Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Оренбургский государственный университет»

Е.С. Шелихов, Э.Л. Греков, А.С. Безгин

ПРИМЕНЕНИЕ ПРОГРАММНО-АППАРАТНЫХ СРЕДСТВ ARDUINO ПРИ РАЗРАБОТКЕ АВТОМАТИЗИРОВАННЫХ СИСТЕМ СВЕТОВОЙ ИНДИКАЦИИ И ВЫВОДА ИНФОРМАЦИИ

Часть I

Учебное пособие

Рекомендовано ученым советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по образовательным программам высшего образования по направлениям подготовки 13.03.02 и 13.04.02 Электроэнергетика и электротехника

Оренбург
2019

УДК 004.4(075.8)
ББК 32.971.32я73
Ш42

Рецензент – доцент кандидат технических наук А.М. Черноусова

Шелихов, Е. С.
Ш42 Применение программно-аппаратных средств Arduino при разработке автоматизированных систем световой индикации и вывода информации В 2 ч. Ч. 1: учебное пособие / Е.С. Шелихов, Э.Л. Греков, А.С. Безгин; Оренбургский гос. ун-т. – Оренбург: ОГУ, 2019. – 127 с.
ISBN

В учебном пособии рассматриваются вопросы разработки и проектирования систем световой индикации и вывода информации на базе микроконтроллеров ATmega. Приведены краткие теоретические сведения для обучающихся, начинающих работать с микропроцессорной техникой, в частности с аппаратной и программной средой Arduino. Описано подробное руководство по применению программных продуктов Fritzing и MS Visio при разработке проектов. Представлены работы, которые могут быть реализованы на практических или лабораторных занятиях по дисциплинам «Микропроцессорная техника в исследовании и управлении энергетических комплексов» и «Компьютерное управление электромеханическими устройствами», а также выполнены самостоятельно. Для самоподготовки и контроля теоретических знаний приводятся контрольные вопросы.

Учебное пособие предназначено для обучающихся по образовательным программам высшего образования по направлениям подготовки 13.03.02 и 13.04.02 Электроэнергетика и электротехника. Может быть полезно для инженеров, магистрантов и аспирантов технических направлений подготовки.

УДК 004.4(075.8)
ББК 32.971.32я73

ISBN

Содержание

Введение

Учебное пособие предназначено для выполнения практических, лабораторных и самостоятельных работ по дисциплинам «Компьютерное управление электромеханическими устройствами» и «Микропроцессорная техника в исследовании и управлении энергетических комплексов» для студентов направлений 13.03.02 и 13.04.02 Электроэнергетика и электротехника.

Выполнение данных работ направлено на формирование следующих компетенций:

– способность формулировать цели и задачи исследования, выявлять приоритеты решения задач, выбирать и создавать критерии оценки;

– способность применять современные методы исследования, оценивать и представлять результаты выполненной работы;

– способность использовать углубленные теоретические и практические знания, которые находятся на передовом рубеже науки и техники в области профессиональной деятельности;

– способность планировать и ставить задачи исследования, выбирать методы экспериментальной работы, интерпретировать и представлять результаты научных исследований;

– готовность обеспечивать требуемые режимы и заданные параметры технологического процесса по заданной методике.

В результате работы с представленным учебным пособием студент будет *знать* физические процессы передачи и преобразования сигналов; основные функции и структуру современных микроконтроллеров; *уметь* программировать микроконтроллеры; использовать углубленные знания в области микропроцессорных систем управления (МСУ) и полупроводникового преобразования энергии в профессиональной деятельности; анализировать внутренние и выходные сигналы МСУ; *владеть* практическими навыками проектирования и исследования МСУ; способностью к профессиональной

эксплуатации и разработки МСУ; методами исследования и анализа временных характеристик МСУ.

Цель самостоятельных работ – обучение работе в среде Arduino при разработке и отладке проектов автоматизированных систем световой индикации и вывода информации.

Задачи самостоятельных работ:

– приобретение практических навыков составления схем подключения и принципиальных схем при помощи программы Fritzing;

– приобретение практических навыков составления алгоритмов работы программных кодов в MS Visio для систем управления;

– приобретение практических навыков составления и грамотного оформления программных (исходных) кодов для управления автоматизированными системами световой индикации и вывода информации;

– приобретение практических навыков работы со светодиодным оборудованием (светодиоды, RGB-светодиоды, семисегментные индикаторы, матричные модули) и специализированными микросхемами;

– приобретение практических навыков работы с LCD-дисплеями с различными интерфейсами.

1 Микроконтроллерная плата Arduino

Зачем нужно использование Arduino в учебном процессе?

При обучении огромное внимание уделяется самостоятельной работе. Качественное усвоение лекционного материала и его практическое закрепление часто невозможно без внеаудиторной работы, которая в первую очередь должна быть не только посильна, но интересна студенту. В этом случае, основанный на достижении современной электроники и схемотехники программно-аппаратный комплекс Arduino может использоваться для организации практических, лабораторных работ, а также для самостоятельного обучения проектированию и реализации систем автоматики и робототехники.

Что же такое Arduino?

В настоящее время Arduino представляет из себя торговую марку, которая занимается разработкой аппаратно-программных средств. Аппаратная реализация представлена рядом печатных плат, основой которых являются широко распространённые микроконтроллеры. Программная составляющая представлена в виде бесплатных программных оболочек (IDE), которые позволяют разрабатывать и компилировать код и использовать его непосредственно для управления внешними устройствами. Открытая архитектура систем на Arduino предусматривает их свободное копирование и дополнение любыми компонентами современной электроники и компьютерной техники.

Сложно ли работать с Arduino?

Огромное количество научно-методических публикаций, учебной и справочной литературы, а также видео уроков позволяют осваивать основы за несколько часов.

Что из себя представляет аппаратная составляющая Arduino?

В основе аппаратной части лежат печатные платы, оснащённые программируемым микроконтроллером, которые подключаются к компьютеру при помощи COM-порта.

Дополнительные возможности реализуются за счёт расширения стандартной платформы вспомогательными модулями «шилдами» (shields), которые подключаются по принципу «бутерброда», как показано на рисунке 1.

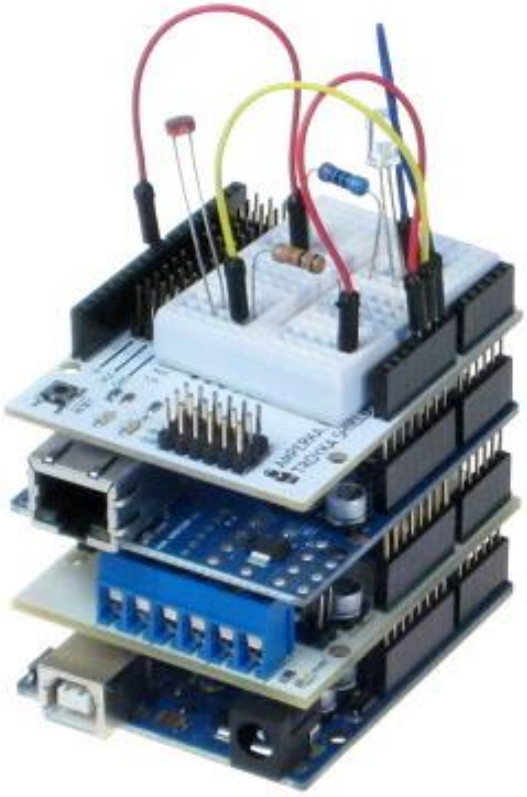


Рисунок 1 – Микроконтроллерная плата Arduino и дополнительные модули

Системы на базе Arduino можно подключать к локальной сети, интернету, GPS, Wi-fi. С их помощью управлять двигателями различных типов и мощностей, принимать сигналы с любых датчиков и многое другое.

Разрабатываемые системы на платформе Arduino не имеют никаких ограничений по количеству элементов или аппаратным особенностям. Возможности проектов зависят лишь от фантазии разработчиков.

Рассмотрим подробнее наиболее используемые микроконтроллерные платы из семейства Arduino (UNO и Mega), опишем конструкцию и основные характеристики, а также укажем на их основные достоинства и недостатки при проектировании простейших систем автоматики и робототехники.

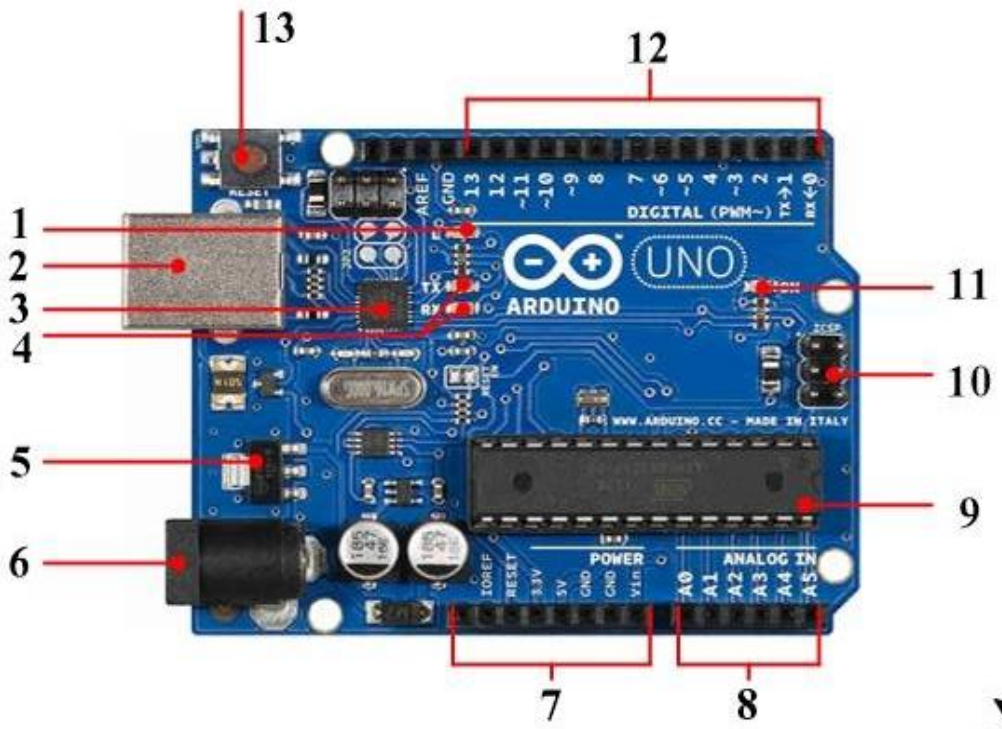
1.1 Arduino Uno

Uno является наиболее распространённой микроконтроллерной платой из всей линейки Arduino. Она наиболее актуальна для проектов, в которых не требуется применения большого количества аналоговых (не больше 6) и цифровых устройств (не больше 14). Небольшие размеры, низкая стоимость и широкая доступность в магазинах электроники во многих случаях определяют выбор именно этой платы. На рисунке 2 представлены вид сверху (а) и снизу (б) Arduino Uno. Основные характеристики и габариты приведены в таблице 1 [5, 11].

а) б)

Рисунок 2 – Внешний вид Arduino Uno

Микроконтроллер работает с тактовой частотой 16 МГц от кварцевого резонатора. Имеются разъемы USB, питания, кнопка сброса и перемычка питания. Питание платы осуществляется через USB порт, либо от внешнего источника (блок питания, аккумулятор) подключаемого к выводам Gnd и Vin разъема POWER. Источник выбирается автоматически. Передача данных на плату осуществляется через USB. Основные элементы указаны на рисунке 3. Описание входов и выходов приведено в таблице 2.



7. Выводы питания на плате

Наименование входа	Описание (назначение)
VIN	Это входное напряжение на плате при использовании внешнего источника питания.
5V	Этот вывод регулятора напряжения. Выход 5V может использоваться в экспериментах с микроконтроллером.
3.3 V	Этот вывод регулятора напряжения. Выход 3.3 V может использоваться в экспериментах с микроконтроллером.
GND	Шина заземления.
IOREF	Это вывод опорного напряжения платы, с которым работает ATmega328. Может использоваться для измерения и контроля напряжения платы.

1. Светодиод Pin13 (L)
2. Разъём USB
3. Микросхема USB интерфейса
4. Светодиоды RX+TX
5. Стабилизатор напряжения
6. Разъём питания
7. Выводы питания
8. Аналоговые входы
9. Микроконтроллер ATmega328
10. Разъём ICSP
11. Светодиод питания
12. Цифровые входы/выходы
13. Кнопка сброса

Рисунок 3 – Основные элементы Arduino Uno

Таблица 1 – Технические характеристики Arduino Uno

Наименование параметра	Значение параметра
Микроконтроллер	ATmega328
Оперативное напряжение, В	5
Рекомендованное входное напряжение, В	7-12
Предельные значения входного напряжения, В	6-20
Аналоговые входы, шт.	6
Цифровые входы/выходы, шт.	14
Максимальный ток входов/выходов, мА	40
Максимальный ток вывода 3,3 В, мА	50
Объем Flash памяти, кБ	32
Объем оперативной памяти, кБ	2
Объем энергонезависимой памяти, кБ	1
Тактовая частота, МГц	16
Длина, мм	68,6
Ширина, мм	53,4
Масса, грамм	25

Таблица 2 – Входы и выходы Arduino Uno

Наименование входа/выхода	Назначение (описание)
Serial: 0 (RX) и 1 (TX)	Используются для получения (RX) и посылки (TX) цифровых данных по последовательной шине.
Внешнее прерывание: 2 и 3	Эти выводы используются для генерирования прерывания настроены так, чтобы генерировать прерывания, если на них в режиме входа меняется входной сигнал.
PWM (ШИМ): 3, 5, 6, 9, 10, 11	8-битный широтно-импульсный модулятор (ШИМ).
SPI: 10(SS), 11(MOSI), 12(MISO), 13(SCK)	Эти выводы могут быть использованы для прошивки других микроконтроллеров с помощью Ардуино.
LED: 13	На 13 выводе встроен светодиод, который горит, если на выводе 13 высокий уровень, и не горит, если на нём низкий уровень.
6 аналоговых входов: (A0 – A5)	Имеют 10 битный аналого-цифровой преобразователь (АЦП). На вход этих выводов следует подавать 5 В от измеряемых сигналов.
TWI: A4 или SDA A5 или SCL	Эти выводы используются для соединения внешних устройств с Ардуино по шине I^2C.
AREF	Этот вывод используется для задания микроконтроллеру внешнего опорного напряжения.
Reset	Вывод, который при замыкании на «землю» (GND), вызывает программный сброс микроконтроллера.

Приведённые в разделе 1.1 сведения достаточны для начинающего пользователя, чтобы иметь представление о структуре и характеристиках основных элементов Arduino Uno.

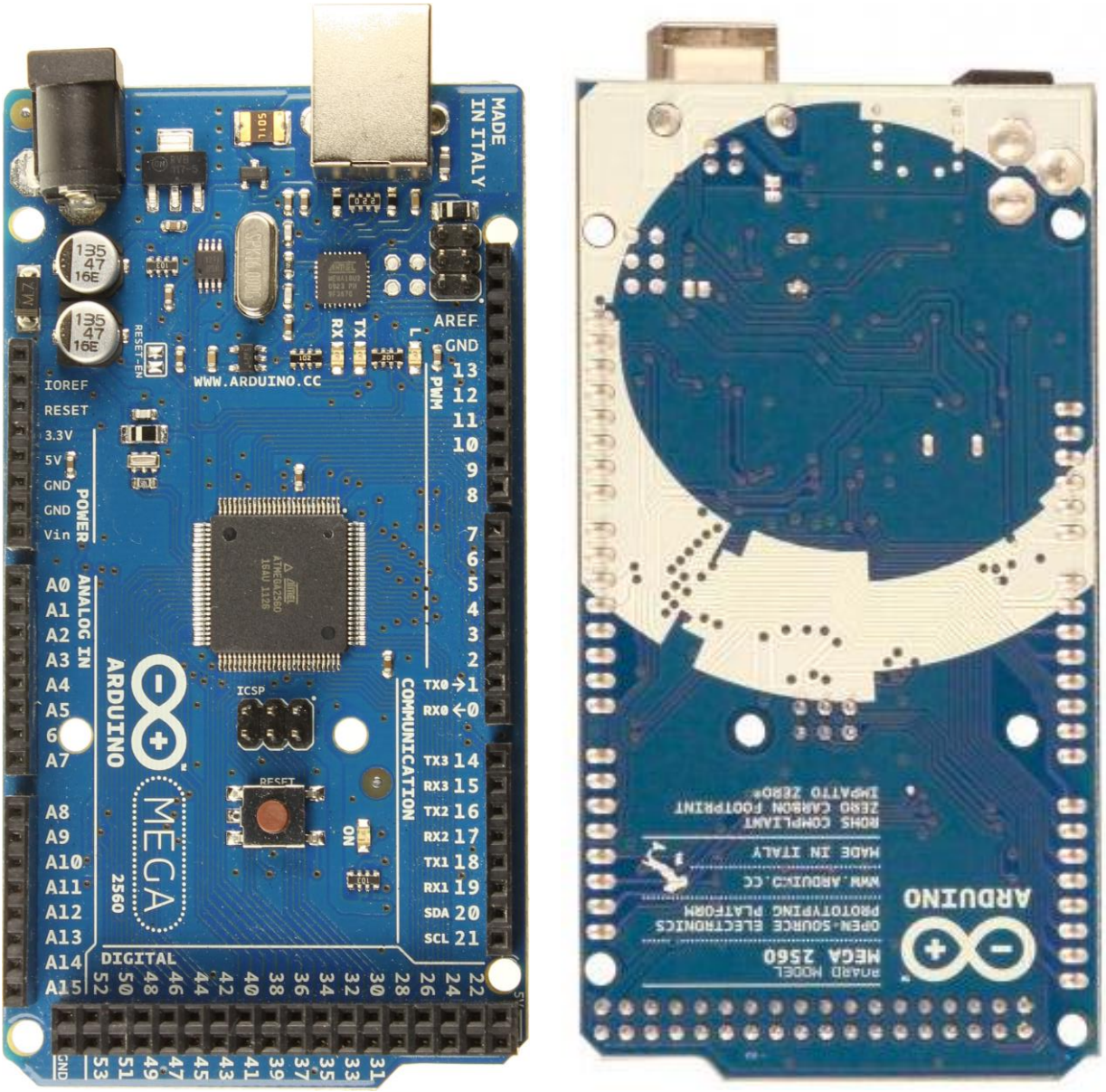
1.2 Arduino Mega

Arduino Mega реализована на микроконтроллере ATmega2560 и применяется в проектах, где требуется большое количество цифровых и аналоговых входов/выходов (до 54 и 16 соответственно) и ресурсов Arduino Uno не хватает. Четырнадцать выводов этой платы поддерживают ШИМ. К основным коммуникационным портам можно отнести четыре UART. На плате размещены разъёмы USB, питания и ICSP (последовательный интерфейс внутрисхемного программирования). Технические характеристики в общем виде представлены в таблице 3 [5, 11].

Таблица 3 – Технические характеристики Arduino Mega

Наименование параметра	Значение параметра
Микроконтроллер	ATmega2560
Оперативное напряжение, В	5
Рекомендованное входное напряжение, В	7-12
Предельные значения входного напряжения, В	6-20
Аналоговые входы, шт.	16
Цифровые входы/выходы, шт.	54
Максимальный ток входов/выходов, мА	40
Максимальный ток вывода 3,3 В, мА	50
Объем Flash памяти, кБ	256
Объем оперативной памяти, кБ	8
Объем энергонезависимой памяти, кБ	4
Тактовая частота, МГц	16
Длина, мм	102
Ширина, мм	53
Масса, грамм	60

Виды сверху (а) и снизу (б) Arduino Mega представлены на рисунке 4. Расположение основных элементов аналогично Arduino Uno.



а) б)

Рисунок 4 – Внешний вид Arduino Mega 2560

Описание входов и выходов приведено в таблице 4.

Для Mega, как и для рассмотренной ранее Uno существует ряд модификаций. Наиболее известные из них Mega, Mega 2560 и Mega ADK. Основными их отличиями являются микропроцессоры ATmega1280 и ATmega2560. На последних двух установлены USB на ATmega8U2. Также

основной особенностью ADK является встроенный USB-host для подключения к устройствам, работающим на Android.

Рассматриваемая платформа может функционировать при внешнем напряжении от 6 до 20 В. Отметим, что рекомендуемый рабочий диапазон входного напряжения от составляет 7 до 12 В.

Arduino Mega оснащена токовой защитой, реализованной самовостанавливающейся плавкой вставкой, которая защищает порт USB от коротких замыканий и сверхтоков. Прерываемый предохранителем ток через порт составляет выше 500 мА.

Таблица 4 – Входы и выходы Arduino Mega

Наименование входа/выхода	Назначение (описание)
0 (RX) и 1(TX), 19 (RX) и 18(TX) 17 (RX) и 16 (TX), 15 (RX) и 14 (TX)	Используются для получения (RX) и посылки (TX) цифровых данных TTL. Выводы подключены к микросхеме последовательной шины ATmega8U2
Внешнее прерывание: 2, 3, 18, 19, 20, 21	Эти выводы настроены так, чтобы генерировать прерывания, если на них в режиме входа меняется входной сигнал.
PWM (ШИМ): 2 – 13, 44 – 46	8-битный широтно-импульсный модулятор.
SPI: 53(SS), 51(MOSI), 50(MISO), 52(SCK)	Эти выводы могут быть использованы для прошивки других микроконтроллеров с помощью Ардуино.
16 аналоговых входов: (A0 – A15)	Имеют 10 битный аналого-цифровой преобразователь (АЦП). На вход этих выводов следует подавать 5 В от измеряемых сигналов.
TWI: 20 (SDA), 21 (SCL)	Эти выводы используются для соединения внешних устройств с Ардуино по шине I^2C.

Представленные платы Uno и Mega чаще всего используются как приёмники информации в системе или являются основой разрабатываемых систем на данной платформе. Однако при разработке автоматизированных систем, чаще для решений задач роботехники, когда устройствам характерны небольшие габариты или выполнение единичных функций (отправка сигнала, выполнение пары-тройки задач) применяют Arduino Nano (существуют и более компактные решения, например Arduino Mini, но они менее востребованы).

