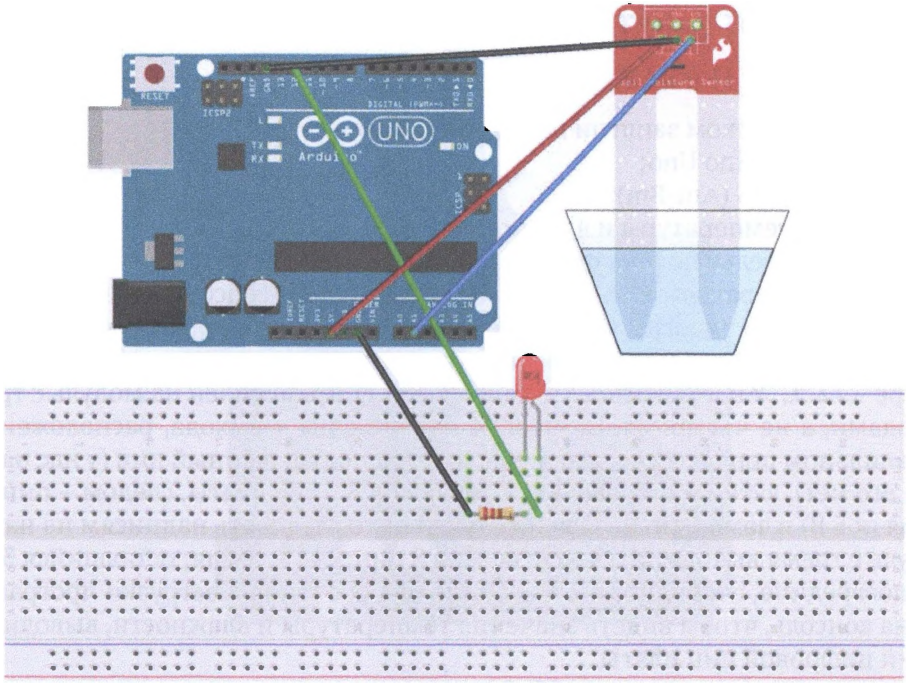


Рис. 92 ❖ Схема подключения с макетной платой для практического занятия 25

Код программы

```
int analogPin = 1; // level sensor connected to an analog port
int led = 12; // LED connected to digital port 12
int val = 0; // define the variable val initial value is 0
int data = 0; // define a variable data initial value is 0
void setup ()
{
pinMode (led, OUTPUT); // define led pin as an output
Serial.begin (9600); // set the baud rate to 9600
}
void loop ()
{
val = analogRead (analogPin); // read the analog value to give the variable val
if (val> 550) {// determine the variable val is greater than 700
digitalWrite (led, HIGH); // variable val is greater than 700, the light LED
}
else {
digitalWrite (led, LOW); // variable val is less than 700, the light goes off
}
data = val; // variable val assigned to the variable data
Serial.println (data); // serial print variable data
delay (100);
}
```

Практическое занятие 26. Эксперимент с сенсором температуры и влажности DHT11

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- сенсор температуры и влажности;
- соединительные провода.

Сенсор температуры и влажности гораздо меньше сенсора уровня воды. Он потребляет мало энергии и измеряет температуру в диапазоне от 0 до 50 °C, влажность – от 20 до 90 %. Точность измерений по влажности – ±5 %, по температуре – ±2 °C. У нас в комплекте сенсор DHT11 прикреплён на модуль с тремя выходами, а не четырьмя. На рис. 94 изображены 4 выхода, расположенных в правильном порядке: слева-направо идёт неподключённый пин (у нас на модуле его нет), затем – подключение сигнала к 8 пину платы, следом – пин питания (к 5 В) и земля. Но на всякий случай надо следовать надписям на нашем модуле с тремя выходами: (S – сигнал, на 8 пин; «–» – земля; оставшийся 3 выход посередине, очевидно, + (5 В)). После подключения и загрузки программы нужна консоль, чтобы видеть значения температуры и влажности, выводимые на 8-й цифровой пин платы.

Схема представлена на рис. 93–94.

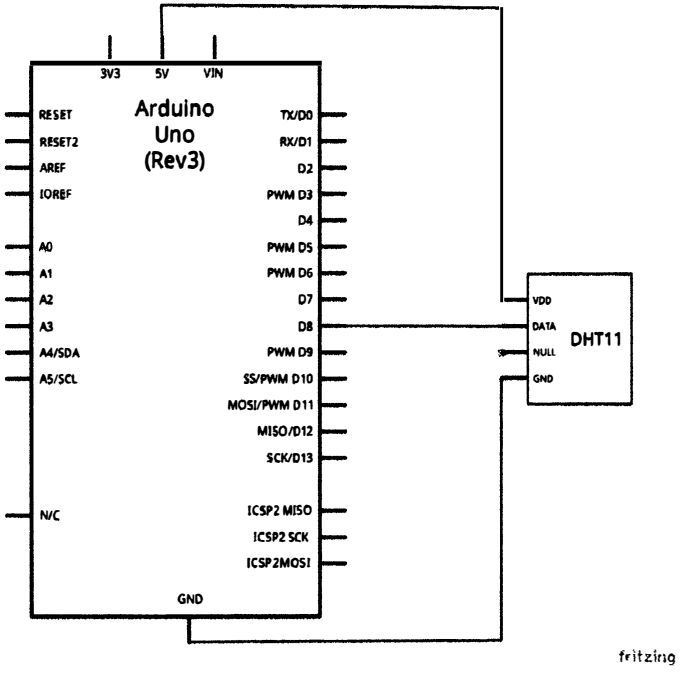


Рис. 93 ❖ Принципиальная схема подключения для практического занятия 26

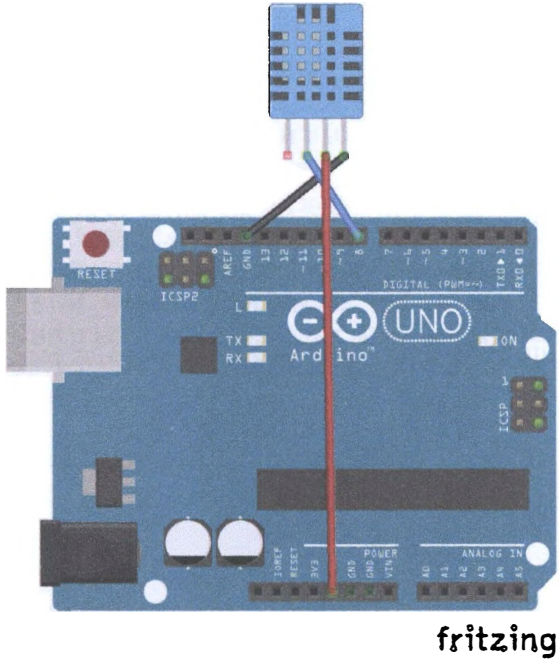


Рис. 94 ❖ Схема подключения для практического занятия 26

Код программы

```
int DHpin = 8;
byte dat [5];
byte read_data ()
{
byte data;
for (int i = 0; i <8; i++)
{
if (digitalRead (DHpin) == LOW)
{
while (digitalRead (DHpin) == LOW); // wait for 50us;
delayMicroseconds (30); // determine the duration of the high level to determine the data
is '0 'or
'1';
if (digitalRead (DHpin) == HIGH)
data |= (1 << (7-i)); // high front and low in the post;
while (digitalRead (DHpin) == HIGH); // data '1 ', wait for the next one receiver;
}
}
return data;
}
void start_test ()
{
```

```
digitalWrite (DHpin, LOW); // bus down, send start signal;
delay (30); // delay greater than 18ms, so DHT11 start signal can be detected;
digitalWrite (DHpin, HIGH);
delayMicroseconds (40); // Wait DHT11 response;
pinMode (DHpin, INPUT);
while (digitalRead (DHpin) == HIGH);
delayMicroseconds (80); // DHT11 a response, pulled the bus 80us;
if (digitalRead (DHpin) == LOW);
delayMicroseconds (80); // DHT11 80us after the bus pulled to start sending data;
for (int i = 0; i <4; i++) // receives temperature and humidity data, the parity bit is not
considered;
dat [i] = read_data ();
pinMode (DHpin, OUTPUT);
digitalWrite (DHpin, HIGH); // sending data once after releasing the bus, wait for the host
to open the next Start signal;
}
void setup ()
{
Serial.begin (9600);
pinMode (DHpin, OUTPUT);
}
void loop ()
{
start_test ();
Serial.print ("Current humdity =");
Serial.print (dat [0], DEC); // display the humidity-bit integer;
Serial.print ('.');
Serial.print (dat [1], DEC); // display the humidity decimal places;
Serial.println ('%');
Serial.print ("Current temperature =");
Serial.print (dat [2], DEC); // display the temperature of integer bits;
Serial.print ('.');
Serial.print (dat [3], DEC); // display the temperature of decimal places;
Serial.println ('C');
delay (700);
}
```

Практическое занятие 27. Эксперимент с релейным модулем

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- релейный модуль;
- синий светодиод;
- резистор на 220 Ом;
- макетная плата;
- соединительные провода.

Релейный модуль используется для переключения электрической схемы при возникновении определённого условия, подающегося на управляющий выход модуля, например определённого сигнала (0 или 1), значения и т. д. Применение такого модуля весьма широкое – от игрушек до систем сигнализации. У релейного модуля всего три выхода, помеченных соответственно +, – и S (управляющий/сигнальный выход). В этом занятии при срабатывании релейного модуля мы будем зажигать светодиод, подключённый не к плате, а к релейному модулю вместе с резистором на 220 Ом. Для этого понадобится отвёртка, чтобы открутить два винта двух контактов релейного модуля. Код программы крошечный: каждую секунду мы то зажигаем светодиод, посылая на 10-й цифровой пин платы 1, то гасим, посылая 0. При этом можно услышать соответствующий звук переключения релейного модуля – не пугайтесь.

Схема представлена на рис. 95–96.

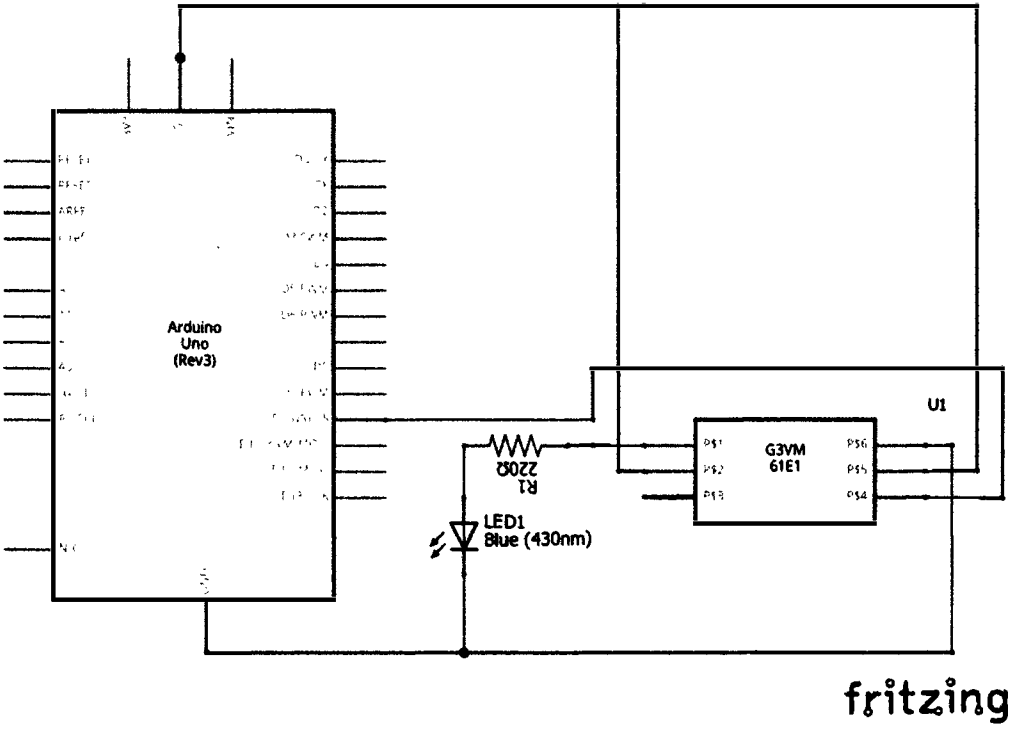


Рис. 95 ❖ Принципиальная схема подключения для практического занятия 27

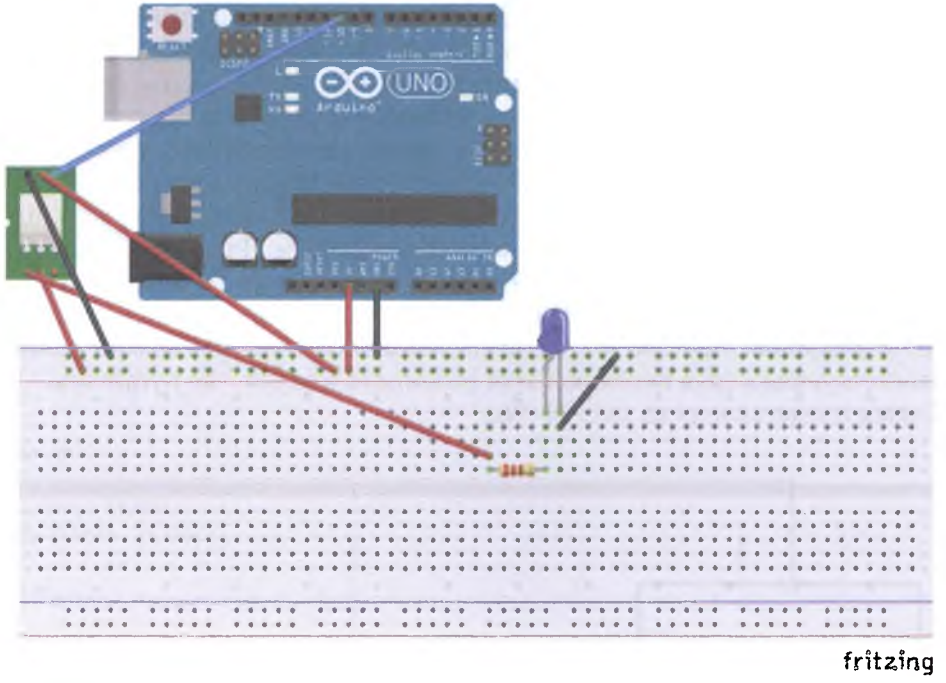


Рис. 96 ❖ Схема подключения с макетной платой для практического занятия 27

Код программы

```
int relay = 10; // relay turns trigger signal - active high;
void setup ()
{
pinMode (relay, OUTPUT); // Define port attribute is output;
}
void loop ()
{
digitalWrite (relay, HIGH); // relay conduction;
delay (1000);
digitalWrite (relay, LOW); // relay switch is turned off;
delay (1000);
}
```

Практическое занятие 28. Эксперимент с жидкокристаллическим монитором LCD1602A

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- жидкокристаллический монитор;
- потенциометр;
- 2 штырьковых коннектора по 8 пинов каждый;

- резистор на 220 Ом;
- макетная плата;
- соединительные провода.

Для работы с жидкокристаллическим монитором LCD1602A, у которого 16 выходов, нам понадобится библиотека LiquidCrystal.h, которая уже встроена в Arduino IDE. Этот мини-монитор может выводить 16 символов в каждой из 2 строк, работает от 5 В; с помощью потенциометра мы сможем регулировать яркость монитора. Фактически мы будем задействовать 12 пинов мини-монитора, поэтому предполагается, что к ним будут припаяны провода. Но можно обойтись и без этого с помощью макетной платы и штырьковых коннекторов двух типов – углового и прямого, каждый из которых содержит 8 пинов-выходов и вставляется через дырки-выходы LCD монитора в макетную плату, или сначала вставляется в макетную плату, а потом на него надеваются дырки вместе с монитором. С помощью представленного ниже кода выведем стандартную для подобных заданий фразу Hello, world! и начнём считать секунды. Обратите внимание, что контакт с монитором должен быть хороший всё время выполнения задания – сначала вы обеспечиваете хороший контакт с пинами монитора, потом подключаете плату к питанию и загружаете скетч. Можно обойтись без штырьковых коннекторов, если воткнуть провода через дырки пинов монитора в макетную плату, тем самым «пригвоздив» его к ней.

Схема представлена на рис. 97–98.