1.3 Arduino Nano

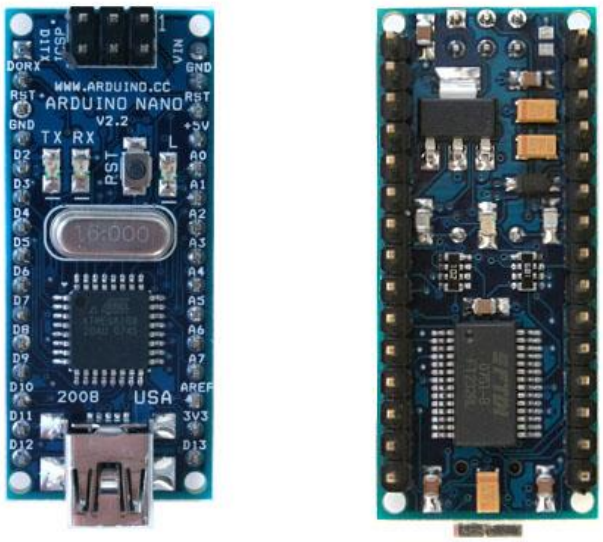
В основе платформы Nano лежит микроконтроллер ATmega328 в поздних версиях ATmega168. Для подключения устройства к компьютеру используется кабель Mini-B USB [5, 11].

Технические характеристики Arduino Nano представлены в таблице 5.

Таблица 5 – Технические характеристики Arduino Nano

Наименование параметра	Значение параметра
Микроконтроллер	ATmega168 или ATmega328
Оперативное напряжение, В	5
Рекомендованное входное напряжение, В	7-12
Предельные значения входного напряжения, В	6-20
Аналоговые входы, шт.	8
Цифровые входы/выходы, шт.	14
Максимальный ток входов/выходов, мА	40
Максимальный ток вывода 3,3 В, мА	50
Объем Flash памяти, кБ	16 и 32
Объем оперативной памяти, кБ	1 и 2
Объем энергонезависимой памяти, кБ	1
Тактовая частота, МГц	16
Длина, мм	18,5
Ширина, мм	42

Виды сверху (а) и снизу (б) Arduino Nano представлены на рисунке 5.



а) б)

Рисунок 5 – Внешний вид Arduino Nano

При сравнении технических характеристик, представленных в таблицах 5 и 1, видно, что структуры Nano и Uno аналогичны, различаются только используемыми микроконтроллерами и размерами.

Отметим, что исполнение выводов портов у Nano отличаются от рассмотренных ранее Uno и Mega (вилки «папа» взамен гнезд «мама»).

Вопросы по разделу:

1. Что понимается под Arduino?
2. Из чего состоит аппаратная часть Arduino?
3. Из чего состоит программная часть Arduino?
4. Каким образом расширяются возможности базовых плат Arduino?
5. В чем отличия между платами Uno, Mega и Nano?
6. Сколько выводов Uno, Mega и Nano поддерживают ШИМ?
7. Какие защиты реализованы в Uno, Mega и Nano?
8. Как передаются данные с компьютера на плату Arduino?
9. Какие критерии определяют выбор платы Arduino для проектов?
10. Какие достоинства и недостатки Arduino вы можете выделить?

2 Теоретические основы

2.1 Сигналы

Сигнал – это некий специальный код, передаваемый в пространство одной или несколькими системами.

В области передачи информации сигналом считают носитель данных, который применяется для передачи некоторых сообщений.

Измерительный радиотехнический сигнал – это электрическое напряжение или ток, изменяющиеся во времени, с заранее известными характеристиками, используемые для измерения характеристик радиотехнических цепей и их контроля [6].

Наиболее распространённые классификации сигналов известны из курса физики. Представим их в виде схемы на рисунке 6.

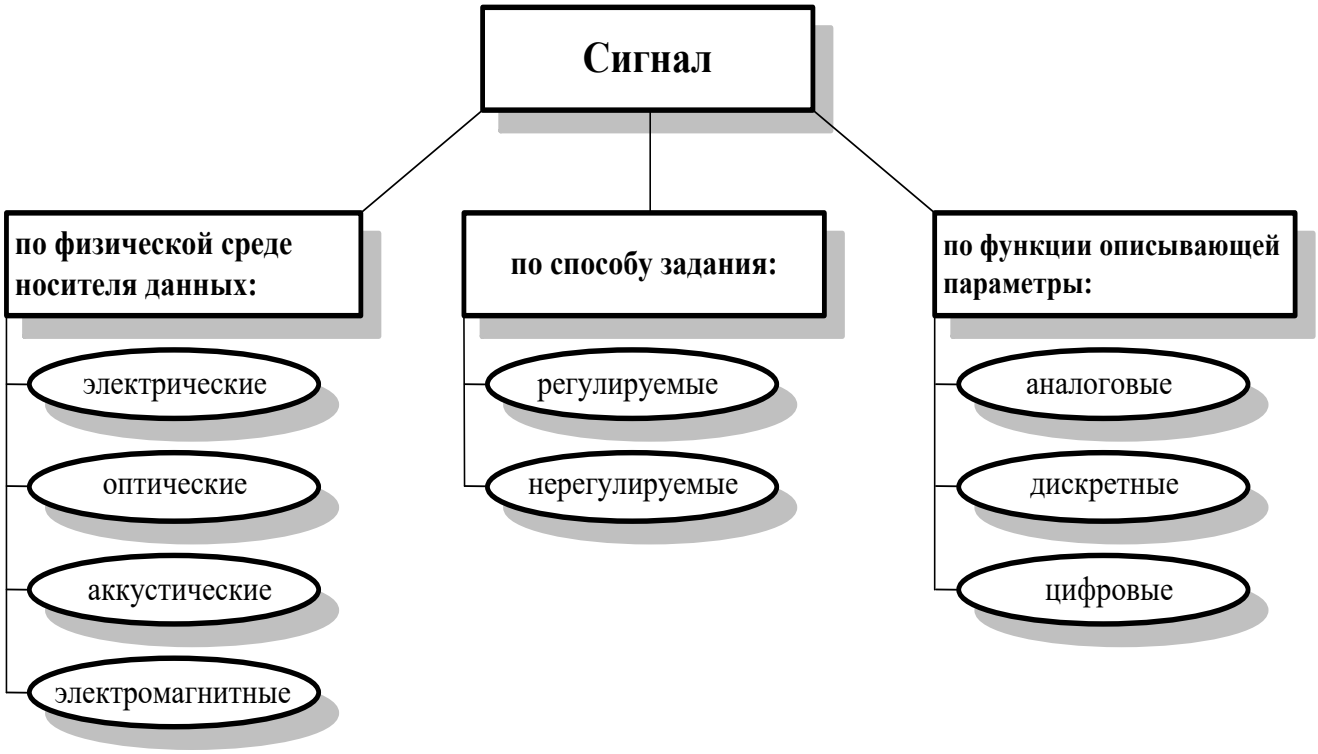


Рисунок 6 – Схема классификаций сигналов

Обработка сигнала осуществляется для передачи и получения информации, зашифрованной (закодированной) в нём. Данные могут быть использованы только после извлечения, например, форматирования, сжатия, восстановления или передачи.

При работе с дискретными, цифровыми и аналоговыми сигналами используются такие методы обработки, как фильтрация, свертка и корреляция. Они применяются для его восстановления от шума и повреждений.

Форматирование сигналов осуществляется специальными устройствами – преобразователями: аналого-цифровым (АЦП) и цифро-аналоговым (ЦАП).

Формы аналогового (сверху) и дискретного (снизу) сигналов в общем виде представлены на рисунке 7.

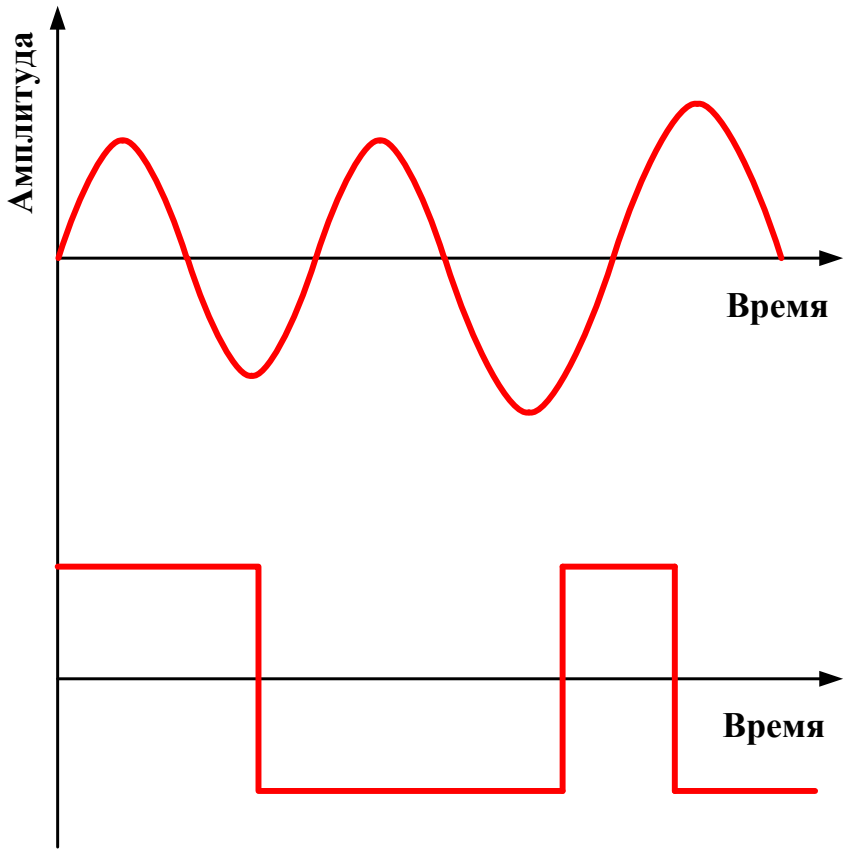


Рисунок 7 – Формы аналогового и дискретного сигналов

Аналоговый сигнал – это непрерывный во времени способ передачи данных. Основным его недостатком является наличие помех, иногда приводящих к полной потере передаваемой информации. Доля шума может составлять значительную часть сигнала и в коде сложно определить, где нужные данные, а где искажения. По этой причине цифровая обработка находит большую популярность и вытесняет аналоговую [8].

Пример искажения рассматриваемых сигналов представлен на рисунке 8.

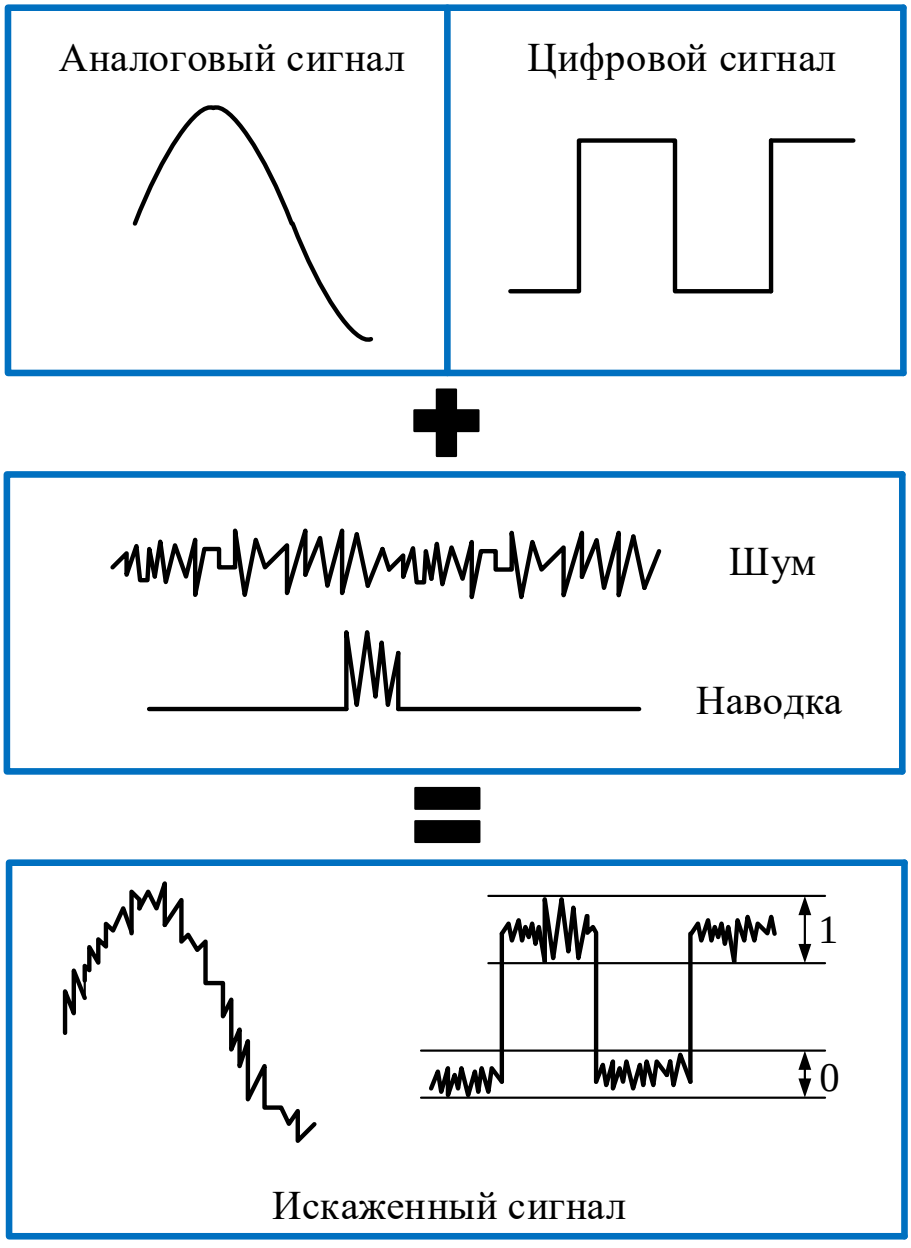


Рисунок 8 – Пример искажения аналогового и дискретного сигналов

Цифровой сигнал – это поток данных, который описывается некой дискретной по времени функцией, и амплитуда может быть принята из уже заданных значений (квантование по уровню). Его главным преимуществом является фильтрация большей части получаемых помех. Этот вид передачи данных представляет собой двоичный поток данных, который позволяет отправлять коды одновременно по нескольким физическим каналам.

В ретрансляторе цифровой сигнал восстанавливается полностью, если шум не будет превышать определенного порога. Однако, если код поступает со значительными искажениями, то он не может быть восстановлен. В случае с аналоговым ретранслятором часть информации может быть воспроизведена. Решение этой проблемы для цифровой передачи данных более энергозатратен, хотя в современной технике реализован и активно применяется.

Дискретный сигнал создаёт не саму форму сигнала, а её цифровой вид, что позволяет уменьшить общее количество передаваемых данных и организовать пакетную отправку. Чаще всего его используют для переноса непрерывной волны, применяемой для голосового или видео общения, где необходим канал с большой пропускной способностью.

Метод цифровой передачи данных заключается в пересылке данных, собранных в особенные блоки, что позволяет отправлять небольшие объёмы информации.

Таким образом, аналоговый сигнал описывает данные непрерывными функциями времени, при этом его амплитуда колебаний может принимать любые значения из определённого диапазона. Цифровой сигнал описывается дискретными функциями, и его амплитуда равна строго заданным значениям (квантование по уровню). Главным преимуществом цифровой передачи данных от аналоговой является малая чувствительность к помехам. Не вся имеющаяся техника способна принимать оба вида сигналов, поэтому переход на цифровые устройства, являющийся очевидным, займет не один год.

2.2 Широтно-импульсная модуляция

Широтно-импульсная модуляция (ШИМ или PWM) – это способ управления мощностью, подаваемой к нагрузке, путём изменения длительности импульса при неизменной амплитуде и частоте [14].

Использование широтно-импульсной модуляции значительно повышает коэффициент полезного действия электрических систем и упрощает их реализацию. ШИМ широко применяется для регулирования яркости свечения LCD дисплеев, скорости вращения двигателей и т.д. Её используют в качестве своеобразного цифро-аналогового преобразователя на выходах микроконтроллера (достаточно добавить фильтрующую RC цепочку).

Рассмотрим основные параметры ШИМ сигнала на рисунке 9.

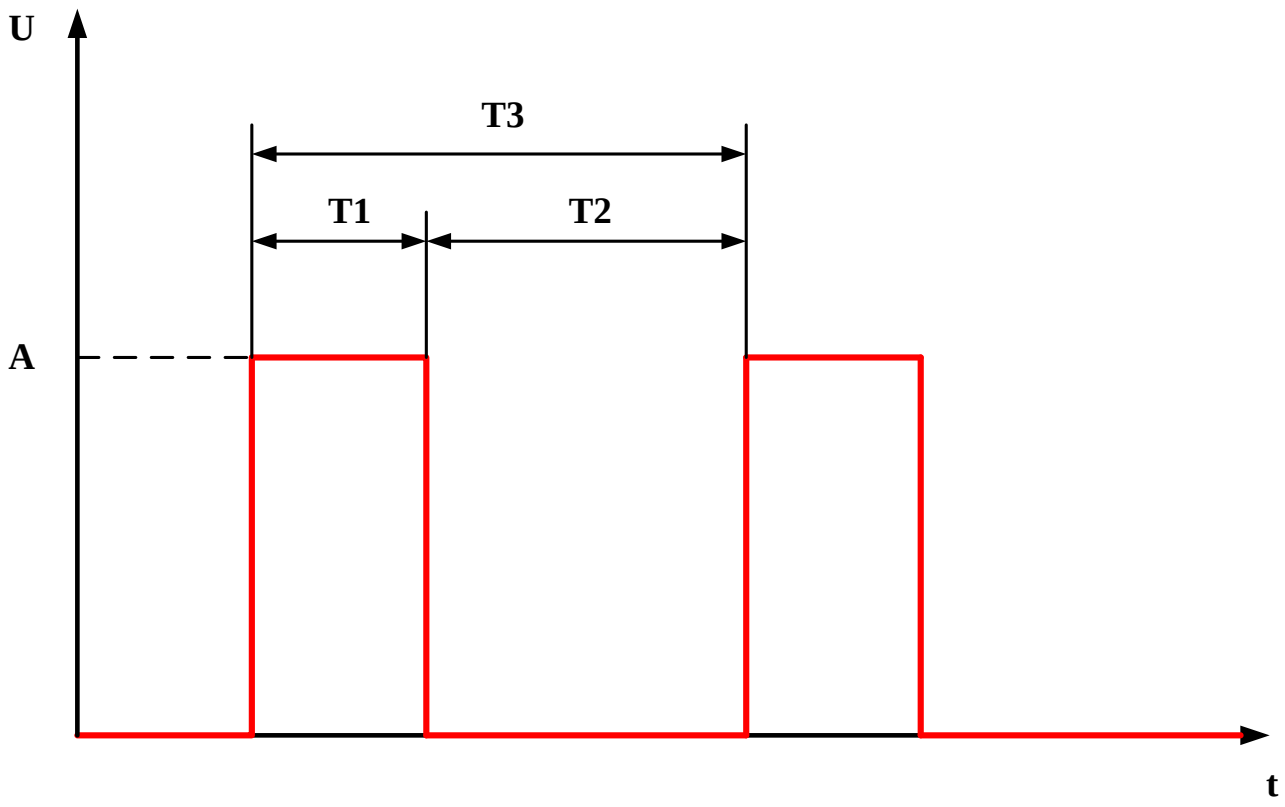


Рисунок 9 – ШИМ сигнал и его основные параметры

На рисунке 9 обозначены следующие параметры: амплитуда импульсов A, время включённого состояния T1, выключенного состояния T2, время периода PWM T3. Мощность на некоторой нагрузке пропорциональна отношению времени T1 и T3. При этом отношение T1 к T3 называют коэффициентом заполнения ШИМ, а обратную величину - скважностью.

На выходе Arduino может быть или напряжение питания 5 В, или земля 0 В. При этом, чтобы управлять уровнем напряжения (яркость светодиода, скорость вращения двигателя и т.д.) используется широтно-импульсная модуляция.

На рисунке 10 представлены желаемое напряжение и соответствующий ей ШИМ сигнал.

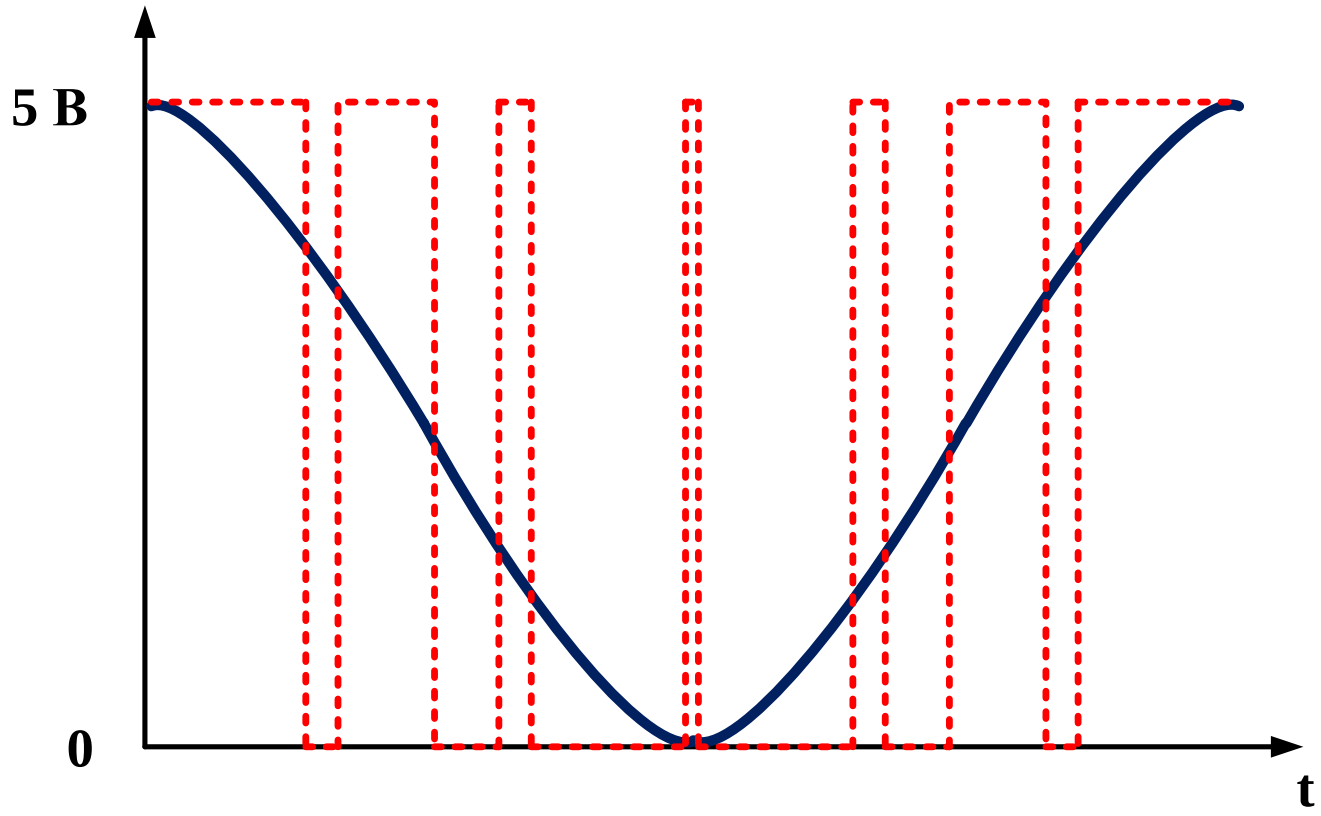


Рисунок 10 – Желаемое напряжение и ШИМ-сигнал

На практике выход микроконтроллера переключается с 0 до 5 В очень быстро (1 – 8 кГц), при этом глаз человека видит мерцания выше 50 Гц, поэтому кажется, что светимость индикаторов меняется плавно.

На рисунке 11 представлено несколько примеров с различной скважностью сигнала для напряжения 5 В.

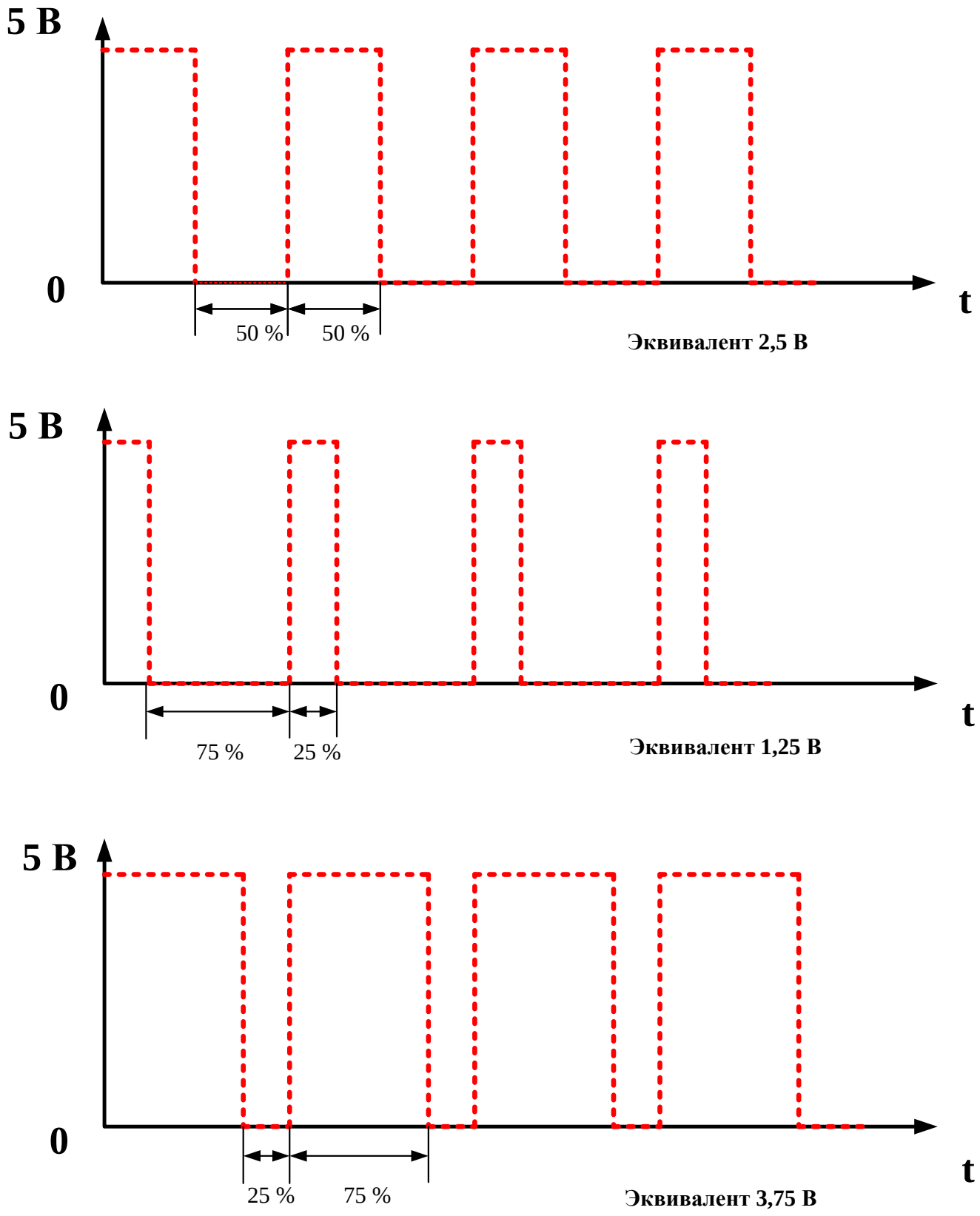


Рисунок 11 – Примеры различной скважности ШИМ

Рассмотренные в первой главе платы Arduino, в зависимости от модели имеют различное количество выводов, поддерживающих широтно-импульсную модуляцию. На плате они обозначены знаком «~», а подробная нумерация приведена в таблицах 2 и 4 соответственно.

Системные функции при работе с ШИМ и особенности функционирования на практике разберем при выполнении самостоятельных работ.

2.3 Основные законы

На практике часто приходится выбирать из имеющихся в наличии элементов (резисторов, конденсаторов, блоков питания и т.д.) подходящие. Именно поэтому не стоит забывать основы для всех электриков, а именно закон Ома и всё с ним связанное. Мнемоническая диаграмма, в которой собраны основные формулы, представлена на рисунке 12.

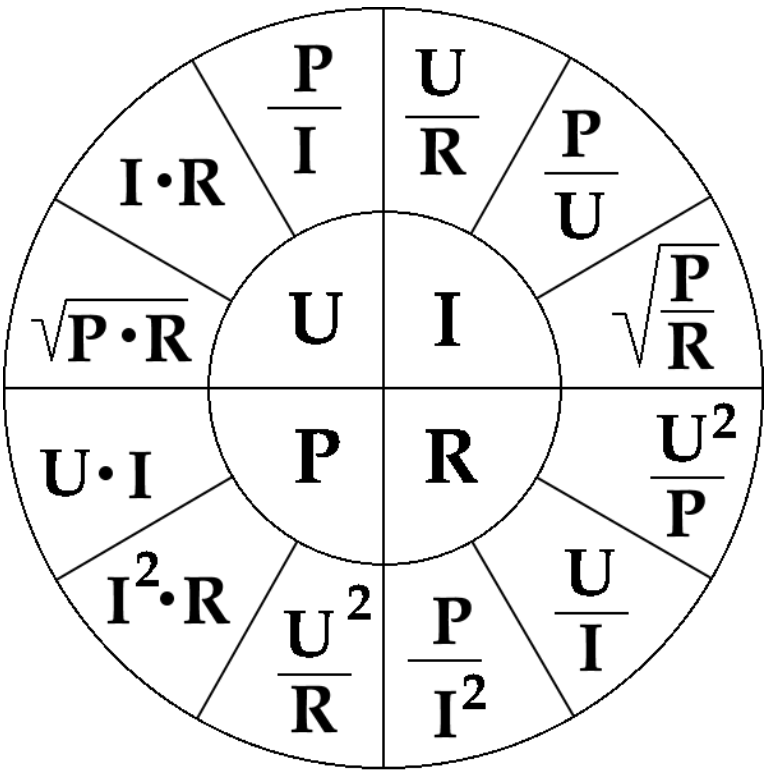


Рисунок 12 – Мнемоническая диаграмма

На рисунке 12 введены следующие обозначения: U – электрическое напряжение, I – сила тока, R – сопротивление, P – электрическая мощность.

На практике умение определять общее сопротивление экономит достаточно много времени. Если в наличии нет требуемого номинала, то следует вспомнить, что существует последовательное и параллельное подключение резисторов:

– для последовательного соединения:

$$R_{общ} = R_1 + R_2 + ... + R_n; \quad (1)$$

– для параллельного соединения:

$$R_{общ} = \frac{1}{R_1} + \frac{1}{R_2} + ... \frac{1}{R_n}. \quad (2)$$

Имея три резистора номиналом 100, 200 и 300 Ом и соединив их параллельно получим 55 Ом, последовательно 600 Ом. Таким образом, перебирая варианты соединений можно получить огромное количество новых номиналов.

Вопросы по разделу:

1. Что понимают под сигналом и измерительным радиотехническим сигналом?

2. Какие существуют методы обработки сигнала?

3. Какую форму могут принимать радиотехнический сигналы?

4. Назовите достоинства и недостатки аналогового сигнала?

5. Назовите достоинства и недостатки цифрового сигнала?

6. Какой сигнал называют дискретным? В чем его достоинства и недостатки?

7. Что такое широтно-импульсная модуляция?

8. На примере использования Arduino приведите пример использования ШИМ.

9. Что такое мнемоническая диаграмма электрика?

10. Какими способами можно варьировать номиналы резисторов?

3 Предварительная подготовка к работе с Arduino

3.1 Arduino IDE

Для разработки программ, их компиляции и загрузки в Arduino используется бесплатная программа Arduino IDE. Для её скачивания необходимо пройти по ссылке: https://www.arduino.cc/en/main/software. Окно выбора версии для скачивания представлено на рисунке 13 [13].

Рисунок 13 – Окно выбора версии для скачивания Arduino IDE

На рисунке 13 область 1 содержит описание текущей версии программы, а область 2 ссылки на скачивание для различных версий операционных систем.

Отметим, что для Windows можно скачать файл установщик с расширением «*.exe» и архив, содержащий файлы программы с расширением «*.zip».

При выборе ссылки с названием подходящей операционной системы, откроется окно загрузки, представленное на рисунке 14. Для перемещения файла на компьютер необходимо нажать на ссылку, отмеченную стрелкой.

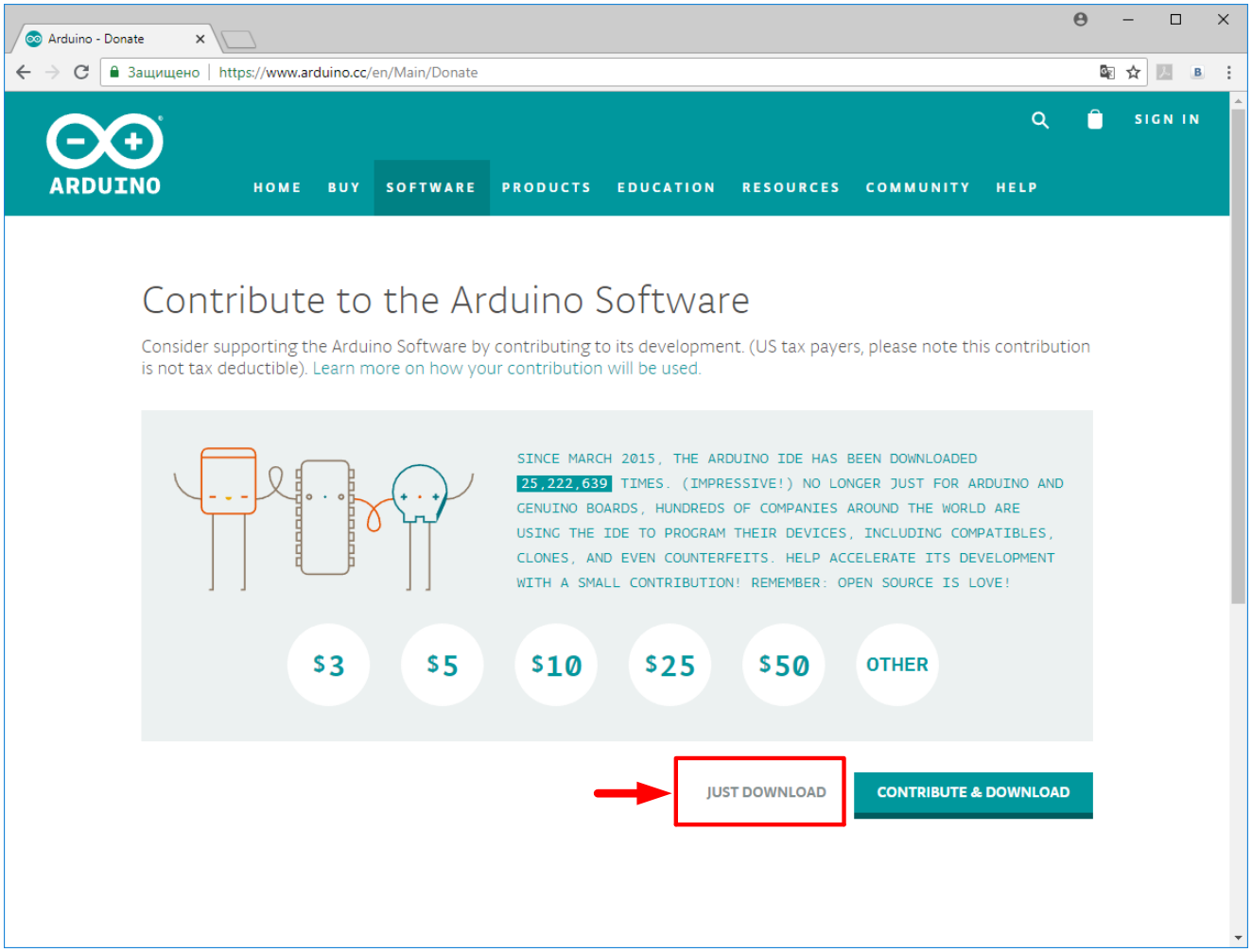


Рисунок 14 – Окно загрузки Arduino IDE

В случае запуска с файла расширением «*.exe» процесс установки включает четыре этапа: согласие с политикой конфиденциальности, выбор дополнительных файлов (драйверов, справок и т.д.), выбор папки для установки и непосредственного разархивирования файлов. Весь процесс установки схематично представлен на рисунке 15.

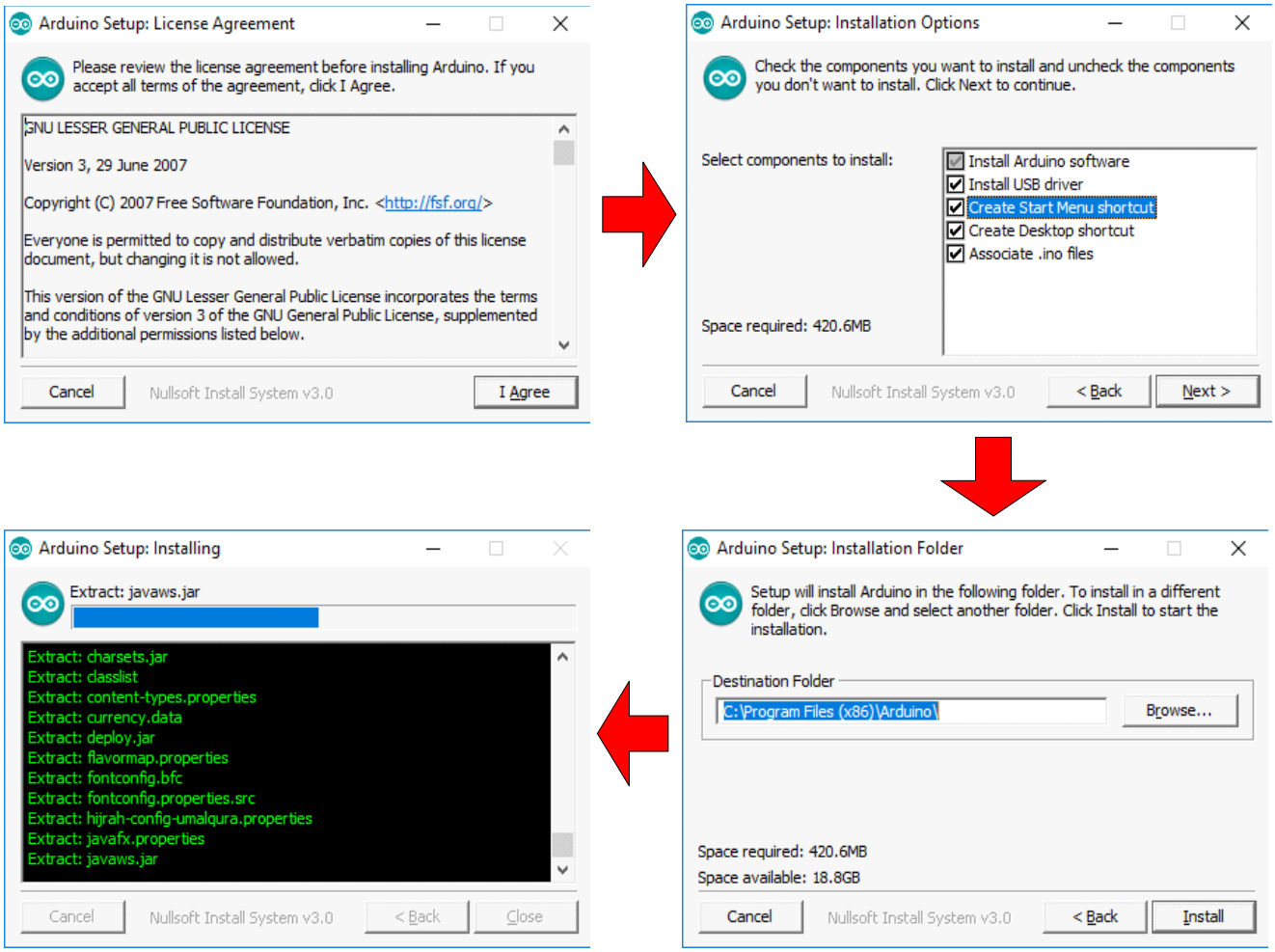


Рисунок 15 – Схема процесса установки Arduino IDE

После завершения установки на рабочем столе отобразится ярлык программы.

Рассмотрим подробнее настройки подключения платы Arduino в программе.

3.2 Настройка подключения

Окно программы Arduino IDE в общем виде представлен на рисунке 16. Область 1 называется панель инструментов. Она содержит пять вкладок: файл (работа с файлами программ загружаемых в микроконтроллер, функции визуализации программы, примеры и т.д.), правка (функции редактирования и работы с текстом программы и т.д.), скетч (функции работы со скетчем, загрузка

стандартных библиотек и т.д.), инструменты (функции работы с портом, мониторинг подключения и т.д.), помощь (справочная информация для начинающих, данные о разработчике и т.д.).

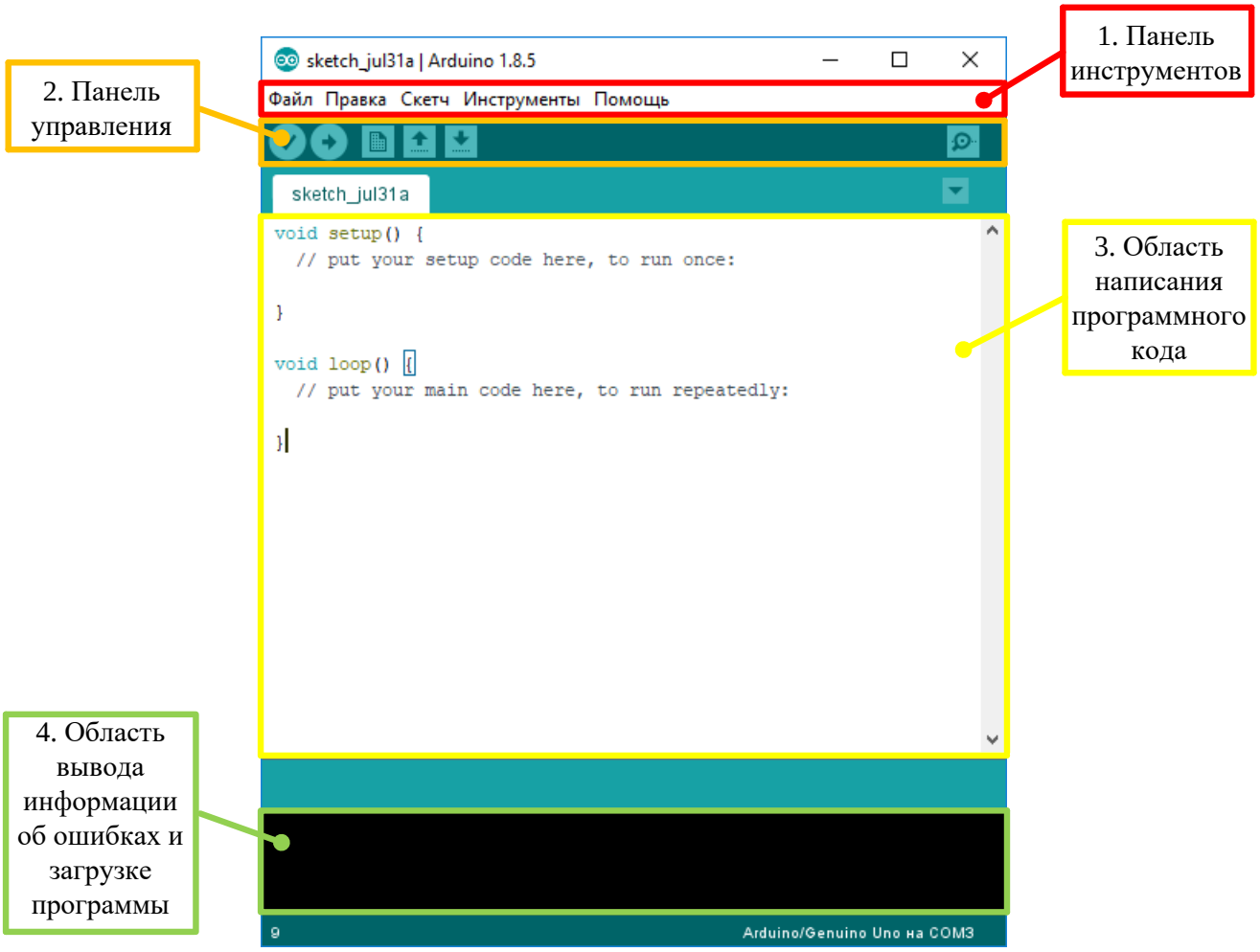


Рисунок 16 – Рабочее окно программы Arduino IDE

Для подключения платы Arduino к компьютеру необходимо воспользоваться USB портом и соответствующим кабелем.