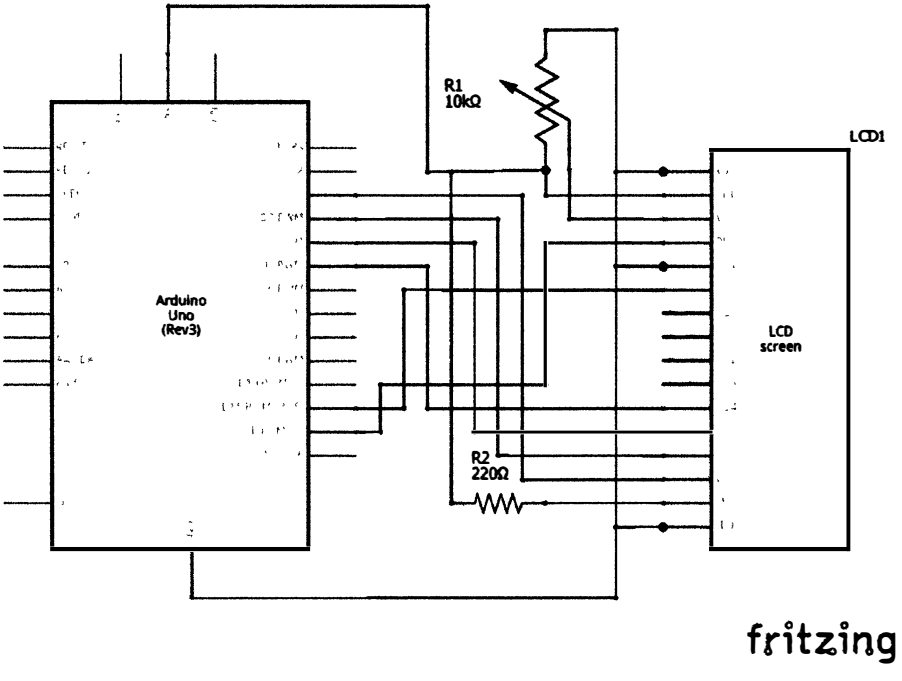


Рис. 97 ❖ Принципиальная схема подключения для практического занятия 28

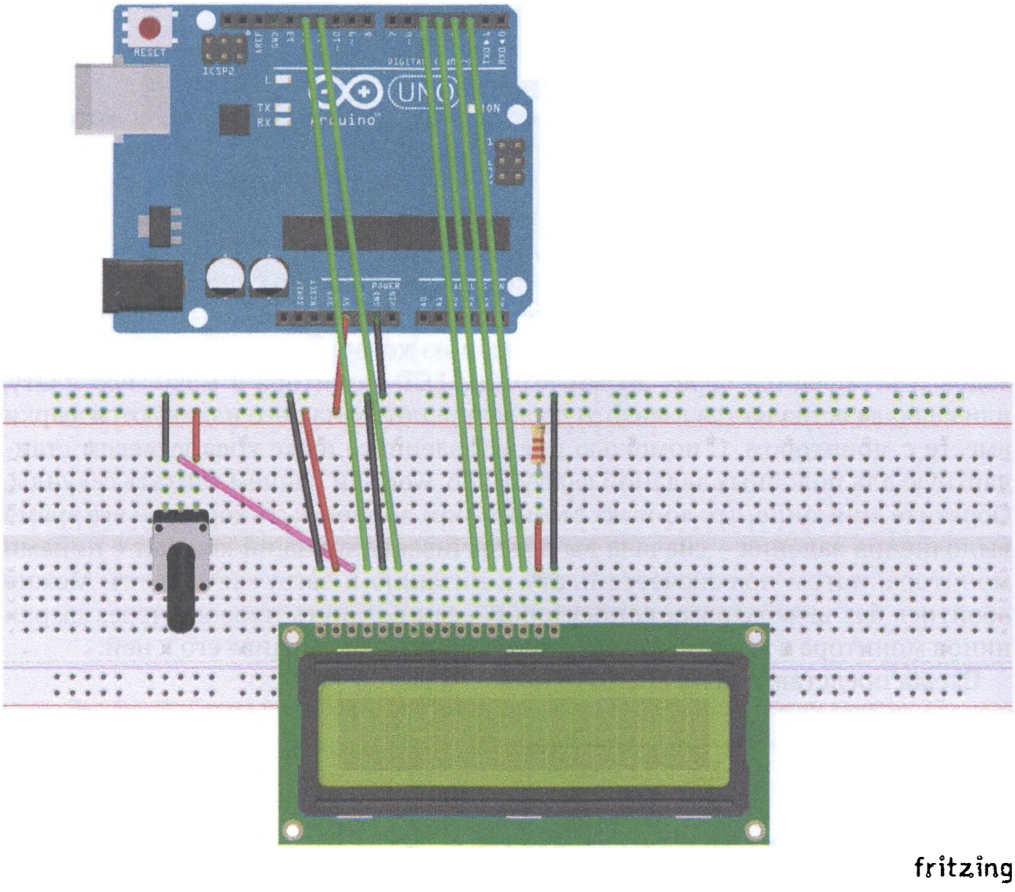


Рис. 98 ❖ Схема подключения с макетной платой для практического занятия 28

Код программы

```
// include the library code:
#include <LiquidCrystal.h>
// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
// set up the LCD's number of columns and rows:
lcd.begin(16, 2);
// Print a message to the LCD.
lcd.print("hello, world!");
}

void loop() {
// set the cursor to column 0, line 1
// (note: line 1 is the second row, since counting begins with 0):
```

```
lcd.setCursor(0, 1);
// print the number of seconds since reset:
lcd.print(millis() / 1000);
}
```

Практическое занятие 29. Эксперимент с шаговым двигателем

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- шаговый двигатель;
- модуль для шагового двигателя;
- потенциометр;
- макетная плата;
- соединительные провода.

Шаговый двигатель при получении импульсного сигнала поворачивает силовой привод (и рукоятку, прикреплённую к нему, или вообще любую деталь – actuator) на определённый угол (шаговый угол). Направление поворота, скорость, ускорение можно контролировать с помощью изменения частоты импульса или количества импульсов. Мы будем использовать четырёхфазный шаговый двигатель диаметром 28 мм с 5 выходами, который работает от 5 В и может поворачивать привод на угол 5,625*(1...64), т. е. на 5,625 градуса, 11,25 градуса и т. д. до 360 (т. е. вращаться до бесконечности). Также нам понадобится модуль ULN2003APG для шагового двигателя (подключается к земле и питанию 5 В соответствующими пинами – и +, а также к выходам 8–11 платы Arduino Uno с помощью пинов IN1–IN4), потенциометр для контроля скорости вращения привода (подключён к пину A0 платы) и библиотека Stepper.h, уже установленная в Arduino IDE. На схемах изображён немного другой шаговый двигатель, т. к. нашего в библиотеке Fritzing не нашлось.

Схема представлена на рис. 99–100.

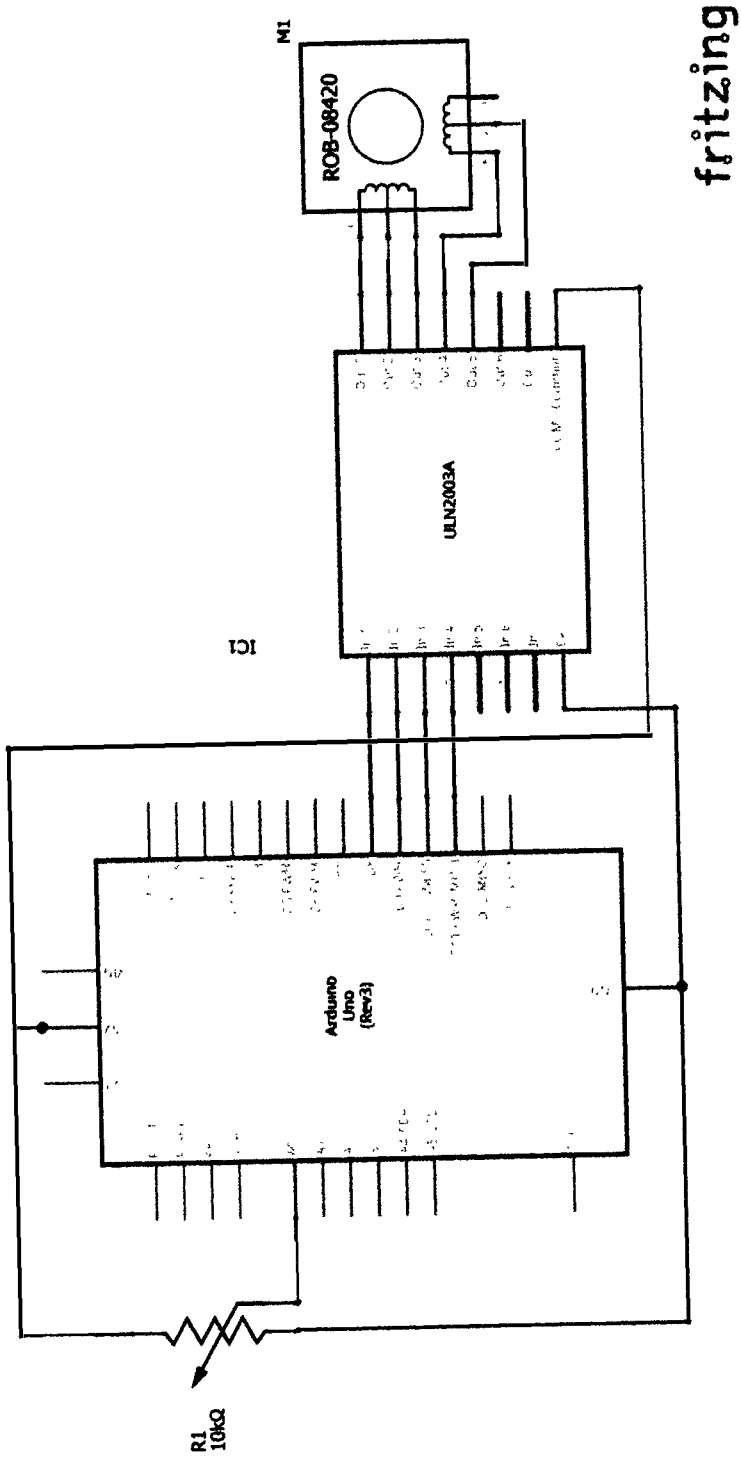


Рис. 99 ❖ Принципиальная схема подключения для практического занятия 29

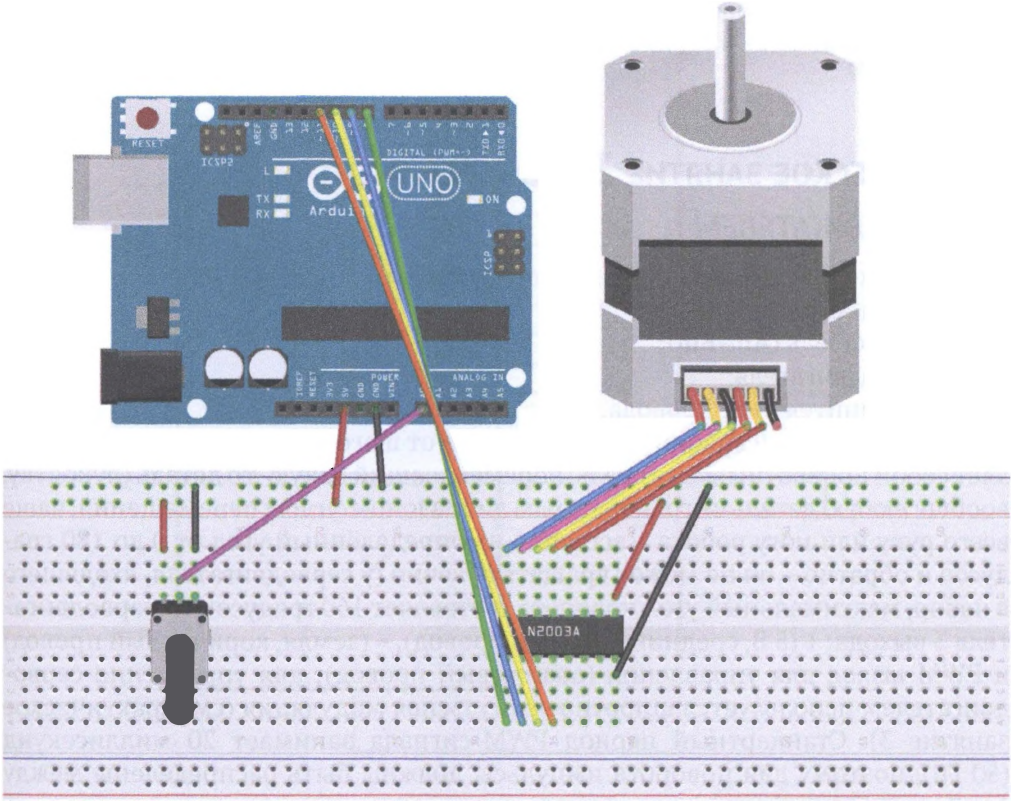


Рис. 100 ❖ Схема подключения с макетной платой для практического занятия 29

Код программы

```
#include <Stepper.h>
// Set here is the number of revolution of the stepper motor step
#define STEPS 100
// Attached to set the number of steps of the stepper motor and pin
Stepper stepper (STEPS, 8, 9, 10, 11);
// Define a variable to store the history of reading
int previous = 0;
void setup ()
{
// Set the motor speed of 90 per minute step
stepper.setSpeed (90);
}
void loop ()
{
// Get the sensor readings
int val = analogRead (0);
// Move to the current reading minus the number of steps historical readings
```

```
stepper.step (val - previous);
// Save history readings
previous = val;
}
```

Практическое занятие 30. Эксперимент с серводвигателем

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- серводвигатель;
- соединительные провода.