Если ваша плата является оригинальной, то драйвер на неё установится автоматически, в противном случае необходимо это сделать самостоятельно. Его можно скачать с интернета (название «CH340» или «CH341»). Состояния драйвера можно проверить в диспетчере устройств, вкладка «Порты (COM и LPT)», как показано на рисунке 17.

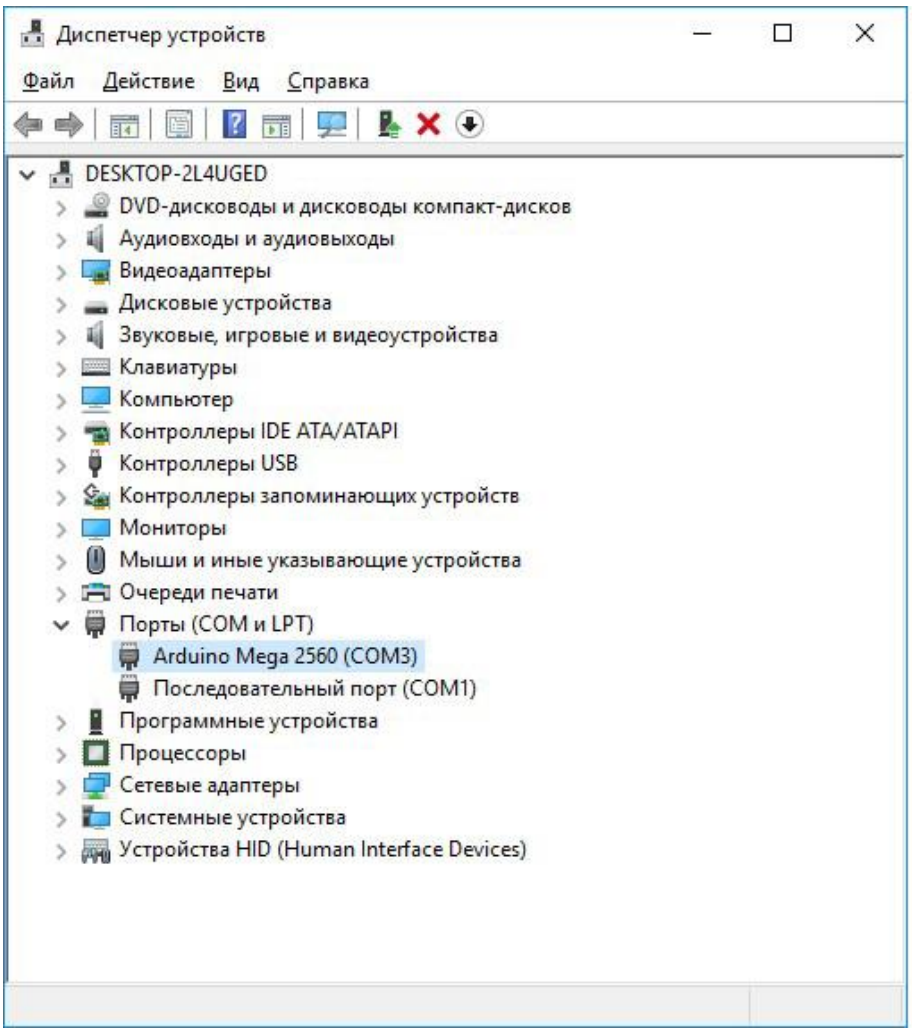


Рисунок 17 – Определение платы Arduino в диспетчере устройств

Проверить подключение непосредственно в программе можно перейдя в панели инструментов по вкладке «Порт:” ”», как показано на рисунке 18. В этой же вкладке переключаются между платами, если работают сразу с несколькими Arduino. В рассматриваемых примерах подключалась оригинальная Mega 2560.

Достаточно часто возникают проблемы с подключением, и, если не записывается программа в контроллер, необходимо в первую очередь проверить соединение. Для этого используется вкладка панели инструментов «Получить информацию о плате». Если соединение нормальное и компьютер «видит» контроллер, то отобразиться окно, пример которого представлен на рисунке 19, в противном случае переподключают устройство в USB порту.

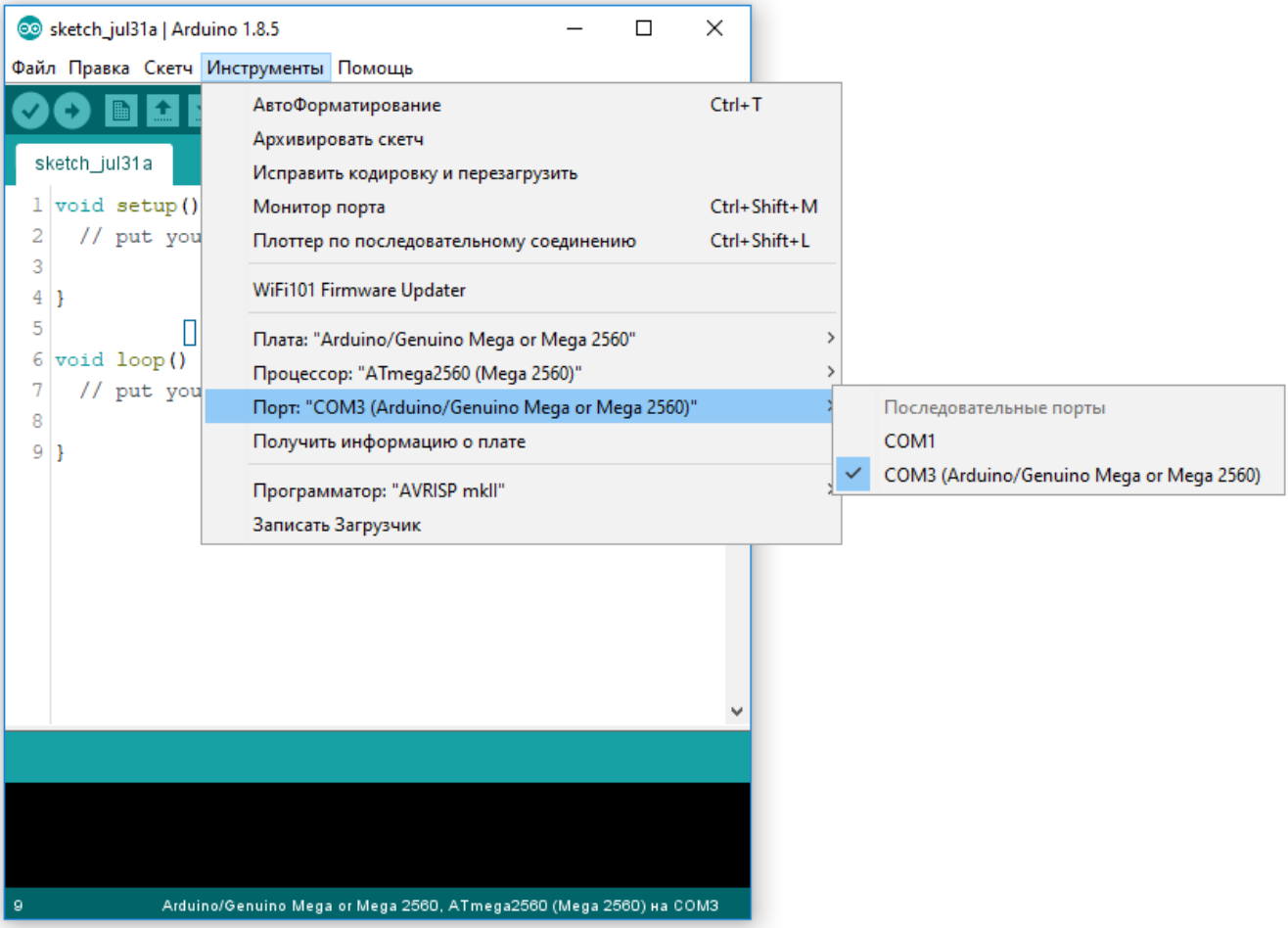


Рисунок 18 – Пример выбора COM порта для подключённой платы Arduino

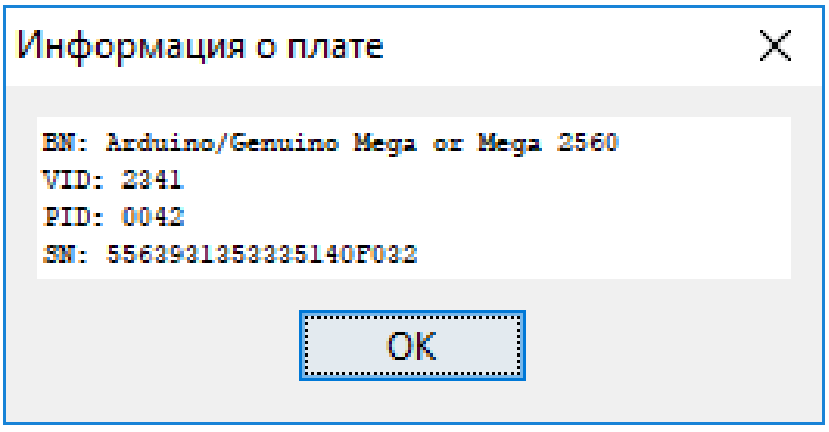


Рисунок 19 – Пример проверки информации о подключённой плате Arduino

Для компиляции программного кода обязательно указывать наименование используемой платы. Так же при использовании нескольких различных плат в проектах необходимо, как показано на рисунке 20, выбирать название Arduino для которой написан код. В противном случае она не будет работать, возможно зависание программы.

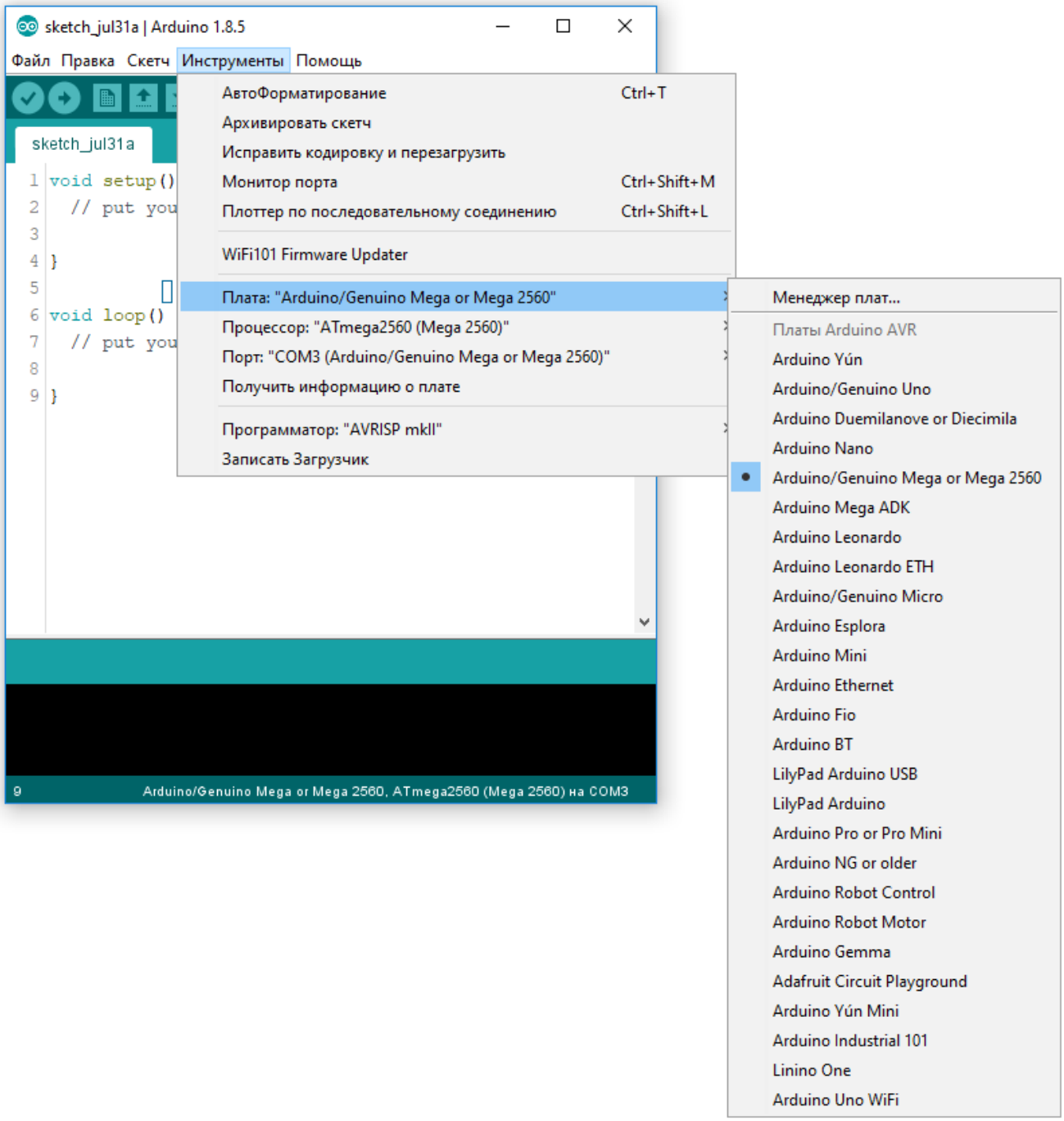


Рисунок 20 – Пример выбора используемой платы Arduino

Таким образом, чтобы избегать проблем с подключением платы к компьютеру необходимо:

– проверить в диспетчере устройств наличие драйвера;

– в программе Arduino IDE указать используемый COM порт;

– в программе Arduino IDE проверить информацию о плате;

– в программе Arduino IDE указать название используемой платы.

3.3 Последовательность работы

В этом разделе остановимся на основных этапах работы непосредственно с платой Arduino:

1. Осуществляется подключение к компьютеру, как описано ранее.

2. Собирается электрическая схема соединений компонентов системы или её части, которой будет управлять контроллер.

3. Разрабатывается программный код (скетч) в Arduino IDE.

4. Программный код проверяется на ошибки и компилируется (по необходимости сохраняется на компьютер).

5. Скетч загружается через COM-порт в микроконтроллер.

Рассмотрим на простейшем примере работу с Arduino. Заставим мигать встроенный в плату светодиод на 13 выводе.

Необходимый скетч уже написан и относится к примерам программ, которые встроены в Arduino IDE. Для его запуска необходимо воспользоваться вкладкой панели инструментов «Файл», как показано на рисунке 21.

Проверка кода на ошибки осуществляется левой крайней кнопкой на панели управления (круг с галочкой внутри). В случае отсутствия ошибок в области 4 рисунка 15 появится надпись: «Компиляция завершена» и далее текст с информацией белого цвета на чёрном фоне. Пример приведён на рисунке 22а с удалёнными комментариями. Аналогично используя вторую кнопку слева (круг со стрелкой) осуществляется загрузка скетча в контроллер, рисунок 22б.

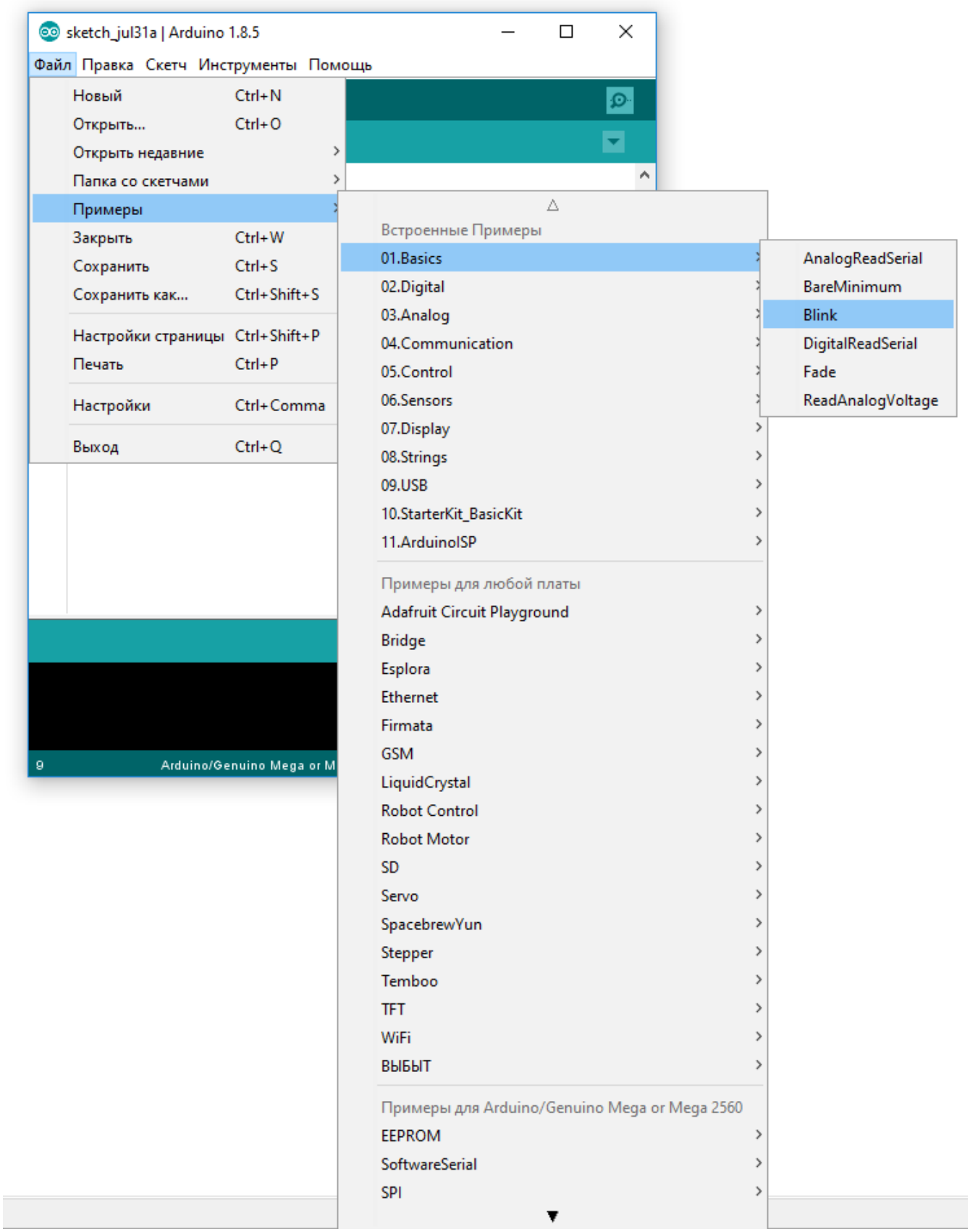
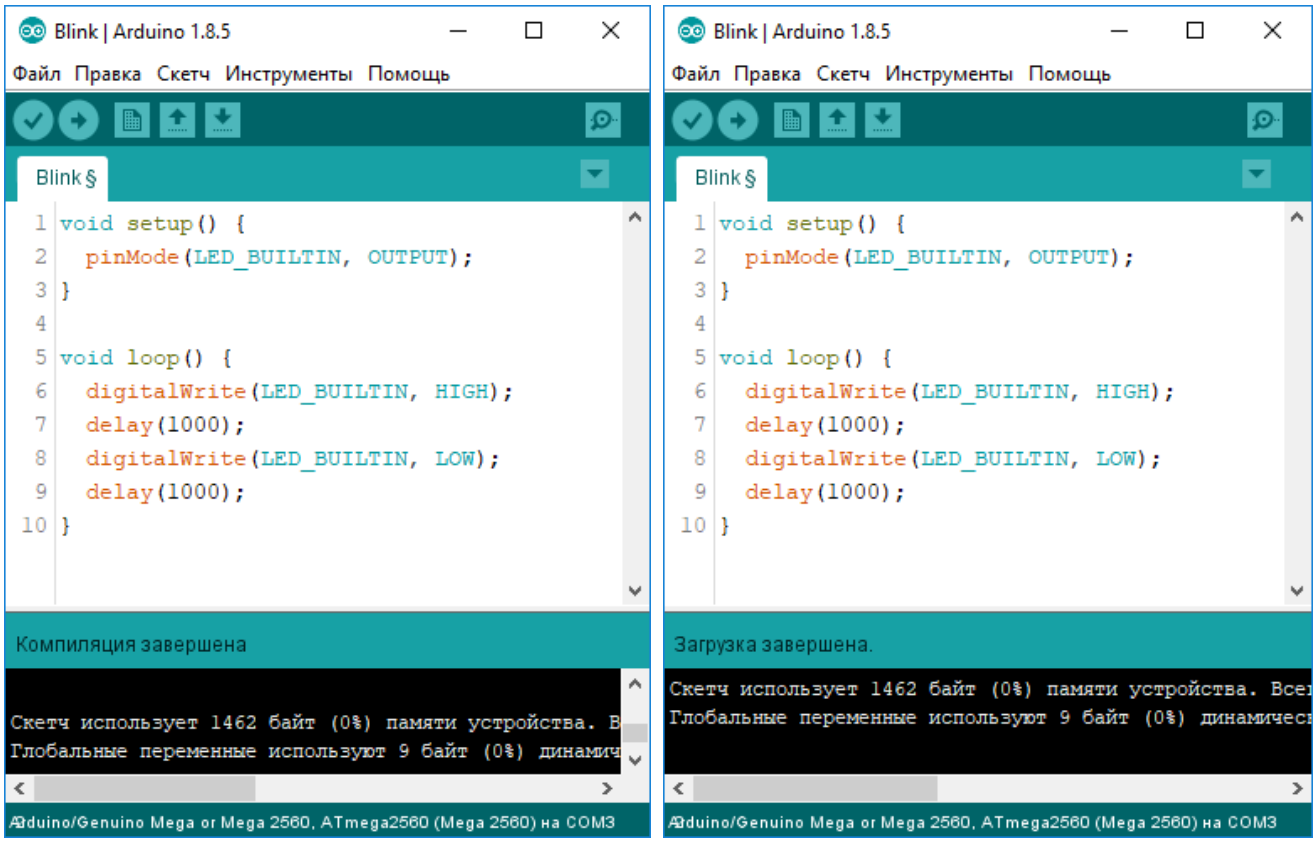


Рисунок 21 – Поиск примера скетча для мигания светодиодом на 13 выводе



а) б)

Рисунок 22 – Пример удачной компиляции и загрузки кода программы

Результат работы данной программы представлен на рисунке 23.

Рисунок 23 – Мигание светодиода на 13 выводе микроконтроллера

В зависимости от сложности управляемой системы в проекте может быть задействовано множество плат Arduino. При этом для каждой из них будет выполняться представленная в разделе последовательность. Важно грамотно распределять нагрузку на каждый задействованный в проекте контроллер и выбирать наиболее подходящие варианты плат под предъявляемые функциональные и конструктивные требования.

Вопросы по разделу:

1. Как установить Arduino IDE на компьютер?
2. Какие основные элементы содержит рабочее окно Arduino IDE?
3. Как определить правильность подключения Arduino к компьютеру?
4. Как в Arduino IDE выставить тип используемой платы?
5. Как исправить проблему подключения Arduino к компьютеру?
6. Опишите методику работы с Arduino IDE?
7. Как в Arduino IDE загрузить скетч из базы стандартных примеров?
8. Как убедиться, что написанный код в Arduino IDE не содержит ошибок, влияющих на работоспособность программы?
9. Как организуется и реализуется последовательность действий при загрузке скетчей на различные платы Arduino, используемые в одном проекте?
10. Какие параметры компьютера необходимы для работы с Arduino IDE?

4 Основы программирования Arduino

4.1 Структура программы

Язык программирования устройств Arduino основан на C/C++. Основными структурными элементами программы являются функции со следующим синтаксисом:

```
«Тип возвращаемого значения» «Имя функции» («Аргументы»)
{ // Начало тела функции
  // Тело функции (команды)
  return «что возвращать»;
} // Конец начала функции
```

Фигурные скобки (левая «{» и правая «}») на языке С обозначают начало и конец конкретных логических блоков – функций, условий, циклов и т.д. Если функция не предполагает возвращение каких-либо значений, то перед именем пишется слово «void». При этом команда return не пишется. Если у функции отсутствуют аргументы, то после имени обязательно ставятся две круглые скобки без текста между ними.

При открытии начального окна программы IDE, как показано на рисунке 24, автоматически отображаются два стандартные функции setup и loop.

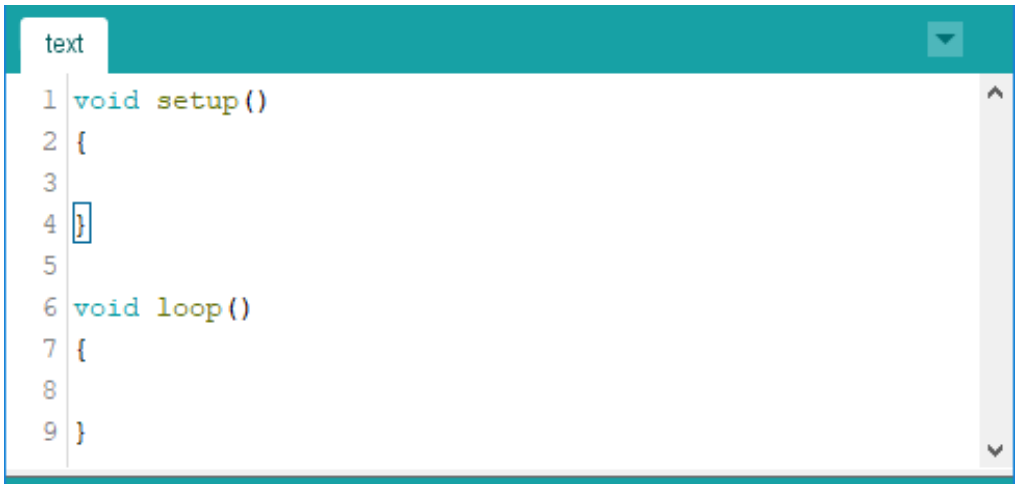


Рисунок 24 – Окно запуска программы Arduino IDE

Функция setup исполняется микроконтроллером один раз при запуске или перезапуске Arduino. Как правило она используется для инициализации входов/выходов платы, дополнительных функций и подгружаемых библиотек. Функция loop выполняется микроконтроллером циклично на протяжении всей работы и предназначена для размещения кода основной программы (цикла) [15].

Рассмотрим в качестве примера код программы, аналогичный представленному в предыдущем разделе и предназначенный для мигания светодиодом на 13 выходе. В общем виде он изображен на рисунке 25.

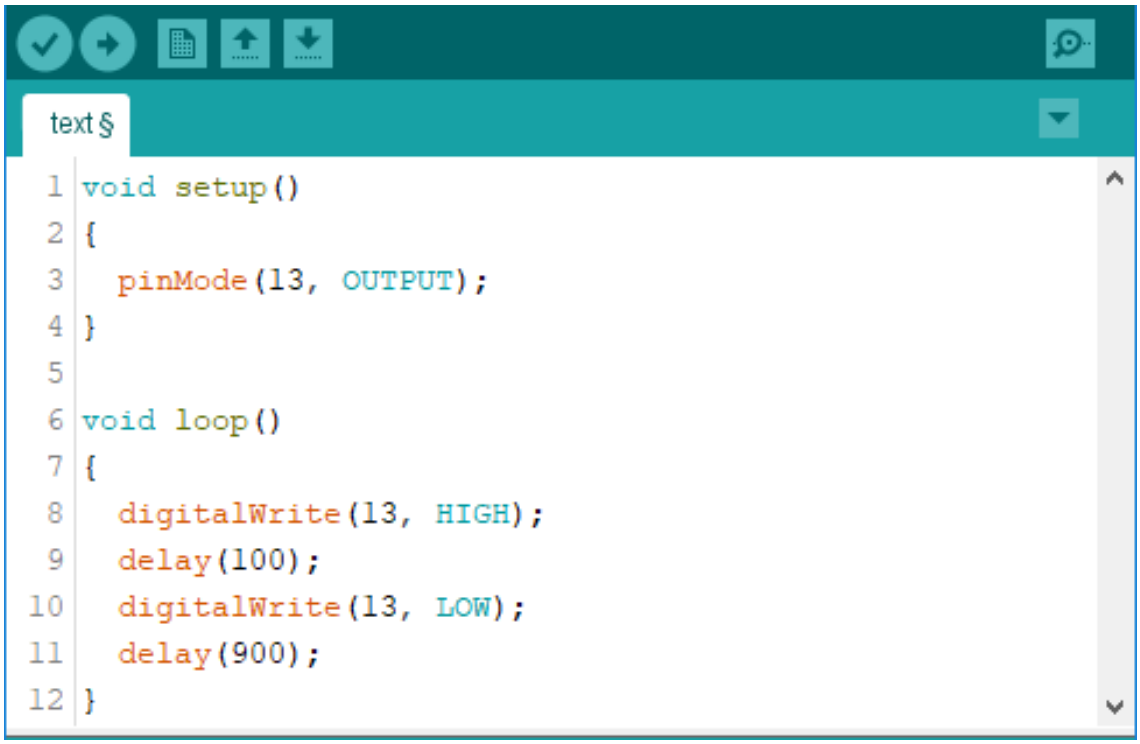


Рисунок 25 – Код программы для мигания светодиодом

Из рисунка видно, что функция setup содержит одну функцию, а loop четыре. Они и являются описанием команд для процессора и выполняются по порядку без пауз.

Выражение в строке 3 вызывает функцию pinMode, которая переводит указанный вывод в состояние приёма или передачи (INPUT или OUTPUT). После названия выражения в скобках указываются номер вывода платы, а затем через

запятую режим его работы. Эти уточняющие значения называются аргументами функции и их количество может быть различным (от 0 до ∞). В нашем примере 13 вывод определяется передающим на протяжении всего времени работы программы, так функция setup и её выражения предназначены для настройки платы.

Функция loop называется основным циклом и предназначена для выполнения полезной (видимой) работы. Содержащиеся в ней выражения вызывают функции digitalWrite и delay. Первая предназначена для подачи на выход логического нуля LOW (0 В) или логической единицы HIGH (5 В), а вторая - для задержки на указанное время (в миллисекундах).

При написании программного кода необходимо руководствоваться рядом правил, которые не являются обязательными при компиляции программы, но делают его более понятным и читаемым для других пользователей [9, 14].

Правило № 1

При написании строк в составе функций, следует применять отступ. Пример показан на рисунке 26.

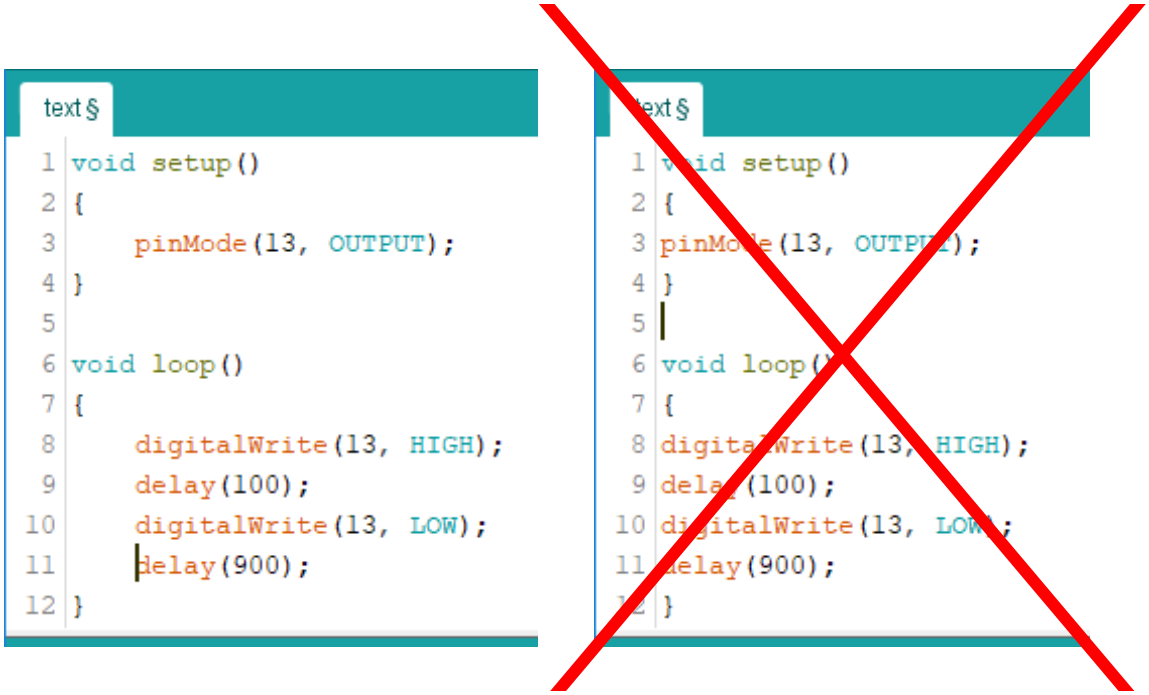


Рисунок 26 – Пример применения отступов при написании программы

Правило № 2

Следует ставить пробелы после знаков препинания и не применять до. Пример показан на рисунке 27.

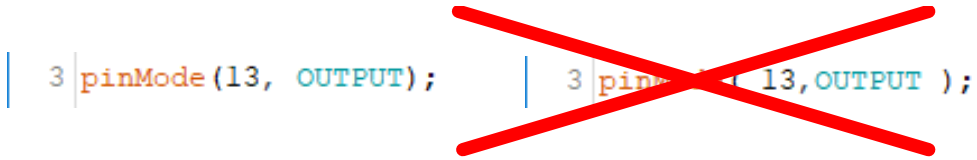


Рисунок 27 – Пример расстановки пробелов и знаков препинания

Правило № 3

Символ начала и конца блока функции следует размещать на отдельных строчках. Пример показан на рисунке 28.

Правило № 4

Для различных по смыслу частей блока функций используют разделительные строки. Пример показан на рисунке 28.

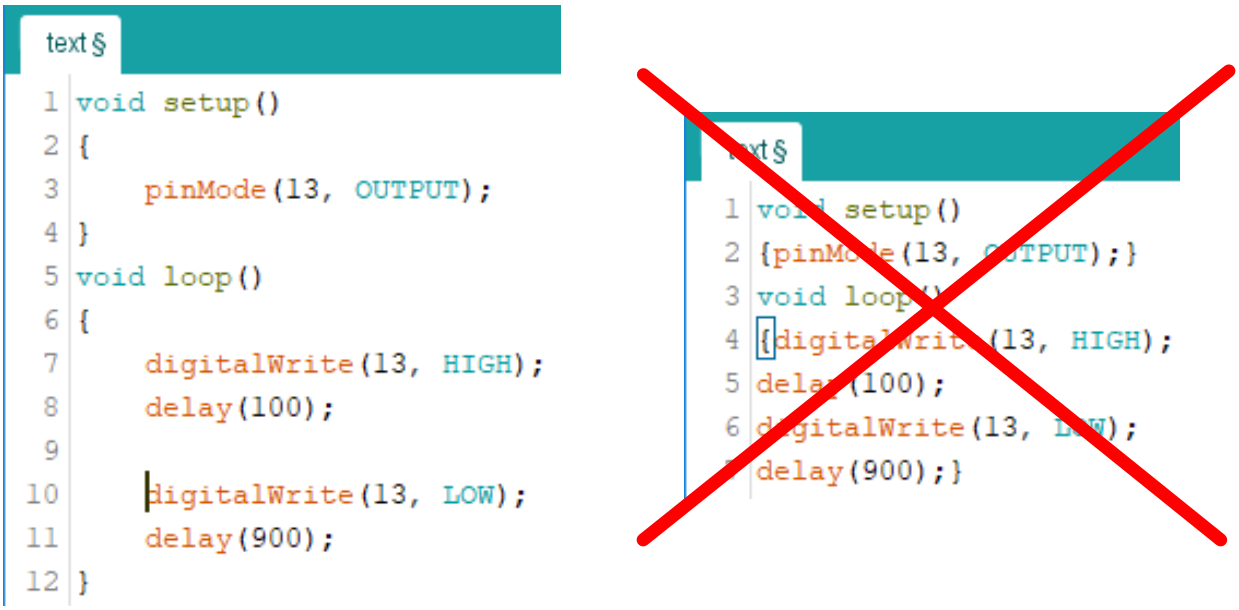


Рисунок 28 – Пример размещения символов начала и конца функции, разделительных строк

Правило № 5

Для разъяснения функционального предназначения отдельных блоков функций и их строк используются комментарии. Для их создания используются символы «/*» и «*/» , а также «//». Пример показан на рисунке 29.

```
    digitalWrite(13, HIGH);
    delay(100); // так пишется коментарий в строку

    digitalWrite(13, LOW);
    delay(900);
}
/*
 * так пишется коментарий
 * по тексту
 */
```

Рисунок 29 – Примеры использования комментариев

Правило № 6

С целью минимизации ошибок и для облегчения понимания в программном коде (особенно в длинном) следует применять макроопределения. Для их создания используется конструкция «#define Имя_константы Значение». Имена констант как правило пишутся с заглавных букв с применением «_» вместо пробела. Пример показан на рисунке 30.

```
#define PIN_LED 13
int timeOUT;
void setup()
{
    pinMode(PIN_LED, OUTPUT);
    timeOUT = 100;
}
 void loop()
{
    digitalWrite(PIN_LED, HIGH);
    delay(timeOUT);
    digitalWrite(PIN_LED, LOW);
```

Рисунок 30 – Пример использования макроопределения и объявления константы PIN_LED

Правило № 7

При определении в коде программы некоторых переменных, имена пишутся с маленькой буквы, однако если она состоит из нескольких слов, то каждое из них пишется слитно с заглавной.

4.2 Справочник языка Arduino

Для программирования Arduino существует множество операторов, функций и других составляющих. Большая и подробная библиотека доступна любому начинающему программисту - называется «Справочник языка Arduino» и находится в интернете по следующей ссылке: http://arduino.ru/Reference. Рассмотрим наиболее часто распространенные конструкции языка программирования, которые можно разделить на три больших группы: операторы, данные и функции [11, 12].

В области данных, их типов и преобразования практически нет отличий от C/C++. При этом существует несколько констант, которые часто используются при программировании Arduino:

– HIGH / LOW служат для считывания/задания значения на входе/выходе цифровых портов;

– INPUT / OUTPUT служат для задания режима порта входа/выхода.

При установке вывода в режим входа при помощи функции pinMode и считывания значения с использованием digitalRead, то результатом будет HIGH при напряжении от 3 В. Если после этого функцией digitalWrite будет записано значение HIGH, то к выводу будет подключен встроенный подтягивающий резистор (20 кОм), что даст постоянное считываемое значение HIGH при отсутствии электрического подключения куда-либо. Запись значения LOW отключает этот резистор. Функция digitalRead будет возвращать значение LOW при напряжении на входе меньше 2 В.

При установке вывода в режим передачи при помощи функции pinMode и записи значения HIGH функцией digitalWrite, на нём будет постоянное значение напряжения уровня 5 В. При записи значения LOW напряжение уменьшится до 0В.

При установке вывода на приём информации (INPUT) он будет иметь большое входное сопротивление (не менее 20 кОм). В режиме выхода (OUTPUT) вывод может управлять нагрузкой до 40 мА.

Основным разделом, включающим ключевые особенности среды программирования Arduino, являются функции. Синтаксис, назначение самых распространённых представлены в таблице 6. Перед именем функции, а также перед некоторыми аргументами записан тип возвращаемого значения:

- void – ничего не возвращает,
- byte - целое без знака значение в диапазоне 0 до 255;
- int – целое значение в диапазоне -32 768 до 32 767;
- unsigned int – целое без знака в диапазоне 0 до 65 535.
- long – целое в диапазоне -2 147 483 648 до 2 147 483 647;
- unsigned long – целое без знака в диапазоне 0 до 4 294 967 295.

Таблица 6 – Основные функции, используемые при программировании

Название	Описание	Синтаксис
1	2	3
pinMode()	Устанавливает режим работы заданного вывода как вход/выход	void pinMode(pin, mode), pin – номер пина, mode – значения INPUT, OUTPUT, INPUT_PULLUP (вход, выход, вход с подтягивающим резистором 20 кОм).
digitalWrite()	Подаёт значение HIGH/LOW на цифровой вход/выход	void digitalWrite(pin, value) pin – номер пина, value – значения HIGH/LOW
digitalRead()	Считывает значение HIGH/LOW с цифрового вход/выход	int digitalRead(pin)
analogRead()	Считывает значение с указанного аналогового порта. Напряжение от 0 – 5 В будет преобразовано в значение от 0 – 1023, с шагом равным 0,0049 В. Максимальная частота считывания 100 мс	int analogRead(pin)

Продолжение таблицы 6

1	2	3
analogReference()	Определяет значение напряжения относительно которого будут осуществляться аналоговые измерения.	void analogReference(type) type – определяет значение опорного напряжения: 1. DEFAULT (стандартное 5 или 3 В); 2. INTERNAL (встроенное 1,1 или 2,56 В); 3. EXTERNAL (внешний источник напряжения AREF)
analogWrite()	Выдаёт на порт входа/выхода ШИМ сигнал (частота 490 Гц, кроме 5 и 6 вывода – 980 Гц).	void analogWrite(pin, value) pin – номер пина, на который подаётся сигнал, value – период рабочего цикла (от 0 до 255).
tone()	Создаёт на входе/выходе сигнал прямоугольной формы с заданной частотой и регулируемой скважностью (длительность может определяться параметром). Используется для подключения пьезодинамика и применяется единовременно только на одном порту. На всех Arduino, кроме Mega данная функция работает только на 3 и 11 выводах.	void tone(pin, frequency) void tone(pin, frequency, duration) pin – номер пина, на котором будет создаваться сигнал, unsigned int frequency – частота сигнала (Гц) unsigned long duration – длительность сигнала (мс)
noTone()	Прекращает использование сигнала, создаваемого функцией tone()	void noTone(pin)
shiftOut()	Побитно выводит байт информации, затем генерирует импульс на синхронизирующем выводе для управления внешним устройством (последовательный протокол с синхронизацией).	void shiftOut(dataPin, clockPin, bitOrder, value) dataPin – номер пина, на который выводятся биты, clockPin – номер пина, по которому производится синхронизация, bitOrder – используемая последовательность вывода битов (MSBFIRST – старший первый или LSBFIRST – младший первый.) byte value – значение (байт) для вывода
pulseIn()	Считывает длину сигнала в микросекундах (от 10 мкс – 3 мин) на указанном порту. Если сигнал не фиксируется, то возвращается 0.	unsigned long pulseIn(pin, value) pulseIn(pin, value, timeout) pin – номер пина, на котором будет ожидаться сигнал, value – тип ожидаемого сигнала (HIGH/LOW), unsigned long timeout – время ожидания сигнала (мкс).

Продолжение таблицы 6

1	2	3
millis()	Определяет количество миллисекунд со старта программы (максимально 50 дней).	unsigned long millis()
micros()	Определяет количество микросекунд с момента выполнения текущей программы (максимально 70 мин).	unsigned long micros()
delay()	Останавливает выполнение программы на заданное количество времени.	void delay(ms) unsigned long ms – количество мс, на которое приостанавливается выполнение программы.
delayMicroseconds()	Останавливает выполнение программы на заданное количество времени в микросекундах (отличается от delay() тем, что работает со временем не превышающим 16 мсек).	void delayMicroseconds(mks) unsigned int mks – количество мкс, на которое приостанавливается выполнение программы.
randomSeed()	Создаёт одну и ту же, длинную последовательность случайных чисел	void randomSeed() void randomSeed(seed) unsigned long seed – параметр, определяющий начало последовательности.
random()	Возвращает случайное число.	long random(max) long random(min, max) long min – минимальное значение генерируемого числа, long max – максимальное значение генерируемого числа.
attachInterrupt()	Задаёт функцию, вызываемую внешним прерыванием.	void attachInterrupt(digitalPinToInterrupt(pin), ISR, mode); pin - номер пина, который будет использоваться в качестве внешнего прерывания (для UNO 2 и 3 вывод) ISR – имя функции, которая будет вызываться при прерывании (обработчик прерывания) mode – режим прерывания: 1. LOW – прерывание при напряжении на входе уровня логического нуля 2. CHANGE – прерывание при изменении напряжения на входе (увеличении или уменьшении) 3. RISING – прерывание по фронту сигнала на входе (при увеличении) 4. FALLING – прерывание по спаду сигнала (при уменьшении)

Продолжение таблицы 6

1	2	3
detachInterrupt()	Выключает обработку внешнего прерывания.	void detachInterrupt(interrupt) interrupt – номер прерывания (0 и 1, Mega: 0 – 5).
Serial.begin()	Определяет последовательное соединение и скорость передачи данных	Serial.begin(speed) speed – скорость (бит/с).
Serial.end()	Закрывает последовательное соединение, порты RX и TX освобождаются и могут быть использованы для ввода/вывода.	Serial.end()
Serial.print()	Передает данные через последовательный порт как ASCII текст.	Serial.print(val) Serial.print(val, format) val – данные для передачи через последовательное соединение format – базис для целых чисел или количество знаков после запятой для вещественных
Serial.println()	Текст передаётся с следующим за ним символом переноса строки и символом новой строки.	Аналогично Serial.print()
Serial.write()	Передаёт данные (один или серия байтов) как бинарный код через последовательное соединение.	Serial.write(val) Serial.write(str) Serial.write(buf, len) val – один байт, str – строка как серия байт, buf – массив данных, len – длинна массива.
Serial.peek()	Возвращает следующий доступный байт (символ) из буфера входящего последовательного соединения, не удаляя его из него.	Serial.peek()
Serial.available()	Функция получает количество байт(символов) доступных для чтения из последовательного интерфейса связи.	Serial.available()
Serial.read()	Считывает очередной доступный байт из буфера последовательного соединения.	Serial.read()

4.3 Библиотеки Arduino

Библиотеки в Arduino представляют собой набор файлов, содержащих готовые элементы кода программы, выполняющие определённые функции.