Серводвигатель (сервопривод), в отличие от шагового двигателя, представляет собой поворотный двигатель, перемещающий какую-то деталь (рукоятку, вообще любую деталь с ограниченным диапазоном/углом перемещения, чаще всего руку или ногу робота – actuator) на определённый угол от 0 до 180 градусов и обратно – он не может вращаться вечно (у серводвигателя, входящего в набор, максимальный угол поворота составляет 160 градусов). У серводвигателя 3 выхода: + (5 В, средний красный провод), – (земля, коричневый провод) и PWM-выход для управления (оранжевый провод); для управления серводвигателем используется широтно-импульсная модуляция (см. практическое занятие 3). Стандартный период PWM-сигнала занимает 20 миллисекунд (50 Гц), поэтому для поворота импульсы должны быть распределены между 1 и 2 миллисекундами, но на практике эти значения могут варьироваться от 0.5 до 2.5 миллисекунды для поворотов на углы от 0 до 180 градусов. Для поворота на 0 градусов (т. е. никакого поворота) на соответствующий PWM-пин платы подаётся сигнал 1000 (1000 микросекунд = 1 миллисекунда), на 45 градусов – 1250, на 180 градусов – 2000 микросекунд. В реальности эти значения могут варьироваться от 500 до 2480 микросекунд. Чтобы увидеть поворот, можно надеть самую большую насадку из маленького пакетика с 3 насадками и 3 шурупами на белую шестерню двигателя. Напишем программу, в которой через консоль будем вводить значения угла, на который мы хотим повернуть пластмассовую насадку, надетую на двигатель. Если вводить цифры от 1 до 9, то можно поворачивать насадку, соответственно, на 20 градусов, 40, 60 и т. д. (9 = 180 градусов; работать не должно, т. к. у серводвигателя, входящего в набор, максимальный угол поворота составляет 160 градусов). Потом напишем другую программу, использующую встроенную библиотеку Servo.h для управления серводвигателем.

Схема представлена на рис. 101–102.

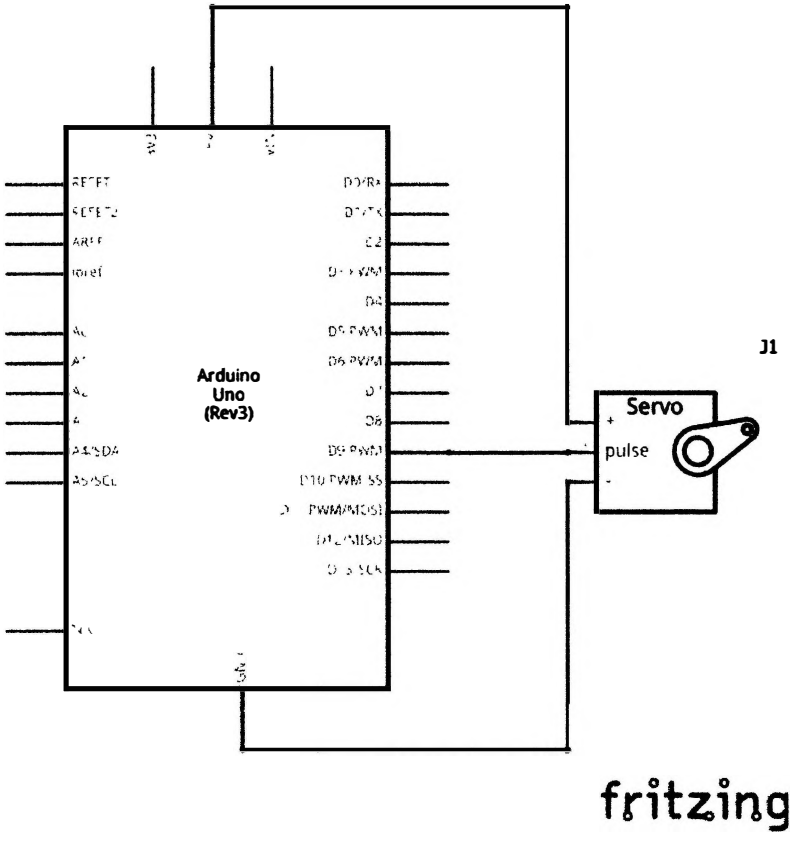


Рис. 101 ❖ Принципиальная схема подключения для практического занятия 30

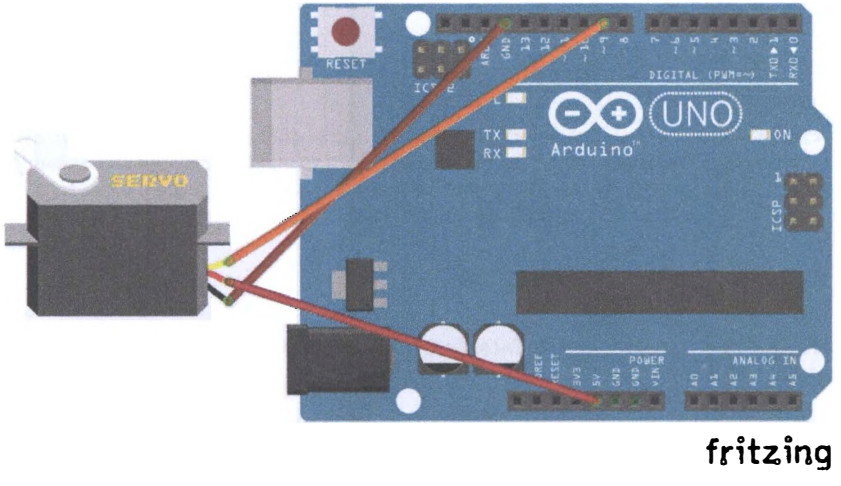


Рис. 102 ❖ Схема подключения для практического занятия 30

Код программы 1

```
int servopin = 9 ;// define the digital interface to connect the servo servo signal line 9
int myangle ;// define the angle variables
int pulsewidth ;// define variable pulse width
int val;
void servopulse (int servopin, int myangle) // define a pulse function
{
    pulsewidth = (myangle * 11) +500 ;// the angle value into a pulse width 500-2480
    digitalWrite (servopin, HIGH) ;// the servo interface level to high
    delayMicroseconds (pulsewidth) ;// number of microseconds delay pulse width value
    digitalWrite (servopin, LOW) ;// the servo interface level to Low
    delay (20-pulsewidth/1000);
}
void setup ()
{
    pinMode (servopin, OUTPUT) ;// set servo interface output interface
    Serial.begin (9600) ;// connect to the serial port, the baud rate is 9600
    Serial.println ("servo = o_seral_simple ready");
}
void loop () // will be 0-9's 0-180 number into perspective, and let the number of times
the  corresponding LED flashes
{
    val = Serial.read () ;// read the value of the serial port
    if (val> '0'&& val <= '9')
    {
        val = val-'0'; // will be converted to numeric variables characteristic quantities
        val = val * (180/9) ;// will figure into perspective
        Serial.print ("moving servo to");
        Serial.print (val, DEC);
        Serial.println ();
        for (int i = 0; i <= 50; i++) // give enough time to let it go to the steering
angle specified
        {
            servopulse (servopin, val) ;// reference pulse function
        }
    }
}
```

Код программы 2

```
#include <Servo.h> // define header files, there is one thing to note, you can directly
click on themenu bar at the Arduino software Sketch> Importlibrary> Servo, Servo function
call, you can also directly enter the # include <Servo.h>, But at the input to the
attention of the # include and <Servo.h> should be a space between, otherwise a compile-
time error.
Servo myservo ;// define a variable name servos
void setup ()
{
myservo.attach (9) ;// define Servo Interface (9,10 all OK, shortcomings can only control
2)
}
void loop ()
{
myservo.write (90) ;// set the steering angle of rotation
}
```

Практическое занятие 31. Эксперимент с игровым джойстиком

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- джойстик;
- соединительные провода.

Игровой джойстик позволяет контролировать значения по осям *x* и *y* и нажатия кнопки; при этом значения по *x* и по *y* отсылаются на аналоговые порты платы (например, A0 и A1 соответственно), а нажатия кнопки – на цифровой порт (например, 7). У джойстика 5 выходов, но они подписаны, так что проблем возникнуть не должно, если учесть, что SW – это определение нажатия кнопки (подаём на 7-й цифровой пин платы). Также нам понадобится консоль (монитор порта, **Ctrl+Shift+M**), чтобы видеть выводимые значения при выполнении программы 1. Вторая программа просто немного по-другому делает то же самое.

Схема представлена на рис. 103–104.

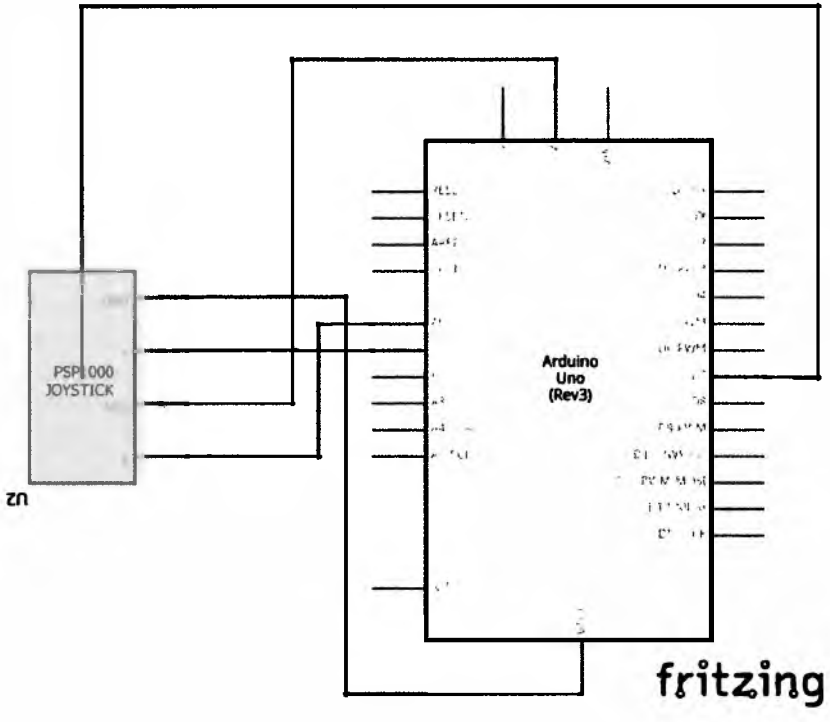


Рис. 103 ❖ Принципиальная схема подключения для практического занятия 31

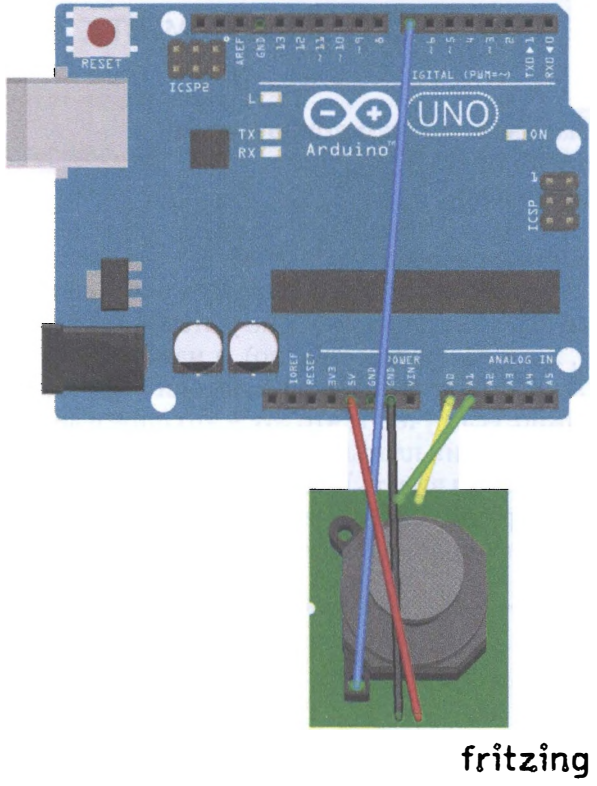


Рис. 104 ❖ Схема подключения для практического занятия 31

Код программы 1

```
int value = 0;
void setup () {
    pinMode (7, INPUT);
    Serial.begin (9600);
}
void loop () {
    value = analogRead (0);
    Serial.print ("X:");
    Serial.print (value, DEC);
    value = analogRead (1);
    Serial.print (" | Y:");
    Serial.print (value, DEC);
    value = digitalRead (7);
    Serial.print (" | Z:");
    Serial.println (value, DEC);
    delay (100);
}
```

Код программы 2

```
int JoyStick_X = 0; // x
int JoyStick_Y = 1; // y
int JoyStick_Z = 7; // key
void setup ()
{
    pinMode (JoyStick_X, INPUT);
    pinMode (JoyStick_Y, INPUT);
    pinMode (JoyStick_Z, INPUT);
    Serial.begin (9600); // 9600 bps
}
void loop ()
{
    int x, y, z;
    x = analogRead (JoyStick_X);
    y = analogRead (JoyStick_Y);
    z = digitalRead (JoyStick_Z);
    Serial.print (x, DEC);
    Serial.print (",");
    Serial.print (y, DEC);
    Serial.print (",");
    Serial.println (z, DEC);
    delay (100);
}
```

Практическое занятие 32. Эксперимент с инфракрасным пультом дистанционного управления

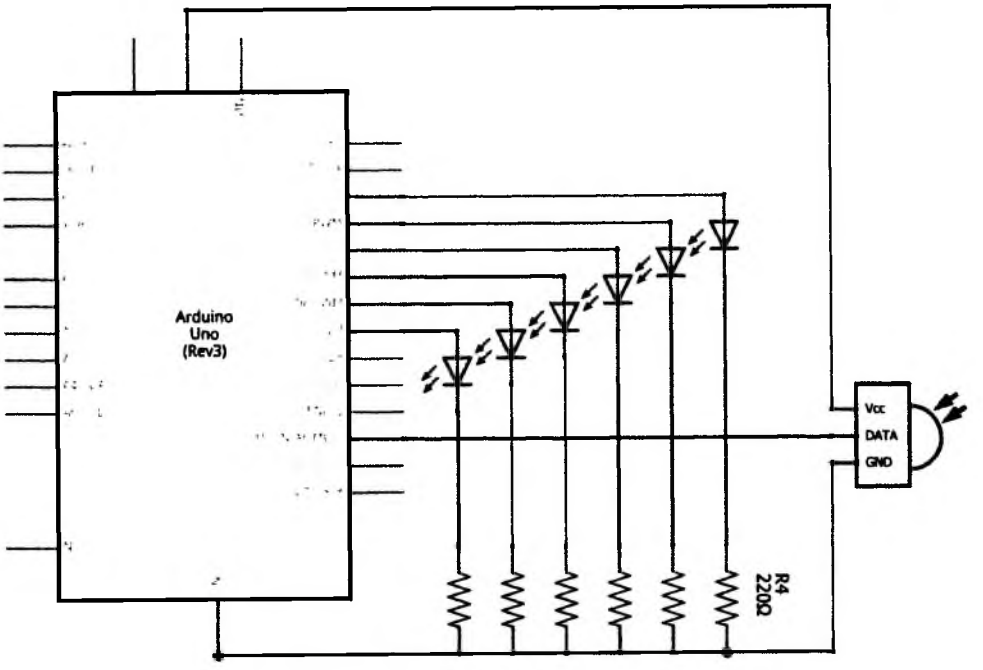
В этом практическом занятии нам **понадобятся**:

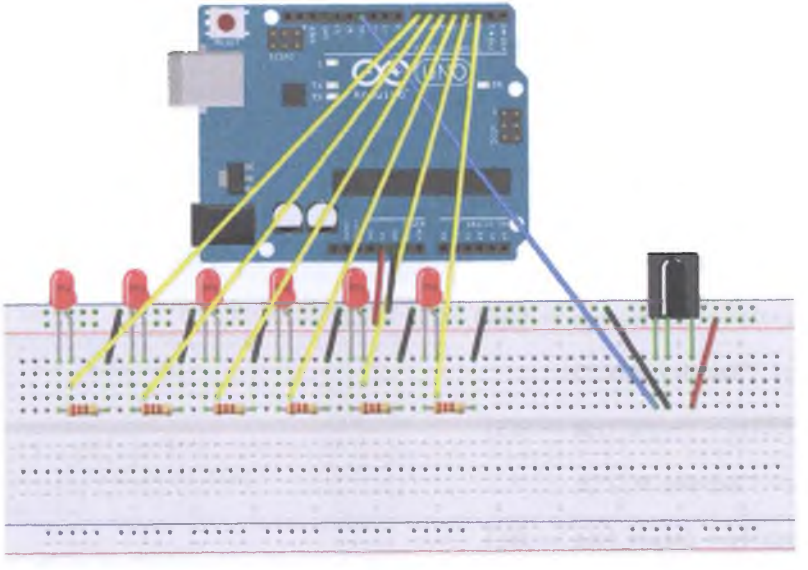
- плата Arduino Uno;
- USB-кабель (Am-Bm);
- инфракрасный пульт дистанционного управления;
- инфракрасный приёмник;
- 6 светодиодов;
- 6 резисторов на 220 Ом;
- макетная плата;
- соединительные провода.

Сигналы, поступающие с инфракрасного пульта дистанционного управления (ИК-пульта ДУ), представляют собой двоичный импульсный код. Чтобы передать без проводов сигнал в ИК-диапазоне, сначала двоичный код проходит модуляцию на определённой частоте, и затем результат высылается ИК-светодиодами в направлении ИК-приёмника, который демодулирует полученный сигнал и превращает его обратно в двоичный код. Оптический сигнал, передаваемый в ИК-диапазоне ИК-светодиодами, преобразуется в слабый электрический сигнал, который затем усиливается, пропускается через фильтр, демодулятор и, таким образом, восстанавливается в исходный сигнал

в двоичном коде. ИК-приёмник сигнала располагает 3 пинами: + (5 В), – (земля) и VOUT для приёма аналогового оптического сигнала. Протокол передачи данных, по которому исходный сигнал кодируется ИК-пультом ДУ и затем декодируется ИК-приёмником сигнала, – это NEC-протокол: он использует 8 адресных битов и 8 битов для команды; адресные и командные биты пересылаются дважды, чтобы убедиться в их правильности; используется импульсно-позиционная модуляция; частота сигнала равна 38 КГц; логическая единица длится 2,25 миллисекунды, в которых первые 0,56 миллисекунды уровень сигнала 1, остальные – 0; логический ноль длится 1,12 миллисекунды, где опять первые 0,56 миллисекунды уровень сигнала равен 1, а остальные – 0. При подаче сигнала первые 9 миллисекунд сигнал равен 1, а затем следует 0 уровень, длящийся 4,5 миллисекунды. Такая комбинация означает, что дальше будет передаваться, собственно, основная информация (адрес + адрес + команда + команда для каждого сигнала). Если нажать на кнопку пульта и не отпускать её, вся подготовка и пересылка команды занимает 110 миллисекунд, которые затем повторяются каждые 110 миллисекунд с помощью повторяющего импульса (9 миллисекунд подаём 1, и 2,25 миллисекунды – 0), если кнопка остаётся нажатой. Для выполнения этого задания нам понадобится библиотека IRemote.h – придётся её установить из [7], т. к. встроенная библиотека немного другая – для роботов; также нам понадобится консоль (монитор порта). Программа, которую мы напишем, будет зажигать светодиоды и выводить в консоль информацию о том, что сигнал распознан (это будет NEC-сигнал, но можно попробовать и другой ИК-пульт ДУ: у меня получилось распознать сигнал с пульта от телевизора SONY). Светодиоды будут зажигаться в соответствии с нажатыми на пульте кнопками: первые 6 кнопок сверху (двух верхних рядов) зажигают светодиоды, вторые 6 кнопок сверху (ряд 3 и 4) гасят светодиоды. Прежде чем нажимать на кнопки пульта, не забудьте вынуть блокирующую контакты батарейки полоску из него! И вставить её обратно после завершения выполнения задания. А теперь самое интересное (для тех, кто дочитал до конца): у ИК-приёмника 3 выхода, и подключать их надо следующим образом: если ИК-приёмник повёрнут к вам «лицом» (на котором «маска» крест-накрест), то справа от вас находится пин питания, слева от вас – пин OUT, который подаётся к 11-му цифровому пину платы, а посередине у ИК-приёмника – земля.

Схема представлена на рис. 105–106.