Программа Arduino IDE по умолчанию содержит базовый набор библиотек, которые можно запустить из меню (Скетч \ Подключить библиотеку \ *Название библиотеки*), как показано на рисунке 31.

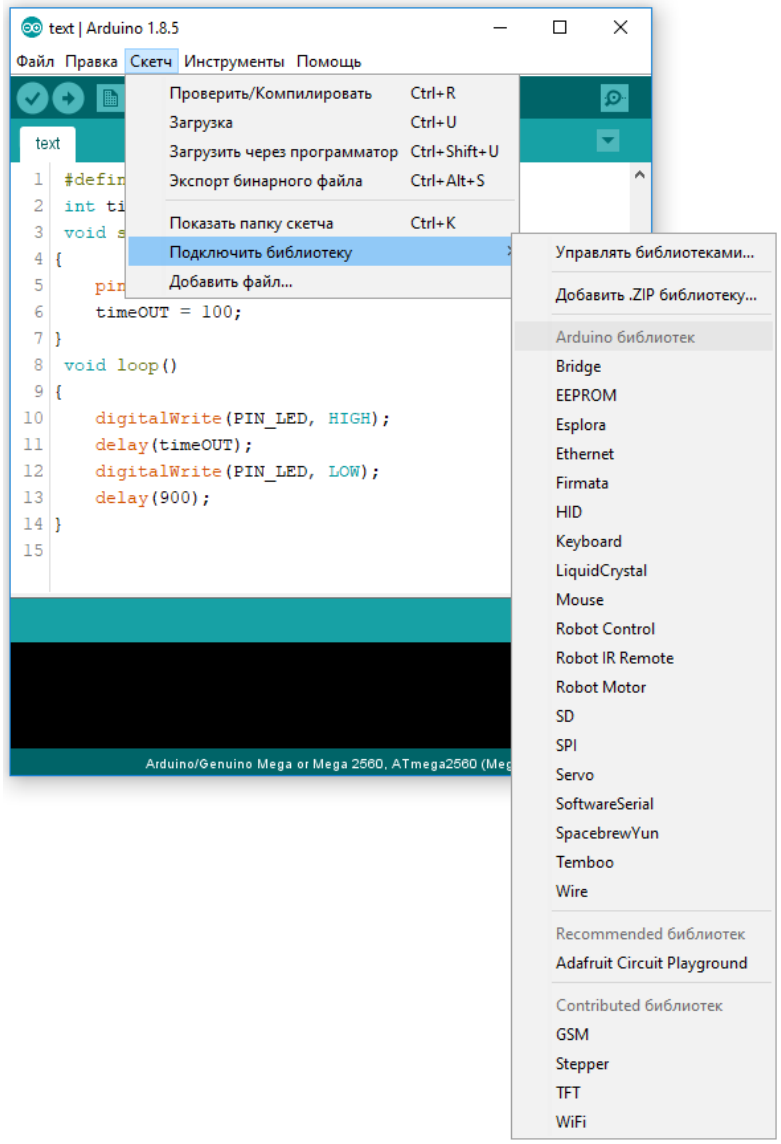


Рисунок 31 – Меню выбора стандартных библиотек в Arduino IDE

Перечень наиболее известных библиотек и их назначение приведены в таблице 7.

Таблица 7 – Библиотеки Arduino

Название	Описание
EEPROM	Работа с постоянной памятью микроконтроллера
Ethernet	Подключение к интернету через плату (шилд) Arduino Ethernet
Firmata	Использование стандартного (последовательного) порта для работы с приложениями
GSM	Для работы с шилдом соединения по GSM/GPRS
LiquidCrystal	Для работы с LCD дисплеями
SD	Для работы с картами памяти
Servo	Для работы с серводвигателями
SPI	Для работы с последовательным интерфейсом SPI
SoftwareSerial NewSoftSerial	Для работы с последовательным интерфейсом на цифровых выводах
Stepper	Для работы с шаговыми двигателями
TFT	Для работы с TFT дисплеями Arduino
WiFi	Для работы в интернете через шилд WiFi
Wire	Для работы с двухпроводным интерфейсом (TWI/I2C)
Messenger	Для работы с текстовыми сообщениями с компьютера
SerialControl	Для управления Arduino по последовательному интерфейсу
Capacitive Sensing	Использование портов в качестве ёмкостных датчиков
Debounce	Для считывания зашумлённых сигналов с цифровых портов
LedControl	Для работы со светодиодными матрицами или семисегментыми индикаторами
Matrix	Для работы с матрицей светодиодов
Sprite	Для работы с спрайтами и анимацией светодиодных матриц
FFT	Частотный анализ аналоговых сигналов
DateTime	Для отслеживания текущей даты и времени
Metro	Для выполнения определённых действий через равные промежутки времени

После добавления библиотеки в коде программы появится строчка #include <Название.h>. Можно просто прописать её и не пользоваться запуском через меню.

Для добавления дополнительной библиотеки необходимо:

1. Скачать требуемые файлы в *.zip;

2. Скетч \ Подключить библиотеку \ Добавить .ZIP библиотеку…

3. Найти требуемый архив и нажать Open;

4. Скетч \ Подключить библиотеку \ Название библиотеки.

Большая база дополнительных библиотек размещена на сайте https://github.com/ (требуется регистрация).

Для разработки собственной библиотеки необходимо создавать два файла: заголовочный (*.h) и файл с кодом (*.cpp).

Вопросы по разделу:

1. Какие части программы в среде программирования Arduino называют определением функции?

2. Как работают функции loop и setup?

3. Как работает функция pinMode?

4. Перечислите правила оформления программного кода для Arduino?

5. Как работают функции digitalWrite и delay?

6. Какие константы наиболее распространены в среде программирования Arduino?

7. Как программно включаются подтягивающие резисторы на выводах Arduino?

8. Какие функции Arduino вы знаете? (опишите не менее пяти)

9. Что понимают под библиотеками Arduino?

10. Как установить библиотеку и использовать её в проекте?

5 Практические задания

В этом разделе описано содержание практических работ по проектированию автоматизированных систем на базе микропроцессорной платы Arduino.

5.1 Работа с Fritzing

Рассмотренная ранее (раздел 3.3) последовательность работы с Arduino включает такой пункт, как сборка схемы системы, которой управляет контроллер. На этом этапе определяется структура дальнейшего кода программы (определится количество и тип используемых входов/выходов микроконтроллера, состав оборудования и библиотек к нему, логика работы и т.д.). Для её разработки можно воспользоваться листком бумаги с ручкой или собрать физически на имеющемся оборудовании, но такие варианты имеют много недостатков (принципиальная схема непонятна другому пользователю, не эргономична, не может быть сохранена для модернизации и реконструкции и т.д.). В данном случае идеальным вариантом будет применение специализированной программы, которая обладает необходимым функционалом и открытым кодом. Одним из самых распространённых приложений в этой области является Fritzing, которое можно скачать с сайта: http://fritzing.org/home/. Руководство по скачиванию представлено в виде схемы на рисунке 32 [14].

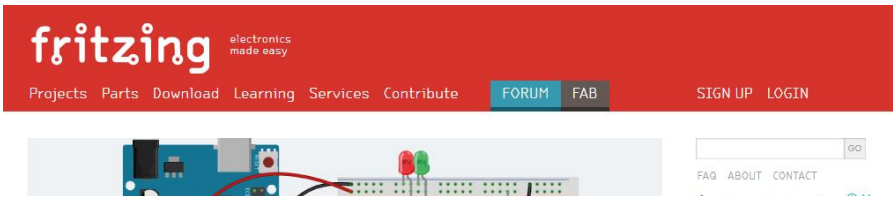
Для запуска программы необходимо:

1. Распаковать скаченный архив в удобное для запуска место.

2. Запустить файл Fritzing.exe.

Fritzing – это программное средство с открытым кодом, предназначенное для облегчения процесса прототипирования проектов на базе популярных микропроцессорных платформ (Arduino, Raspberry Pi и т.д.).

1. Перейдите по ссылке http://fritzing.org/home/

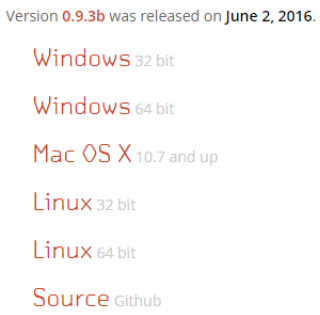


2. Сверху в меню (красный фон) нажать на раздел Download

3. Отметьте галочкой пункт «No Donation», если не планируете использовать программу в коммерческих целях

4. Нажмите кнопку «Download»

5. Нажмите на название ОС установленной на вашем ПК



6. Дождитесь полного скачивания архива (≈128 Мб)

Рисунок 32 – Схема скачивания Fritzing

В Fritzing содержится множество виртуальных моделей оборудования (Arduino, резисторы, транзисторы, переключатели и т.д.), которые можно разместить на виртуальной макетной плате и обозначить все соединения. Программа достаточно проста в использовании, а также позволяет разрабатывать макет платы для дальнейшего его изготовления. Её основные функции, их применение на практике, а также способы загрузки дополнительных элементов подробнее рассмотрены в примере выполнения самостоятельной работы.

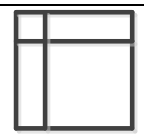
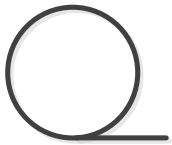
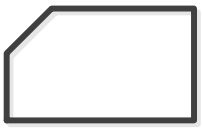
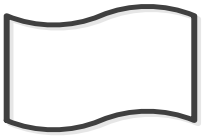
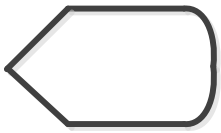
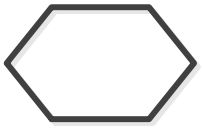
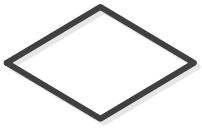
5.2 Составление алгоритмов программы в Visio

Перед тем, как приступить к написанию программного кода рекомендуется разработать алгоритм (последовательность действий) его работы. Графическим представлением в данном случае считается блок-схема. Основные правила по их составлению приведены в ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения» [7]. Аналогично составляются схемы данных, работы системы (например, отдельных частей программируемого робота) и взаимодействия программ (например, при соединении нескольких плат Arduino). В таблице 8 представлены символы, используемые при разработке блок-схем, их назначение и область применения.

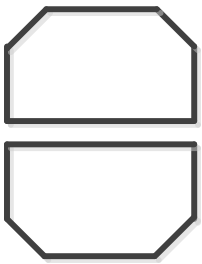
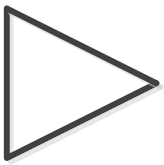
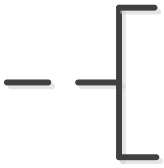
Таблица 8 – Символы блок-схем

Символ	Наименование	Схема данных	Схема программы	Схема работы системы	Схема взаим. программ
1	2	3	4	5	6
	Данные	+	+	+	+
	Запоминаемые данные	+	-	+	+

Продолжение таблицы 8

1	2	3	4	5	6
	Оперативное запоминающее устройство	+	-	+	+
	Запоминающее устройство с последовательной выборкой	+	-	+	+
	Запоминающее устройство с прямым доступом	+	-	+	+
	Документ	+	-	+	+
	Ручной ввод	+	-	+	+
	Карта	+	-	+	+
	Бумажная лента	+	-	+	+
	Дисплей	+	-	+	+
	Процесс	+	+	+	+
	Предопределённый процесс	-	+	+	+
	Ручная операция	+	-	+	+
	Подготовка	+	+	+	+
	Решение	-	+	+	-

Продолжение таблицы 8

1	2	3	4	5	6
	Параллельные действия	-	+	+	+
	Границы цикла	-	+	+	-
	Линия	+	+	+	+
	Передача управления	-	-	-	+
	Канал связи	+	-	+	+
	Пунктирная линия	+	+	+	+
	Соединитель	+	+	+	+
	Терминатор	+	+	+	-
	Комментарий	+	+	+	+

Инструментом для разработки блок-схем рекомендуется использовать функционал векторного графического редактора Microsoft Visio. При его запуске отображается окно, представленное на рисунке 33, в котором необходимо нажать на кнопку «Простая блок-схема».

Основное окно программы Microsoft Visio представлено на рисунке 34. В интерфейсе выделим четыре основные области.

В главном меню 1 расположено восемь вкладок (Файл, Главная, Вставка, Конструктор, Данные, Процесс, Рецензирования, Вид), смена которых меняет содержание панели инструментов 2. На ней расположены кнопки выбора

основных элементов, например, простейших фигур, как показано на рисунке 35а и изменения их параметров, как показано на рисунке 35б.

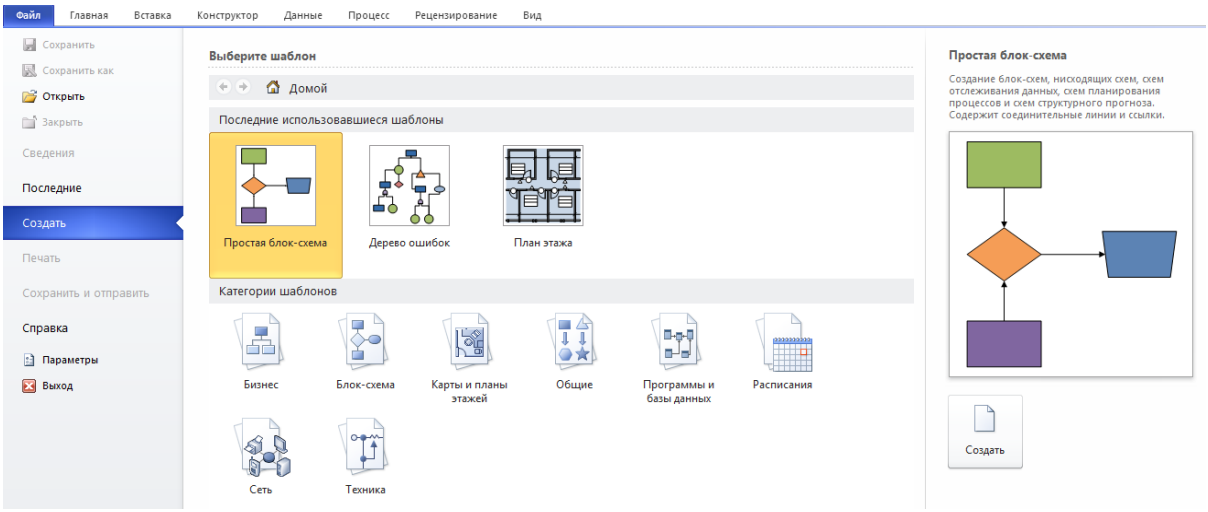


Рисунок 33 – Окно запуска векторного графического редактора MS Visio

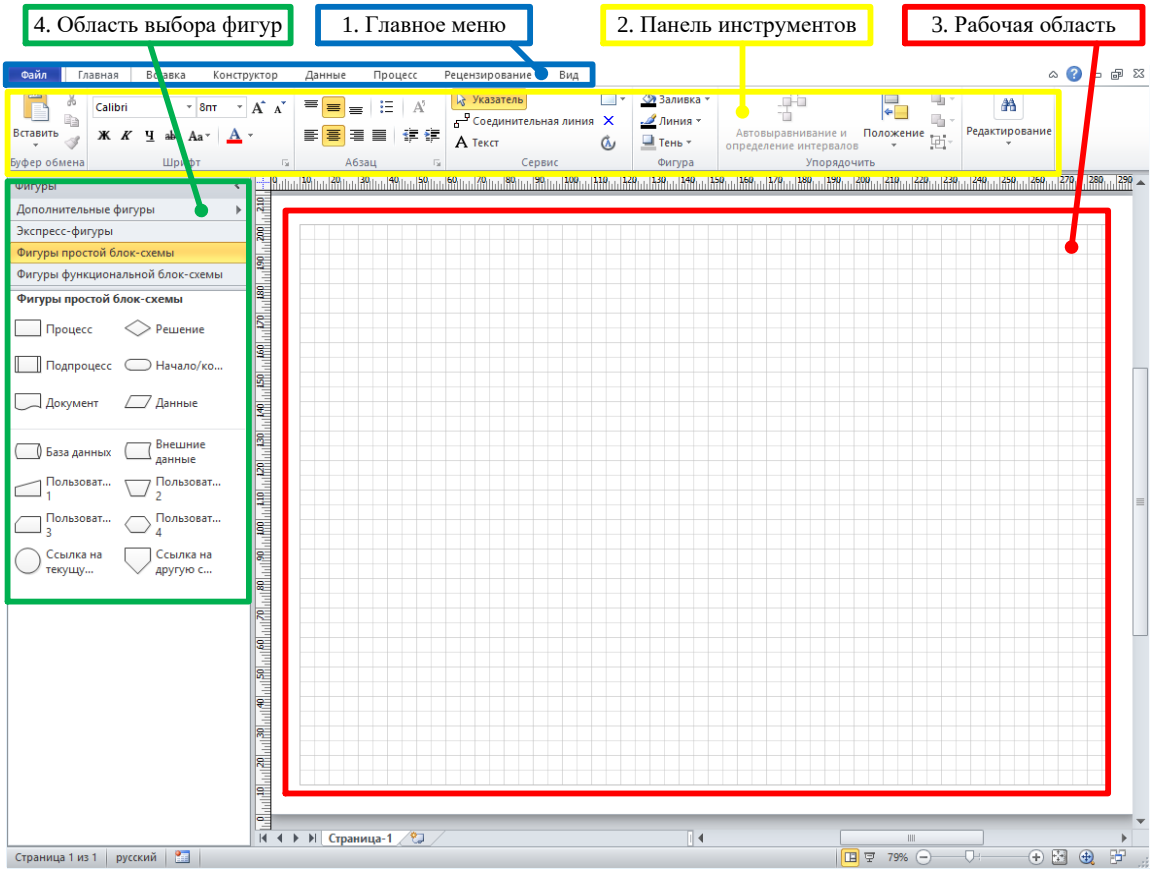
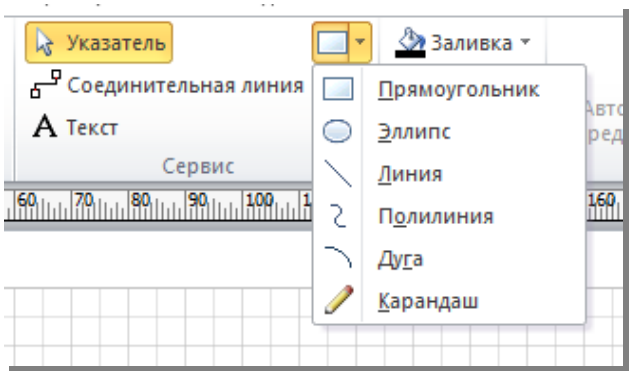
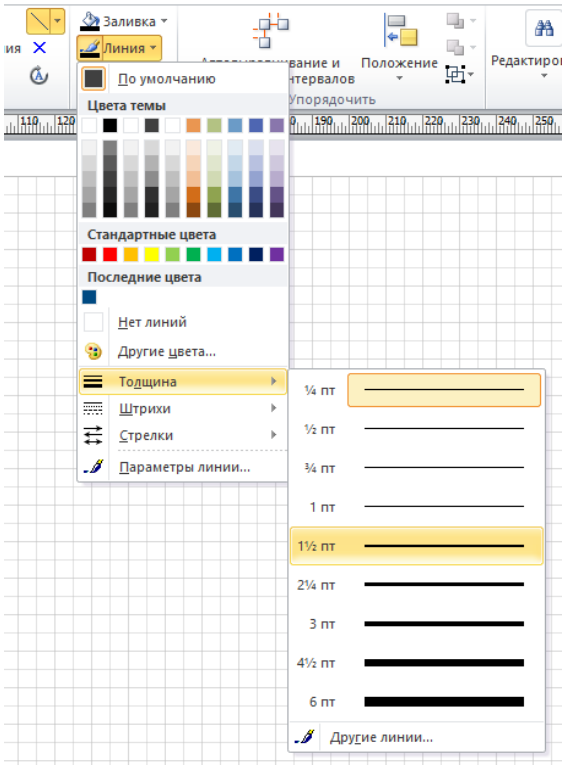


Рисунок 34 – Основное окно Microsoft Visio



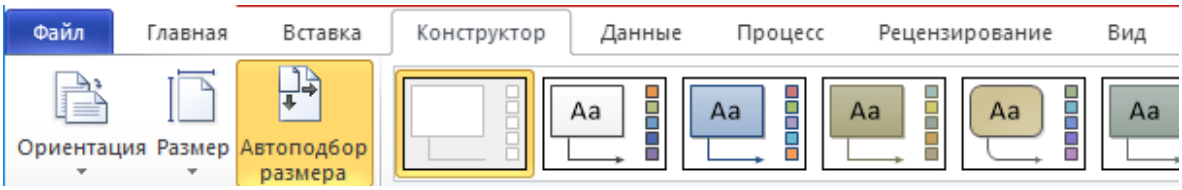
а)



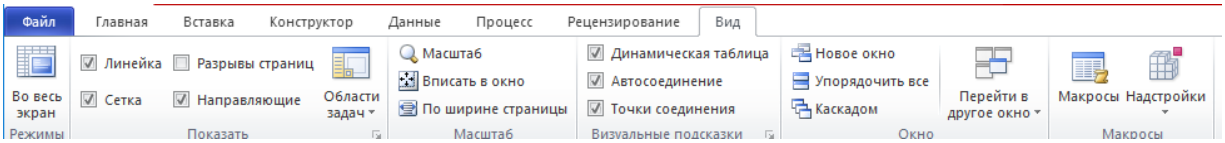
б)

Рисунок 35 – Элементы панели инструментов

Из вкладок главного меню необходимо обратить внимание на «Конструктор» и «Вид». В первой, как показано на рисунке 36а, устанавливается размер рабочей области 3 (рисунок 34), а настройки второй, представленные на рисунке 36б, необходимы для удобной работы в программе.