Arduino
Uno
(Rev3)
Vcc
DATA
GND
R4
220Ω



Код программы

```
#include <IRremote.h>
int RECV_PIN = 11;
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
int LED6 = 7;
long on1 = 0x00FFA25D;
long off1 = 0x00FFE01F;
long on2 = 0x00FF629D;
long off2 = 0x00FFA857;
long on3 = 0x00FFE21D;
long off3 = 0x00FF906F;
long on4 = 0x00FF22DD;
long off4 = 0x00FF6897;
long on5 = 0x00FF02FD;
long off5 = 0x00FF9867;
long on6 = 0x00FFC23D;
long off6 = 0x00FFB04F;
IRrecv irrecv (RECV_PIN);
decode_results results;
// Dumps out the decode_results structure.
// Call this after IRrecv :: decode ()
// Void * to work around compiler issue
// Void dump (void * v) {
// Decode_results * results = (decode_results *) v
void dump (decode_results * results) {
int count = results-> rawlen;
if (results-> decode_type == UNKNOWN)
{
Serial.println ("Could not decode message");
}
else
{
if (results-> decode_type == NEC)
{
Serial.print ("Decoded NEC:");
}
else if (results-> decode_type == SONY)
{
Serial.print ("Decoded SONY:");
}
else if (results-> decode_type == RC5)
{
Serial.print ("Decoded RC5:");
}
else if (results-> decode_type == RC6)
{
Serial.print ("Decoded RC6:");
}
```

```
Serial.print (results-> value, HEX);
Serial.print ("(");
Serial.print (results-> bits, DEC);
Serial.println ("bits)");
}
Serial.print ("Raw (");
Serial.print (count, DEC);
Serial.print ("):");
for (int i = 0; i <count; i++)
{
if ((i%2) == 1) {
Serial.print (results-> rawbuf [i] * USECPERTICK, DEC);
}
else
{
Serial.print (- (int) results-> rawbuf [i] * USECPERTICK, DEC);
}
Serial.print ("");
}
Serial.println ("");
}
void setup ()
{
pinMode (RECV_PIN, INPUT);
pinMode (LED1, OUTPUT);
pinMode (LED2, OUTPUT);
pinMode (LED3, OUTPUT);
pinMode (LED4, OUTPUT);
pinMode (LED5, OUTPUT);
pinMode (LED6, OUTPUT);
pinMode (13, OUTPUT);
Serial.begin (9600);
irrecv.enableIRIn (); // Start the receiver
}
int on = 0;
unsigned long last = millis ();
void loop ()
{
if (irrecv.decode (& results))
{
// If it's been at least 1/4 second since the last
// IR received, toggle the relay
if (millis () - last> 250)
{
on =! on;
// DigitalWrite (8, on? HIGH: LOW);
digitalWrite (13, on? HIGH: LOW);
dump (& results);
}
if (results.value == on1)
digitalWrite (LED1, HIGH);
if (results.value == off1)
```

```
digitalWrite (LED1, LOW);
if (results.value == on2)
digitalWrite (LED2, HIGH);
if (results.value == off2)
digitalWrite (LED2, LOW);
if (results.value == on3)
digitalWrite (LED3, HIGH);
if (results.value == off3)
digitalWrite (LED3, LOW);
if (results.value == on4)
digitalWrite (LED4, HIGH);
if (results.value == off4)
digitalWrite (LED4, LOW);
if (results.value == on5)
digitalWrite (LED5, HIGH);
if (results.value == off5)
digitalWrite (LED5, LOW);
if (results.value == on6)
digitalWrite (LED6, HIGH);
if (results.value == off6)
digitalWrite (LED6, LOW);
last = millis ();
irrecv.resume (); // Receive the next value
}
}
```

Практическое занятие 33. Эксперимент с RFID-модулем RC522

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- RFID-модуль;
- RFID-ключ;
- RFID-карта;
- штырьковый коннектор на 8 пинов;
- макетная плата;
- соединительные провода.

RFID – Radio Frequency IDentification, т. е. идентификация по радиочастоте. Принцип действия схож с ИК-пультом ДУ и ИК-приёмником, только здесь используется радиосигнал, а не оптический сигнал. Тоже есть источник сигнала (у нас это будет карточка, также он называется тег, метка, носитель сигнала, транспондер) и приёмник – RFID-модуль (считыватель, ридер, сканер). **Для RFID-модуля надо использовать питание 3.3 В, иначе он сгорит.** Также в этом занятии нам понадобятся RFID-ключ и RFID-карта – это 2 отдельных носителя сигнала, или метки, а ещё – библиотека SPI.h, которая уже встроена в Arduino IDE, библиотека MFRC522.h, которую можно скачать и установить из

[8], и консоль, в которой мы будем, поднося к RFID-модулю сначала карточку, потом ключ, считывать данные, записанные на них. Карту надо удерживать у модуля секунд 10, чтобы прочитались все данные! На схемах ниже представлен другой элемент – RFID USB Reader из набора SparkFun библиотеки Fritzing, но главное, что у него 8 выходов, и последовательность подключения сохранена (лучше смотреть цветную схему с макетной платой). У нашего RFID-модуля RC522 тоже 8 выходов, и подключаются они следующим образом: 3,3V – 3,3V, RST – D9 (или просто 9 – здесь имеются в виду только цифровые выходы платы), GND – GND, MISO – 12, MOSI – 11, SCK – 13, SDA – 10. Пропускаем мы только выход прерываний IRQ – он не подключён ни к чему. Обратите внимание, что контакт с RFID-модулем должен быть хороший всё время выполнения задания – сначала вы обеспечиваете хороший контакт с пинами модуля, потом подключаете плату к питанию и загружаете скетч. Можно обойтись без штырьковых коннекторов, если воткнуть провода через дырки пинов RFID-модуля в макетную плату.

Схема представлена на рис. 107–108.

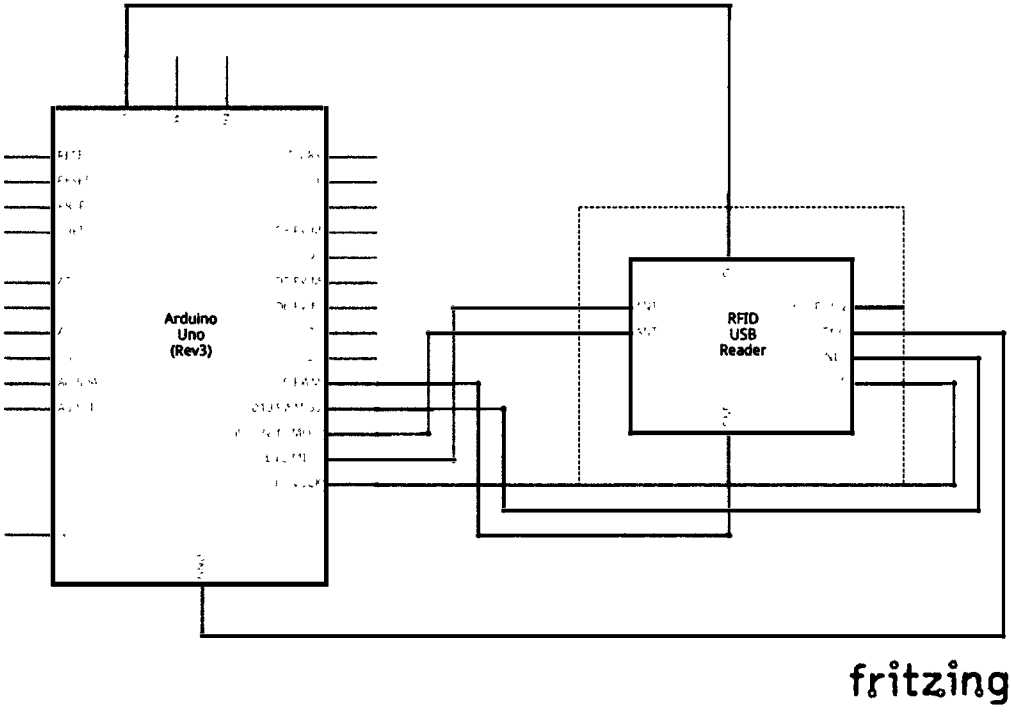


Рис. 107 ❖ Принципиальная схема подключения для практического занятия 33

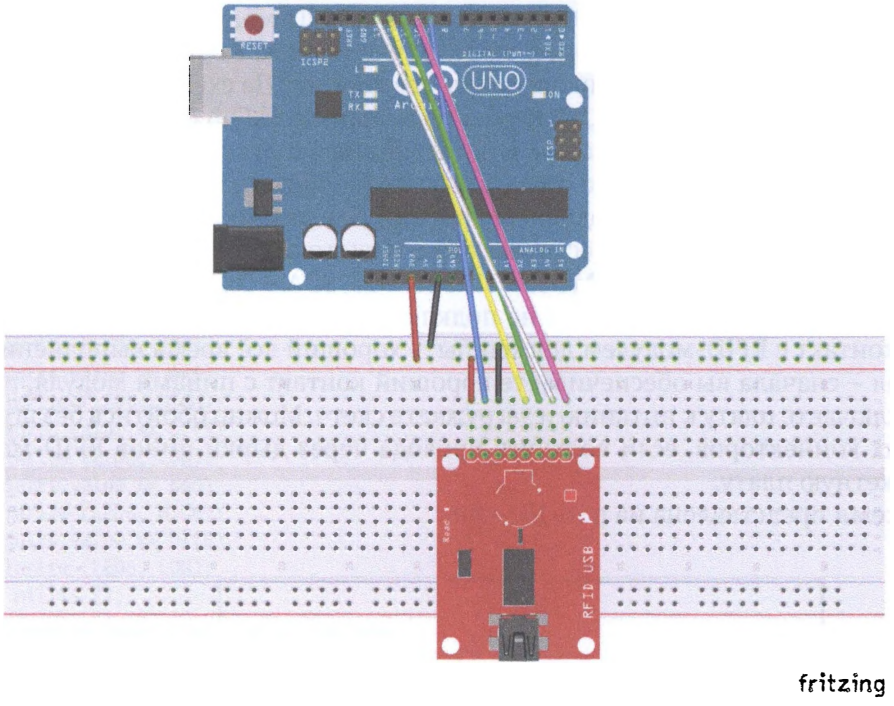


Рис. 108 ❖ Схема подключения с макетной платой для практического занятия 33

Код программы

```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         9          //
#define SS_PIN          10         //

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance

void setup() {
Serial.begin(9600);                // Initialize serial communications with the PC
while (!Serial);                   // Do nothing if no serial port is opened (added for Arduinos based
on ATMEGA32U4)
SPI.begin();                       // Init SPI bus
mfrc522.PCD_Init();                // Init MFRC522
ShowReaderDetails();               // Show details of PCD - MFRC522 Card Reader details
Serial.println(F("Scan PICC to see UID, type, and data blocks..."));
}

void loop() {
// Look for new cards
if ( ! mfrc522.PICC_IsNewCardPresent()) {
return;
}

// Select one of the cards
if ( ! mfrc522.PICC_ReadCardSerial()) {
```

```
return;
}

// Dump debug info about the card; PICC_HaltA() is automatically called
mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}

void ShowReaderDetails() {
// Get the MFRC522 software version
byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
Serial.print(F("MFRC522 Software Version: 0x"));
Serial.print(v, HEX);
if (v == 0x91)
Serial.print(F(" = v1.0"));
else if (v == 0x92)
Serial.print(F(" = v2.0"));
else
Serial.print(F(" (unknown)"));
Serial.println("");
// When 0x00 or 0xFF is returned, communication probably failed
if ((v == 0x00) || (v == 0xFF)) {
Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?"));
}
}
```

Практическое занятие 34. Эксперимент с системой контроля доступа

В этом практическом занятии нам **понадобятся**:

- плата Arduino Uno;
- USB-кабель (Am-Bm);
- RFID-модуль;
- RFID-ключ;
- RFID-карта;
- штырьковый коннектор на 8 пинов;
- релейный модуль;
- 2 светодиода – красный и синий (зелёный);
- 2 резистора на 220 Ом;
- макетная плата;
- соединительные провода.

В этом занятии нам понадобится опыт предыдущего занятия и 27-го занятия, где мы работали с релейным модулем. Суть задачи в том, чтобы собрать схему с RFID-модулем и релейным модулем, в которой будут 2 светодиода – красный и синий (или зелёный, но зелёных у нас в наборе нет), и затем написать программу, которая при поднесении RFID-карты с правильным записанным на неё паролем к RFID-модулю будет переключать релейный модуль и включать синий светодиод, сигнализируя о том, что доступ открыт (access granted), в противном случае (неправильный пароль, записанный на RFID-ключ) будет гореть красный светодиод (доступ закрыт, access denied). **Не забудьте, что**

RFID-модуль подключается к 3.3 В, а релейный модуль – к 5 В! Сначала мы должны запустить код в программе 1 (SetPassword), открыть консоль (монитор порта, **Ctrl+Shift+M**) и поднести RFID-карту к RFID-модулю, чтобы изменить пароль; при этом мы увидим сообщение «The password has been changed!» среди бесчисленных сообщений «Error!» в консоли. RFID-ключ мы не подносим, т. е. пароль на нём не меняется на правильный. Таким образом, на карте у нас записан правильный пароль, на ключе – неправильный. Затем мы запускаем код программы 2 (DoorCon) и подносим RFID-карту – релейный модуль переключается, и на 5 секунд должен загореться синий светодиод, сигнализирующий о том, что на карте пароль правильный; при этом красный светодиод гаснет на те же 5 секунд. Затем подносим RFID-ключ и видим, что ничего не происходит.

Схема представлена на рис. 109–110.

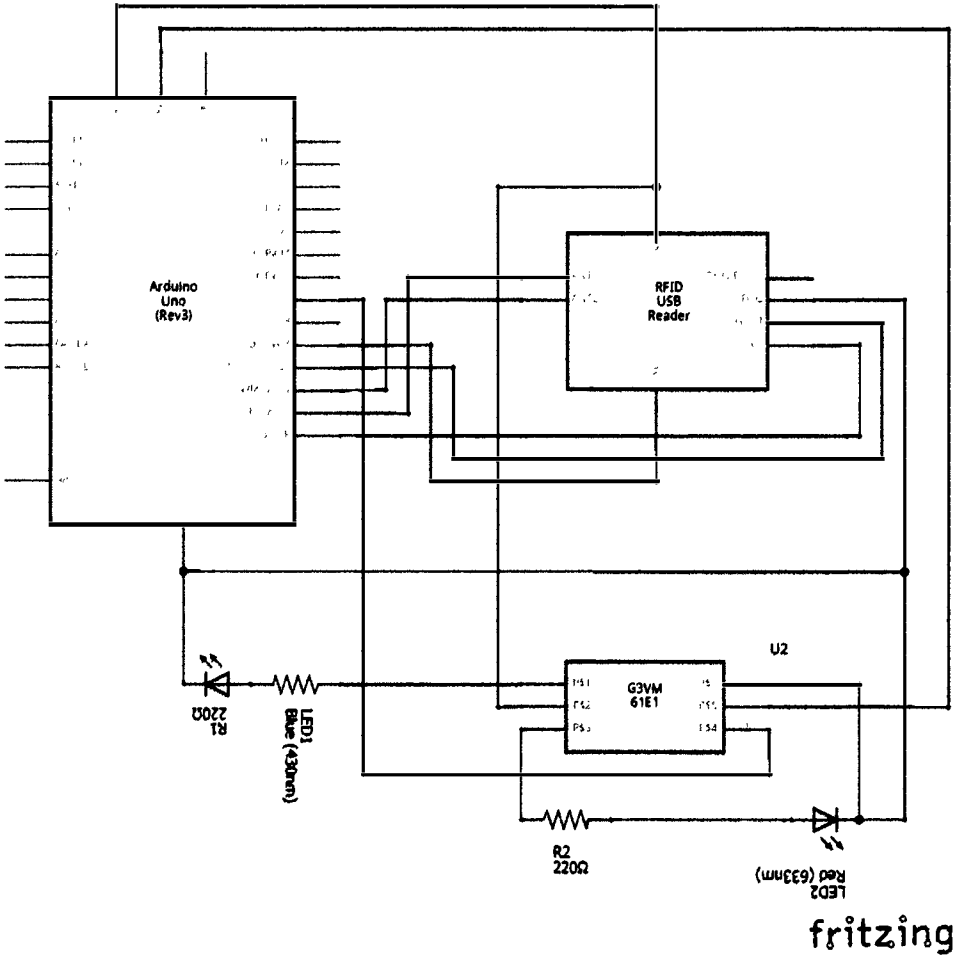


Рис. 109 ❖ Принципиальная схема подключения для практического занятия 34

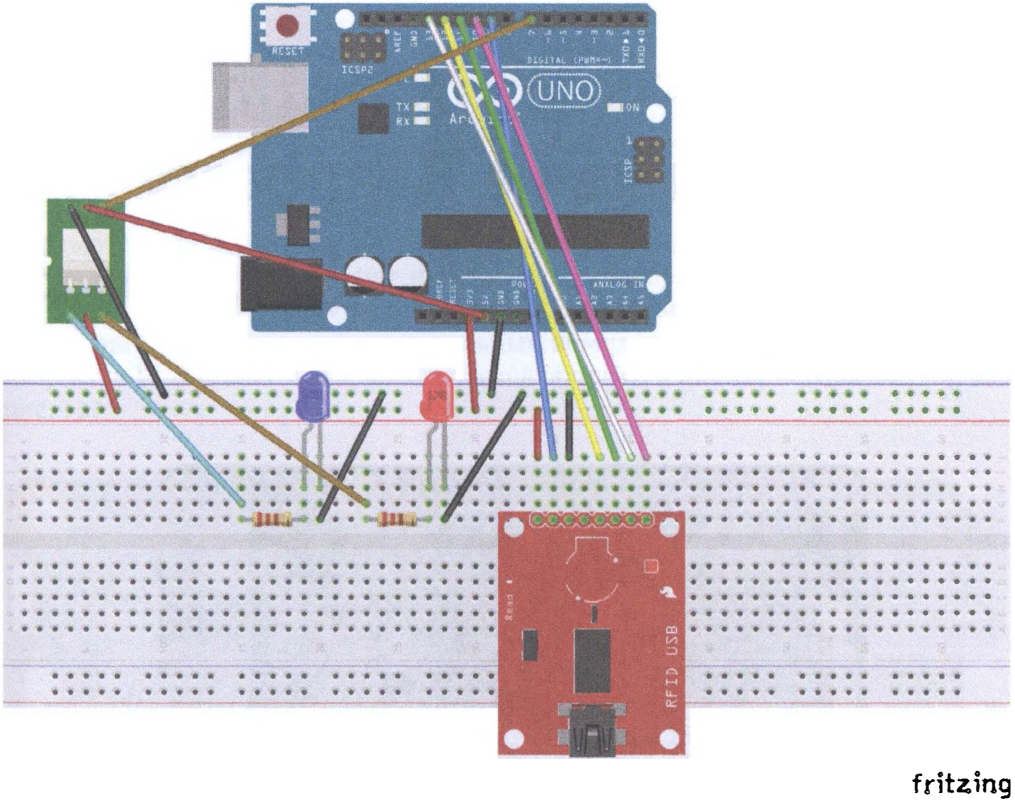


Рис. 110 ❖ Схема подключения с макетной платой для практического занятия 34

Код программы 1 (SetPassword) и **код программы 2 (DoorCon)** загружены по адресам [13] и [14] соответственно ввиду их большого объёма и большого количества ошибок при их копировании в Arduino IDE.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ПРОМЫШЛЕННЫЙ ИНТЕРНЕТ-ВЕЩЕЙ

Методические указания к самостоятельным работам

Направление подготовки 09.03.02 Информационные системы и технологии
Направленность (профиль) «Цифровые технологии химических производств» Квалификация выпускника – бакалавр

(Электронный документ)

Невинномысск 2025

Методические указания предназначены для студентов направления подготовки 09.03.02 Информационные системы и технологии и других технических специальностей. Они содержат рекомендации по органи-зации самостоятельных работ студента для дисциплины «Управление информационными проектами и ресурсами».