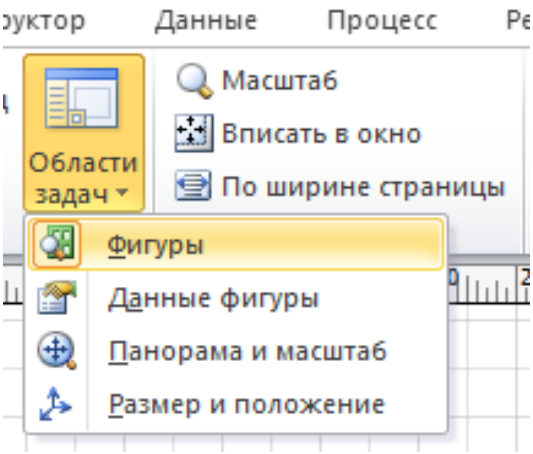
а)



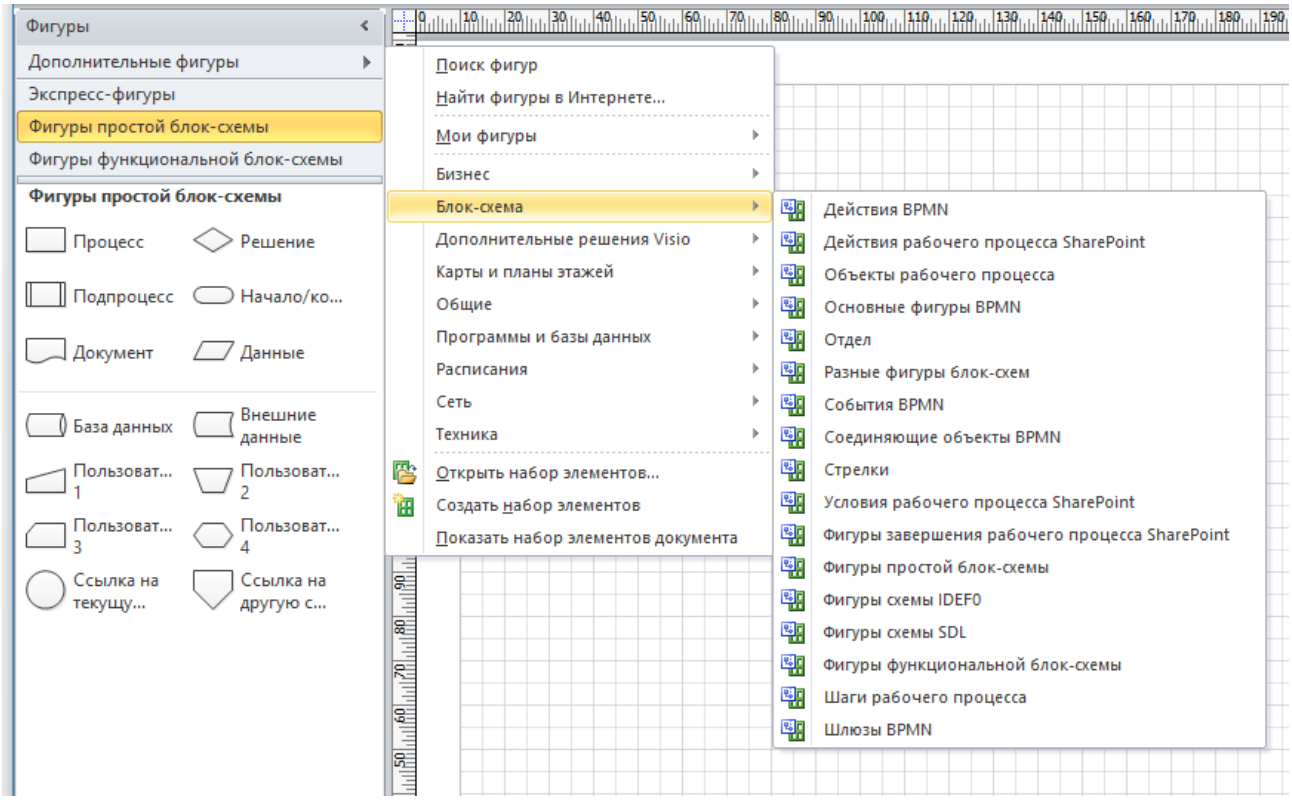
б)

Рисунок 36 – Дополнительные рабочие настройки Microsoft Visio

Область выбора фигур (запускается во вкладке главного меню «Вид», как показано на рисунке 37а), содержит библиотеки стандартных элементов графического редактора Microsoft Visio, одной из которых является «Блок-схема». Её выбор и основные разделы представлены на рисунке 37б.



а)



б)

Рисунок 37 – Выбор библиотеки «Блок-схема» и её разделов

Практически все элементы в таблице 8 имеются в представленной библиотеке. В случае, если не удаётся найти необходимый блок, то его можно нарисовать при помощи простейших элементов панели инструментов (линия, прямоугольник, круг и т.д.).

В рабочую область переносятся блоки из области 2 и 4, где из них комбинируется блок-схема. В случае, если блок составляется из элементов панели инструментов, то их необходимо сгруппировать, как показано на рисунке 38.

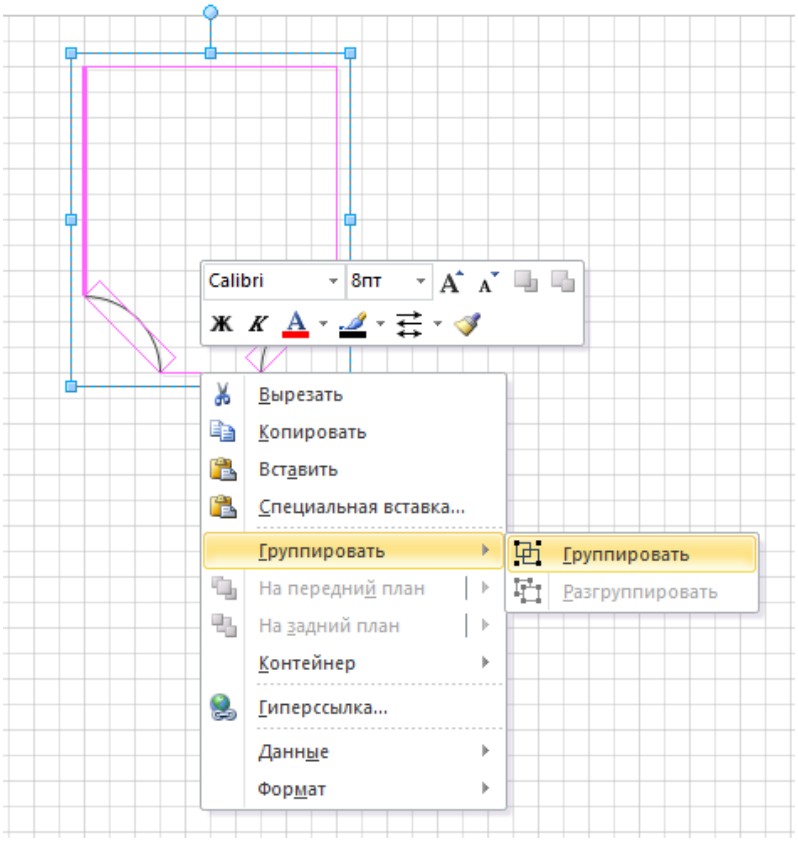


Рисунок 38 – Группировка простейших элементов в единый блок

Для сохранения готового проекта блок-схемы можно воспользоваться вкладкой «Файл» \ «Сохранить как» и выбрать путь и формат будущего файла или скопировать с помощью (Ctrl+C) выделенный элемент и вставить (Ctrl+V) в документ Microsoft Word. Во втором случае рисунок можно будет редактировать при открытии через контекстное меню.

5.3 Последовательность выполнения самостоятельных работ

Последовательность выполнения заданий относится ко всем последующим самостоятельным работам и должна быть полностью отражена в отчёте, который будет оцениваться преподавателем как итоговая форма контроля практических и лабораторных работ.

Вариант присваивается студенту индивидуально. Если задачи имеют дополнительные условия, то они определяются последней цифрой в зачётной книжке.

Последовательность выполнения работы:

1. Разработать аппаратную схему системы, описанной в условии задачи, согласно соответствующему варианту.

2. Разработать принципиальную схему проектируемой системы.

3. Составить и описать схему работы программного кода для проектируемой системы.

4. Разработать программный код в соответствии с составленным в п.3 алгоритмом. Он должен соответствовать правилам, описанным в разделе 4.1.

5. Собрать аппаратную схему на макетной плате.

6. Загрузить разработанный в п.4 программный код в микроконтроллер Arduino.

7. Отладить функционирование системы и продемонстрировать её работу преподавателю.

8. Составить развернутый вывод о проделанной работе.

Для выполнения пунктов 1 и 2 необходимо использовать программу Fritzing. В пункте 3 применяется векторный графический редактор Microsoft Visio. Разработка программного кода осуществляется в приложении Arduino IDE.

Отчёт по работе должен содержать результаты выполнения 1 – 4 и отражать 8 этапов представленной последовательности.

Задачи не содержат в условии вспомогательные элементы, влияющие на её работоспособность (например, в условии нет информации по подтягивающим резисторам и элементам питания).

Вопросы к разделу:

1. Опишите основные функциональные возможности Fritzing.

2. Как во Fritzing загрузить недостающие элементы для аппаратной схемы?

3. Как во Fritzing создать принципиальную схему проекта Arduino?

4. Для чего составляется алгоритм программного кода?

5. Какие основные блоки согласно ГОСТ 19.701-90 вам известны?

6. Какие основные рабочие области включает Visio?

7. Как запустить библиотеку MS Visio с элементами для разработки «Блок-схем»?

8. Как в MS Visio сохранить готовый проект алгоритма?

Самостоятельная работа № 1 «Схемы со светодиодами»

Оборудование: макетная плата, светодиоды, Arduino, соединительные провода, кнопки, потенциометры, резисторы.

Резистор – это сопротивление (Ом), которое позволяет ограничивать силу тока (А), переводя при этом часть энергии в тепло.

Любой резистор характеризуется тремя основными параметрами: номиналом (Ом), точностью (%) и допустимой мощностью (Ватт). Необходимо уметь их определять по маркировке (цифры или цветовые знаки). Примеры цифровой марки: 2,2 Ом - 2R2; 240 Ом - 240R; 510 Ом - K51; 7,5 кОм - 7K5; 15 кОм - 15K; 120 кОм - M12; 2,2 МОм - 2M2.

Цветовая маркировка резисторов приведена на рисунке 39.

Светодиод – полупроводниковый прибор, который излучает свет при пропускании через него в прямом направлении (от анода к катоду) электрического тока. Пример вида сверху и слева представлен на рисунке 40 (а и б), а условно-графическое изображение (УГО) на схеме (в).

Наиболее часто используются в проектах с Arduino светодиоды с диаметром линзы 3 и 5 мм. Вывод анода длиннее, а корпус линзы со стороны катода подрезан.

У двух и трёхцветных светодиодов имеется соответственно три и четыре вывода, один из которых катод, а остальные аноды.

Существуют также варианты светодиодов для поверхностного монтажа, как на плате Arduino Uno, который рассматривался в разделе 3.3.

При подключении светодиодов к выводам микроконтроллера последовательно с ним устанавливают токоограничивающий резистор.

Кнопка – простейшее электромеханическое устройство (коммутатор цепи).

Они бывают двух типов: переключатели (с двумя фиксированными положениями) и тактовые (после нажатия возвращаются в исходное положение).

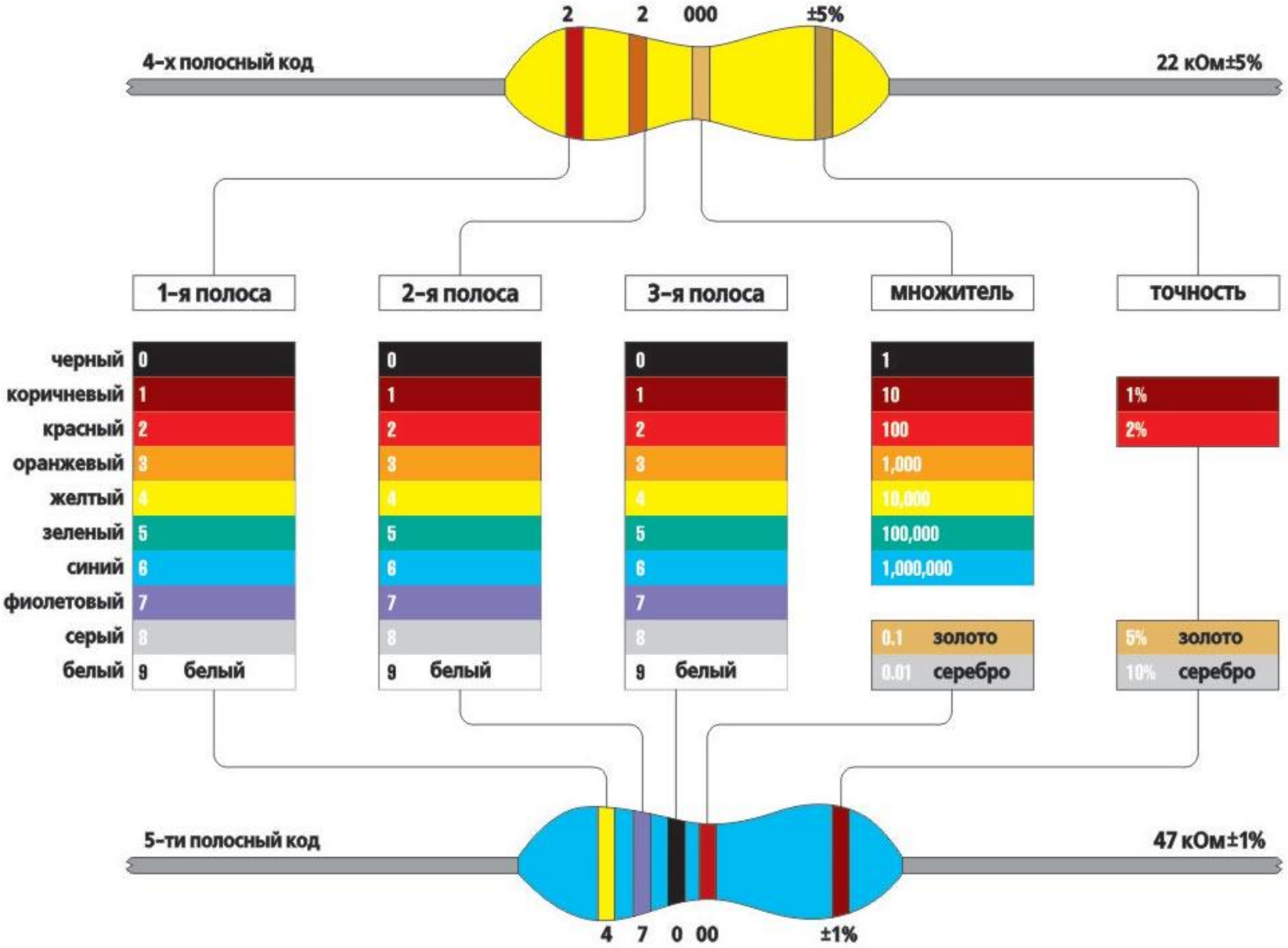


Рисунок 39 – Цветная маркировка резисторов

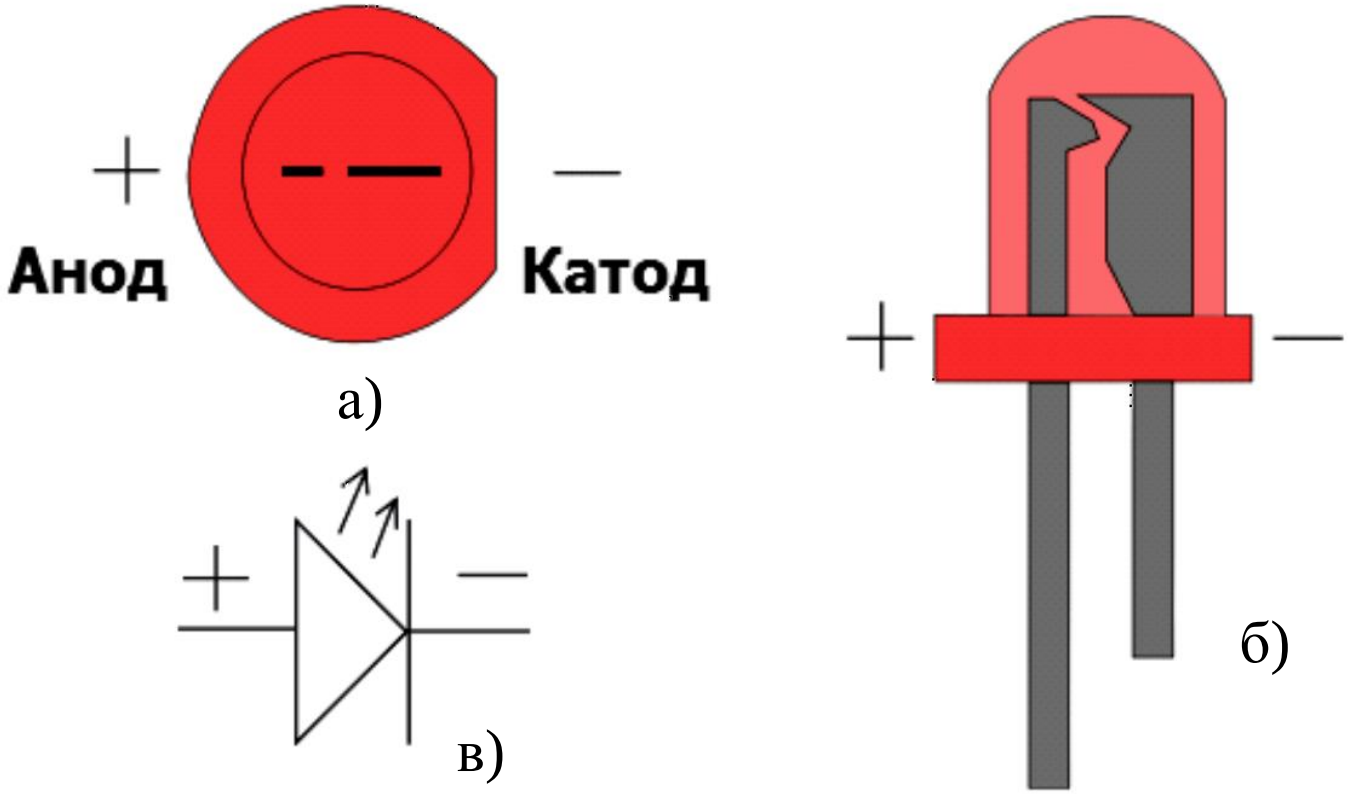


Рисунок 40 – Вид светодиода сверху (а), сбоку (б) и его изображение на схемах (в)

Подключение кнопок к выводу микроконтроллера без подтягивающих резисторов не рекомендуется, так как при разомкнутом контакте вследствие помех может «наводится» неопределенный уровень напряжения.

Подтягивающие резисторы подключаются к плюсу питания (верхние, pull-up) или к общему проводу (нижние, pull-down). В Arduino pull-up резисторы встроены, активируются программно и имеют номинал 20 кОм. В этом случае кнопку подключают одним выводом к микроконтроллеру, другим на общий провод (земля). Если кнопка имеет нормально разомкнутый контакт, то при нажатии на входе микроконтроллера сигнал изменяется с высокого HIGH на низкий LOW уровни [16].

Делитель напряжения – это два резистора, которые включены последовательно между собой, как показано на рисунке 41.

Напряжение на выходе рассчитывается:

$$U_{вых} = U_{вх} \cdot \frac{R_2}{(R_1 + R_2)}. \quad (1)$$

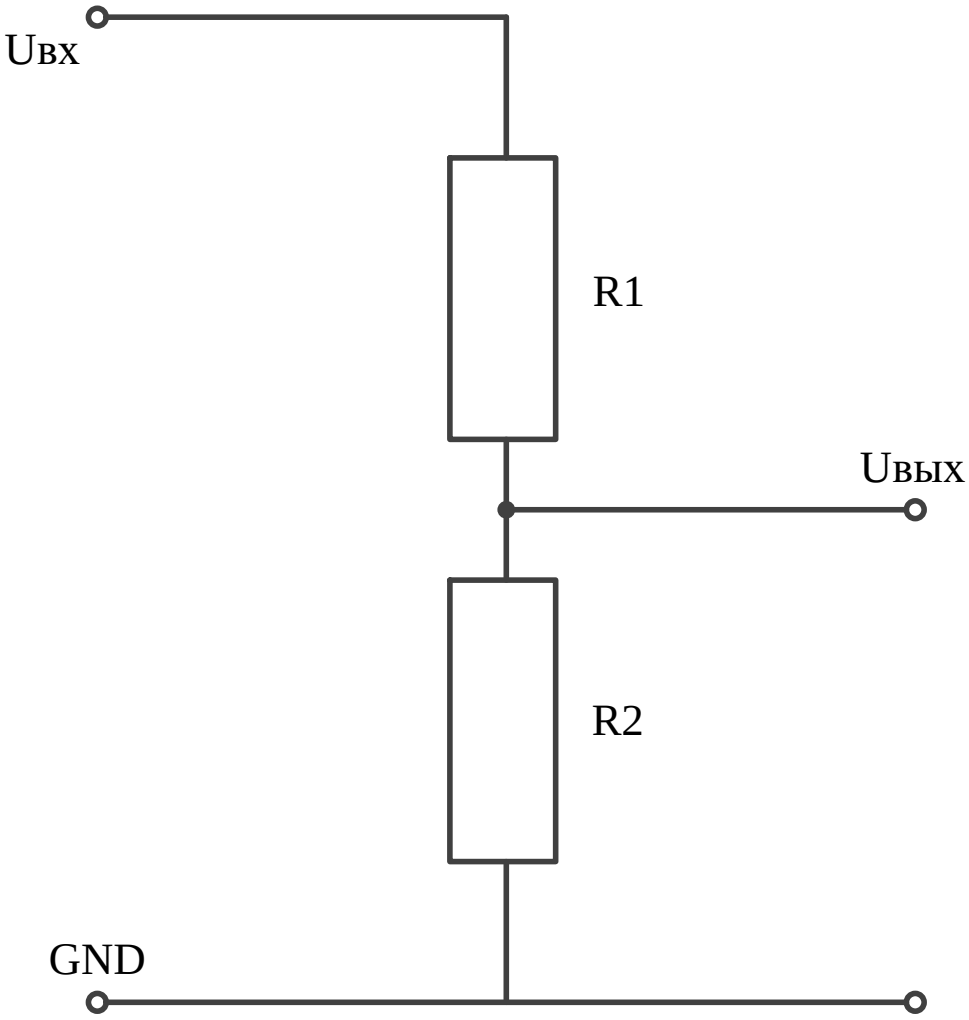


Рисунок 41 – Схема делителя напряжения

Для того, чтобы не нагружать источник измеряемого сигнала, рекомендуется выбирать номиналы сопротивлений в пределах 4,7 кОм – 50 кОм.