Методические указания разработаны в соответствии с требованиями ФГОС ВО в части содержания и уровня подготовки выпускников направления 09.03.02 Информационные системы и технологии

Содержание

1 Подготовка к лекциям..... 4
2 Подготовка к практическим занятиям 6
4 Самостоятельное изучение темы. Конспект..... 8

1 Подготовка к лекциям

Главное в период подготовки к лекционным занятиям – научиться методам самостоятельного умственного труда, сознательно развивать свои творческие способности и овладевать навыками творческой работы. Для этого необходимо строго соблюдать дисциплину учебы и поведения. Четкое планирование своего рабочего времени и отдыха является необходимым условием для успешной самостоятельной работы. В основу его нужно положить рабочие программы изучаемых в семестре дисциплин.

Каждому студенту следует составлять еженедельный и семестровый планы работы, а также план на каждый рабочий день. С вечера всегда надо распределять работу на завтрашний день. В конце каждого дня целесообразно подводить итог работы: тщательно проверить, все ли выполнено по намеченному плану, не было ли каких-либо отступлений, а если были, по какой причине это произошло. Нужно осуществлять самоконтроль, который является необходимым условием успешной учебы. Если что-то осталось невыполненным, необходимо изыскать время для завершения этой части работы, не уменьшая объема недельного плана.

Слушание и запись лекций – сложный вид вузовской аудиторной работы. Внимательное слушание и конспектирование лекций предполагает интенсивную умственную деятельность студента. Краткие записи лекций, их конспектирование помогает усвоить учебный материал. Конспект является полезным тогда, когда записано самое существенное, основное и сделано это самим студентом. Не надо стремиться записать дословно всю лекцию. Такое «конспектирование» приносит больше вреда, чем пользы. Запись лекций рекомендуется вести по возможности собственными формулировками. Желательно запись осуществлять на одной странице, а следующую оставлять для проработки учебного материала самостоятельно в домашних условиях.

Конспект лекций лучше подразделять на пункты, параграфы, соблюдая красную строку. Этому в большой степени будут способствовать пункты плана лекции, предложенные преподавателям. Принципиальные места,

определения, формулы и другое следует сопровождать замечаниями «важно», «особо важно», «хорошо запомнить» и т.п. Можно делать это и с помощью разноцветных маркеров или ручек. Лучше если они будут собственными, чтобы не приходилось присить их у однокурсников и тем самым не отвлекать их во время лекции. Целесообразно разработать собственную «маркографию» (значки, символы), сокращения слов. Не лишним будет и изучение основ стенографии. Работая над конспектом лекций, всегда необходимо использовать не только учебник, но и ту литературу, которую дополнительно рекомендовал лектор. Именно такая серьезная, кропотливая работа с лекционным материалом позволит глубоко овладеть знаниями.

2 Подготовка к практическим занятиям

Подготовку к каждому практическому занятию студент должен начать с ознакомления с методическими указаниями, которые включают содержание работы. Тщательное продумывание и изучение вопросов основывается на проработке текущего материала лекции, а затем изучения обязательной и дополнительной литературы, рекомендованную к данной теме. На основе индивидуальных предпочтений студенту необходимо самостоятельно выбрать тему доклада по проблеме и по возможности подготовить по нему презентацию.

Если программой дисциплины предусмотрено выполнение практического задания, то его необходимо выполнить с учетом предложенной инструкции (устно или письменно). Все новые понятия по изучаемой теме необходимо выучить наизусть и внести в глоссарий, который целесообразно вести с самого начала изучения курса. Результат такой работы должен проявиться в способности студента свободно ответить на теоретические вопросы семинара, его выступлении и участии в коллективном обсуждении вопросов изучаемой темы, правильном выполнении практических заданий и контрольных работ.

В зависимости от содержания и количества отведенного времени на изучение каждой темы практическое занятие может состоять из четырех-пяти частей:

1. Обсуждение теоретических вопросов, определенных программой дисциплины.

2. Доклад и/ или выступление с презентациями по выбранной проблеме.

3. Обсуждение выступлений по теме – дискуссия.

4. Выполнение практического задания с последующим разбором полученных результатов или обсуждение практического задания.

5. Подведение итогов занятия.

Первая часть – обсуждение теоретических вопросов – проводится в виде фронтальной беседы со всей группой и включает выборочную проверку преподавателем теоретических знаний студентов. Примерная продолжительность

— до 15 минут. Вторая часть — выступление студентов с докладами, которые должны сопровождаться презентациями с целью усиления наглядности восприятия, по одному из вопросов практического занятия. Обязательный элемент доклада – представление и анализ статистических данных, обоснование социальных последствий любого экономического факта, явления или процесса. Примерная продолжительность — 20-25 минут. После докладов следует их обсуждение – дискуссия. В ходе этого этапа практического занятия могут быть заданы уточняющие вопросы к докладчикам. Примерная продолжительность – до 15-20 минут. Если программой предусмотрено выполнение практического задания в рамках конкретной темы, то преподавателями определяется его содержание и дается время на его выполнение, а замет идет обсуждение результатов. Подведением итогов заканчивается практическое занятие.

В процессе подготовки к практическим занятиям, студентам необходимо обратить особое внимание на самостоятельное изучение рекомендованной учебно-методической (а также научной и популярной) литературы. Самостоятельная работа с учебниками, учебными пособиями, научной, справочной и популярной литературой, материалами периодических изданий и Интернета, статистическими данными является наиболее эффективным методом получения знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала, формирует у студентов свое отношение к конкретной проблеме. Более глубокому раскрытию вопросов способствует знакомство с дополнительной литературой, рекомендованной преподавателем по каждой теме семинарского или практического занятия, что позволяет студентам проявить свою индивидуальность в рамках выступления на данных занятиях, выявить широкий спектр мнений по изучаемой проблеме.

3 Самостоятельное изучение темы. Конспект

Конспект – наиболее совершенная и наиболее сложная форма записи. Слово «конспект» происходит от латинского «conspectus», что означает «обзор, изложение». В правильно составленном конспекте обычно выделено самое основное в изучаемом тексте, сосредоточено внимание на наиболее существенном, в кратких и четких формулировках обобщены важные теоретические положения.

Конспект представляет собой относительно подробное, последовательное изложение содержания прочитанного. На первых порах целесообразно в записях ближе держаться тексту, прибегая зачастую к прямому цитированию автора. В дальнейшем, по мере выработки навыков конспектирования, записи будут носить более свободный и сжатый характер.

Конспект книги обычно ведется в тетради. В самом начале конспекта указывается фамилия автора, полное название произведения, издательство, год и место издания. При цитировании обязательная ссылка на страницу книги. Если цитата взята из собрания сочинений, то необходимо указать соответствующий том. Следует помнить, что четкая ссылка на источник – непременное правило конспектирования. Если конспектируется статья, то указывается, где и когда она была напечатана.

Конспект подразделяется на части в соответсвтии с заранее продуманным планом. Пункты плана записываются в тексте или на полях конспекта. Писать его рекомендуется четко и разборчиво, так как небрежная запись с течением времени становиться малопонятной для ее автора. Существует правило: конспект, составленный для себя, должен быть по возможности написан так, чтобы его легко прочитал и кто-либо другой.

Формы конспекта могут быть разными и зависят от его целевого назначения (изучение материала в целом или под определенным углом зрения, подготовка к докладу, выступлению на занятии и т.д.), а также от характера произведения (монография, статья, документ и т.п.). Если речь идет просто об

изложении содержания работы, текст конспекта может быть сплошным, с выделением особо важных положений подчеркиванием или различными значками.

В случае, когда не ограничиваются переложением содержания, а фиксируют в конспекте и свои собственные суждения по данному вопросу или дополняют конспект соответствующими материалами их других источников, следует отводить место для такого рода записей. Рекомендуется разделить страницы тетради пополам по вертикали и в левой части вести конспект произведения, а в правой свои дополнительные записи, совмещая их по содержанию.

Конспектирование в большей мере, чем другие виды записей, помогает вырабатывать навыки правильного изложения в письменной форме важные теоретических и практических вопросов, умение четко их формулировать и ясно излагать своими словами.

Таким образом, составление конспекта требует вдумчивой работы, затраты времени и труда. Зато во время конспектирования приобретаются знания, создается фонд записей.

Конспект может быть текстуальным или тематическим. В текстуальном конспекте сохраняется логика и структура изучаемого произведения, а запись ведется в соответствии с расположением материала в книге. За основу тематического конспекта берется не план произведения, а содержание какой-либо темы или проблемы.

Текстуальный конспект желательно начинать после того, как вся книга прочитана и продумана, но это, к сожалению, не всегда возможно. В первую очередь необходимо составить план произведения письменно или мысленно, поскольку в соответствии с этим планом строится дальнейшая работа. Конспект включает в себя тезисы, которые составляют его основу. Но, в отличие от тезисов, конспект содержит краткую запись не только выводов, но и доказательств, вплоть до фактического материала. Иначе говоря, конспект – это

расширенные тезисы, дополненные рассуждениями и доказательствами, мыслями и соображениями составителя записи.

Как правило, конспект включает в себя и выписки, но в него могут войти отдельные места, цитируемые дословно, а также факты, примеры, цифры, таблицы и схемы, взятые из книги. Следует помнить, что работа над конспектом только тогда будет творческой, когда она не ограничена текстом изучаемого произведения. Нужно дополнять конспект данными из другими источников.

В конспекте необходимо выделять отдельные места текста в зависимости от их значимости. Можно пользоваться различными способами: подчеркиваниями, вопросительными и восклицательными знаками, репликами, краткими оценками, писать на полях своих конспектов слова: «важно», «очень важно», «верно», «характерно».

В конспект могут помещаться диаграммы, схемы, таблицы, которые придадут ему наглядность.

Составлению тематического конспекта предшествует тщательное изучение всей литературы, подобранной для раскрытия данной темы. Бывает, что какая-либо тема рассматривается в нескольких главах или в разных местах книги. А в конспекте весь материал, относящийся к теме, будет сосредоточен в одном месте. В плане конспекта рекомендуется делать пометки, к каким источникам (вплоть до страницы) придется обратиться для раскрытия вопросов. Тематический конспект составляется обычно для того, чтобы глубже изучить определенный вопрос, подготовиться к докладу, лекции или выступлению на семинарском занятии. Такой конспект по содержанию приближается к реферату, докладу по избранной теме, особенно если включает и собственный вклад в изучение проблемы.