Потенциометр – переменный резистор, с регулируемым значением сопротивления.

Принцип действия аналогичен делителю напряжения, в котором величина сопротивления изменяется перемещением ручки. Устройство имеет три вывода (контакта): вход, выход и землю соответственно.

Макетная плата – это устройство для временного монтажа элементов электрической схемы.

Их наиболее распространённые модели представлены на рисунке 42.

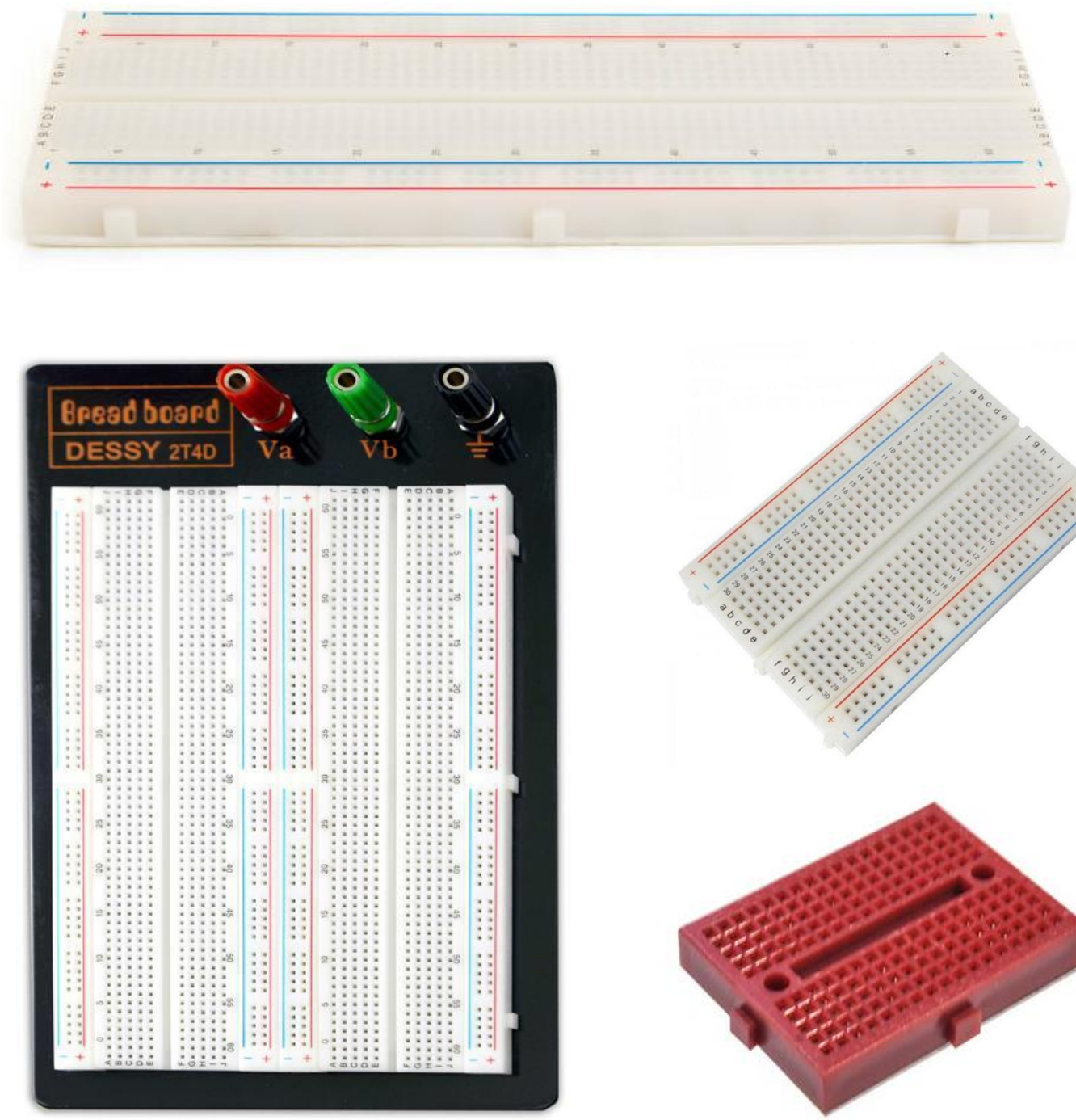


Рисунок 42 – Виды макетных плат

Для правильного подключения элементов на плате необходимо представлять, как конструктивно соединены контакты внутри, что и изображено на рисунке 43.

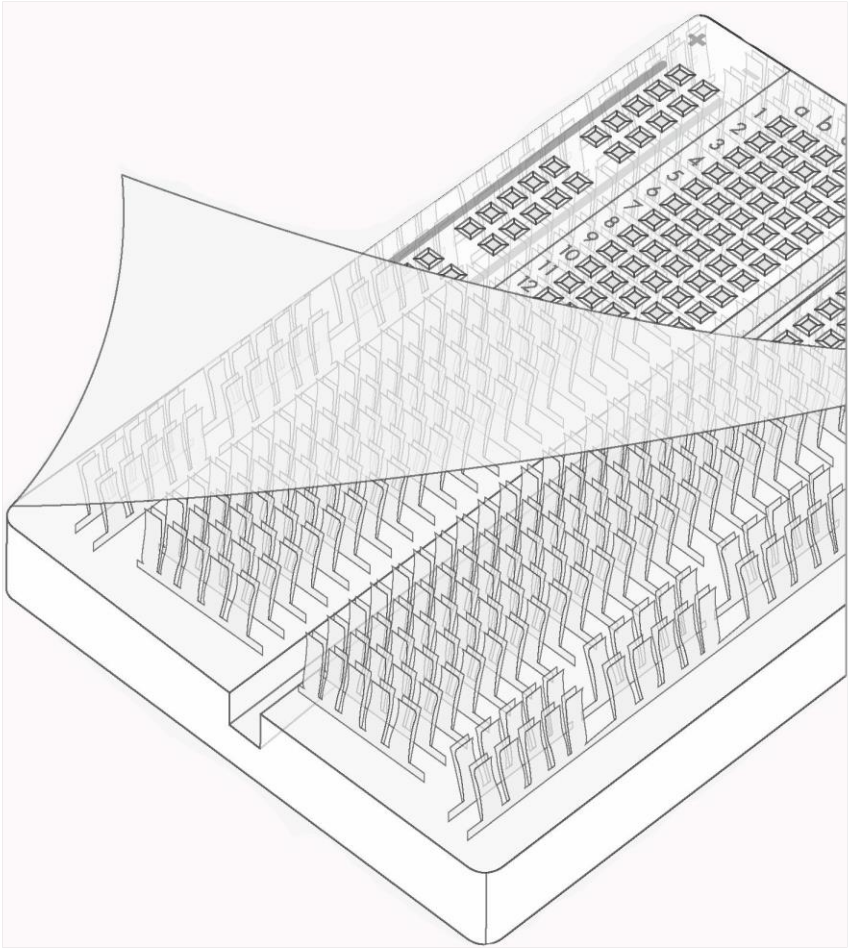


Рисунок 43 – Соединение контактов внутри макетной платы

Из рисунка 43 видно, что пластины центральной части устройства состоят из пяти совмещённых по горизонтали контактов (на внешней стороне отверстий). А по краям находятся две вертикальные сплошные пластины, предназначенные для подвода питания. Например, для сборки последовательного включения двух элементов необходимо разместить их на макетной плате таким образом, чтобы не было коротких замыканий и два контакта были расположены на одной пластине (воткнуты в две из пяти отверстий одного ряда).

Задания для самостоятельной работы №1

Пример составления отчёта представлен далее, варианты задач по первой самостоятельной работе указаны в таблице 9.

Таблица 9 – Варианты заданий для самостоятельной работы № 1

Вариант	0	1	2	3	4	5	6	7	8	9
Задачи	1,6	2,7	3,8	4,9	5,10	1,11	2,12	3,13	4,14	5,15
Здание повышенной сложности	16, 17, 18									

Задача № 1

Схема состоит из одного светодиода. Необходимо реализовать его непрерывное мигание сначала с аналогового, затем с цифрового выхода Arduino. Времена включения и паузы выбираются самостоятельно.

Задача № 2

Схема состоит из одного светодиода и тактовой кнопки. Необходимо реализовать его свечение при первом нажатии и выключение при повторном.

Задача № 3

Схема состоит из одного светодиода и потенциометра. Необходимо реализовать регулирование яркости светодиода.

Задача № 4

Схема состоит из одного трёхцветного светодиода. Необходимо реализовать последовательное зажигание красного, зеленого и синего цветов. Времена включения и паузы выбираются самостоятельно.

Задача № 5

Схема состоит из одного трёхцветного светодиода и тактовой кнопки. Необходимо реализовать плавное изменение красного, зеленого и синего цветов при нажатии кнопки.

Задача № 6

Светодиод должен зажечься три раза с периодом 0,5 с, затем три раза с периодом 1 с. Цикл должен повторится три раза. Таким образом, светодиод будет передавать сигнал SOS.

Задача № 7

Схема содержит 3 светодиода, потенциометр и тактовую кнопку. В исходном состоянии светодиоды не горят. При первом нажатии на кнопку они должны загораться максимально ярко, при повторном выключаться. Яркость регулируется потенциометром только при включении.

Задача № 8

Схема содержит 3 светодиода: зелёный, жёлтый и красный. Мигание должно осуществляться последовательно с заданными в соответствии с вариантом интервалами в секундах, указанном в таблице 10.

Задача № 9

Схема содержит N1 светодиодов одного цвета и N2 другого цвета. Мигание осуществляется группой поцветно с заданной в таблице 10 задержкой (паузой) и временем включения T в секундах.

Задача № 10

Схема содержит два светодиода и три тактовых кнопки. В исходном состоянии светодиоды не включены. При нажатии на первую кнопку они начинают мигать попеременно с временем включения T и заданной паузой. Вторая и третья кнопки меняют количество миганий первого N1 и второго N2 светодиода до переключения. Значения N1 и N2 указаны в таблице 10. Повторное нажатие на кнопку исполняемого режима должно возвращать систему в исходное состояние.

Задача № 11

Схема содержит ряд из шести светодиодов и двух тактовых кнопок. При нажатии на первую, светодиоды последовательно загораются и гаснут, образуя «бегущий огонь». При нажатии на вторую меняется направление «бегущего огня». Пауза и время включения задаются самостоятельно.

Задача № 12

Схема содержит один RGB-светодиод и кнопку. Необходимо организовать последовательное мигание синим, красным и зелёным цветами. Нажатие кнопки должно менять последовательность на обратную. Времена включения и паузы каждого из цветов представлены в таблице 10.

Задача № 13

Схема содержит один RGB-светодиод и кнопку. Необходимо организовать последовательное мигание желтым, голубым и фиолетовым цветами. Нажатие кнопки должно менять последовательность на обратную. Времена включения и паузы каждого из цветов выбираются самостоятельно.

Задача № 14

Схема состоит из одного RGB-светодиода и трёх обычных (синий, красный, зелёный). Необходимо, чтобы при последовательном изменении цветов первого включался один из трех соответствующего цвета. Времена включения и паузы выбираются самостоятельно.

Задача № 15

Схема состоит из трёх RGB-светодиодов и трех тактовых кнопок. В исходном состоянии светодиоды не включены. Первое нажатие любой из кнопок зажигает на соответствующем ей светодиоде красный цвет, второе - синий и третье - зелёный. Если на одном из трёх светодиодов уже включен один из цветов,

то два других его использовать не могут. Когда все цвета будут распределены, то через 5 секунд система возвращается в исходное состояние.

Задача № 16

Схема содержит один трехцветный светодиод. При помощи ШИМ необходимо реализовать плавное изменение цветов в следующей последовательности: красный, жёлтый, зелёный, голубой, синий, фиолетовый. Время переключения выбирается самостоятельно.

Задача № 17

Схема содержит один трехцветный светодиод и три потенциометра, подключённых к аналоговым входам. Программа должна непрерывно считывать значения сопротивления с потенциометров и менять интенсивность цвета соответствующей компоненты светодиода.

Таблица 10 – Дополнительные условия к задачам (предпоследняя цифра зачётки)

Вариант		0	1	2	3	4	5	6	7	8	9
Задача № 8, 13	Зеленый	1	2	3	2	1	2	2	2	1	2
	Желтый	2	1	2	1	2	1	2	3	2	1
	Красный	1	3	1	1	3	2	1	2	2	5
	Задержка	0,5	1	0,5	1	0,5	1	0,5	1	0,5	1
Задача № 9	N1	1	2	2	1	3	2	3	2	4	1
	N2	1	2	1	2	3	3	2	3	2	3
	T	1	2	3	1	2	3	1	2	3	4
	Задержка	1	0,5	1	0,5	1	0,5	1	0,5	1	0,5
Задача № 10	T	1	2	3	4	3	2	1	1	2	4
	Задержка	0,5	1	1,5	2	0,5	1	1,5	2	1	0,5
	N1	2	3	4	2	3	2	1	2	4	3
	N2	2	3	4	1	1	3	4	4	2	2

Пример выполнения работы

Рассмотрим простейший пример выполнения одной из задач самостоятельной работы и выделим содержание контрольного отчёта.

Задача № 0.1

Схема состоит из одного светодиода. Необходимо реализовать его непрерывное мигание. Времена включения и паузы выбираются самостоятельно.

Аппаратная схема системы

Для разработки аппаратной схемы запустим программу Fritzing (файл Fritzing.exe). Подробнее о её установке описано ранее в п 5.1. При первом запуске отобразится окно, представленное на рисунке 44.

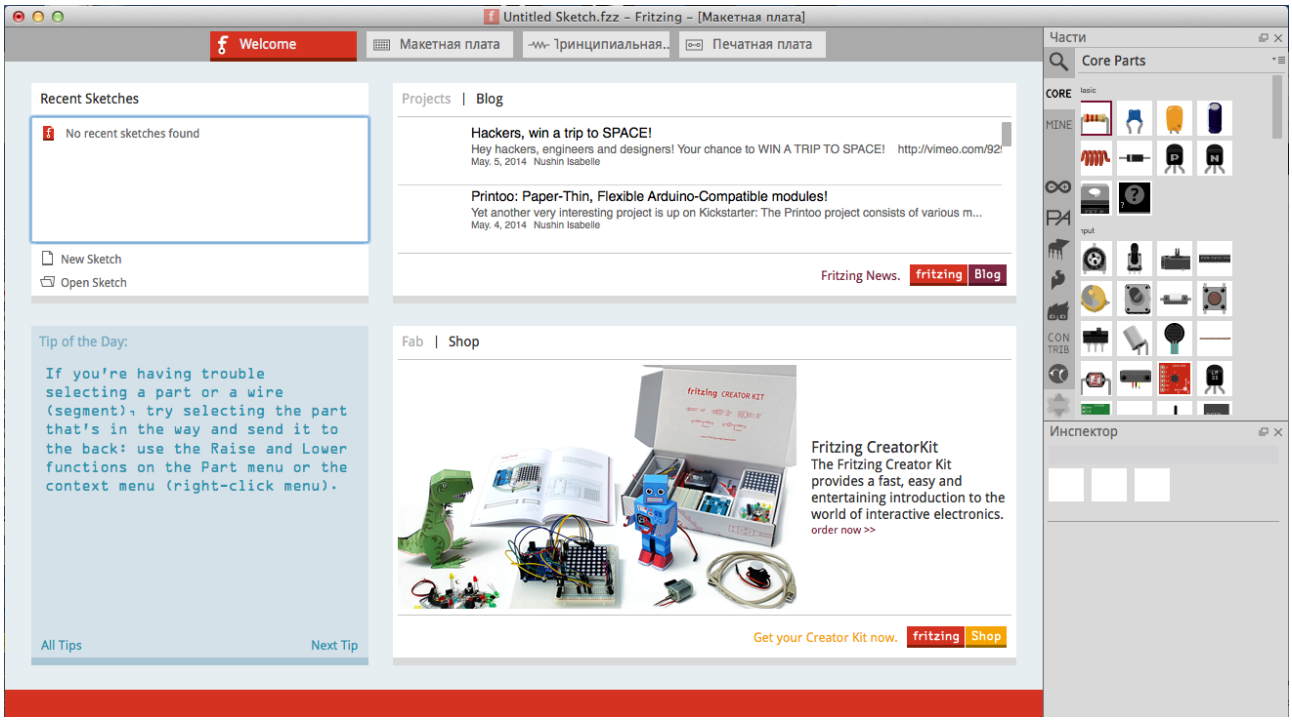


Рисунок 44 – Экран программы Fritzing при первом запуске

Для начала работы в окне, представленном на рисунке 44, необходимо нажать на кнопку «Макетная плата», которая располагается в верхней части. Рабочая область программы примет вид, как показано на рисунке 45.

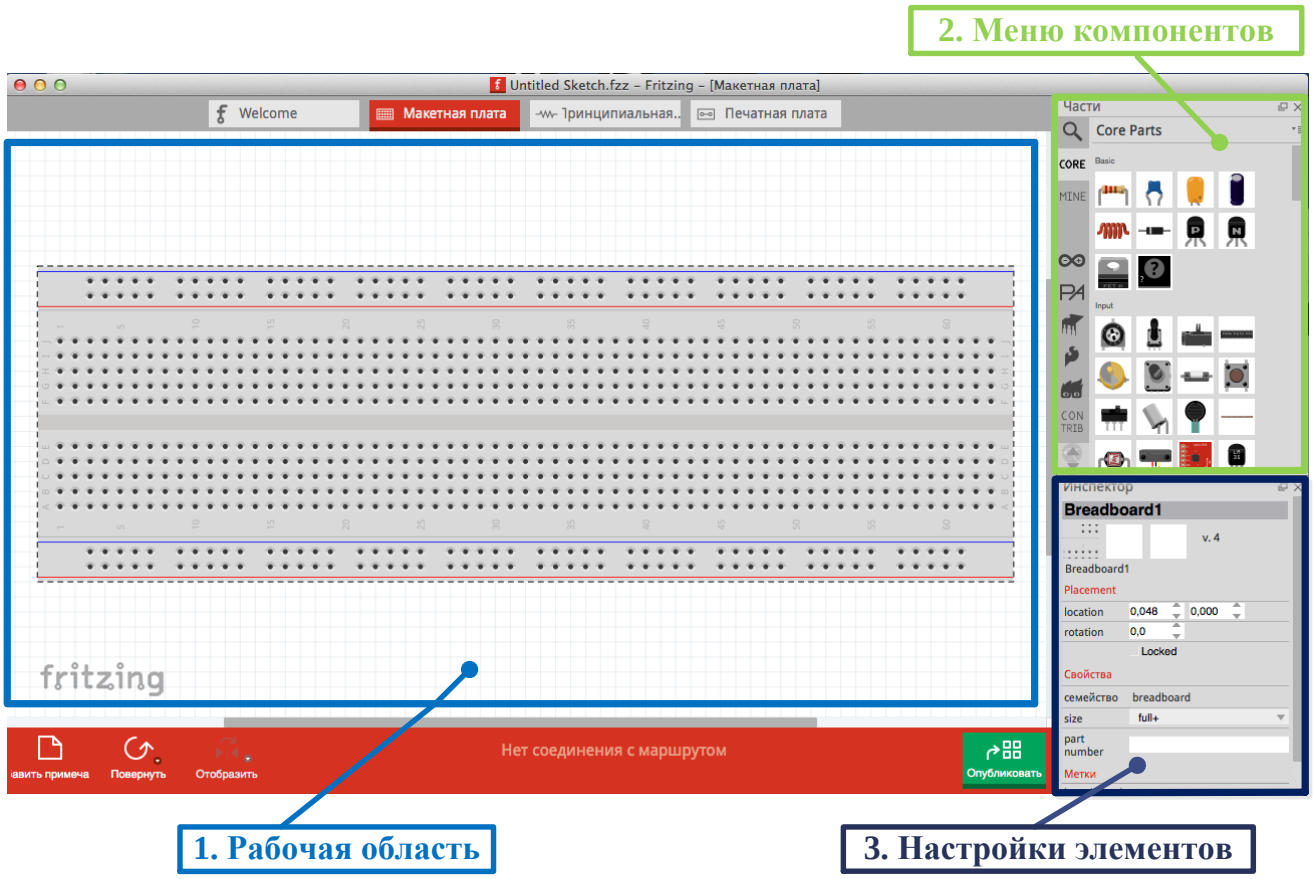


Рисунок 45 – Окно для сборки аппаратной схемы во Fritzing

Окно Fritzing для сборки аппаратных схем разделить на три основных части, как показано на рисунке 45:

1. Рабочая область содержит изображение макетной платы, на которой размещаются элементы проектируемой системы.

2. Меню компонентов содержит библиотеки (наборы) электрических устройств.

3. Настройки элементов содержат редактируемые параметры компонентов, которые будут перемещаться из меню 2 в рабочую область 1.

По условию задачи необходимо разместить на макетной плате и соединить следующие элементы: Arduino, резистор и светодиод. Для этого в меню 2 их необходимо найти и перенести в область 1, учитывая принцип её устройства, как показано на рисунке 46. При размещении элементов на макетной плате, ряд соединительных гнёзд выделяется зелёным.

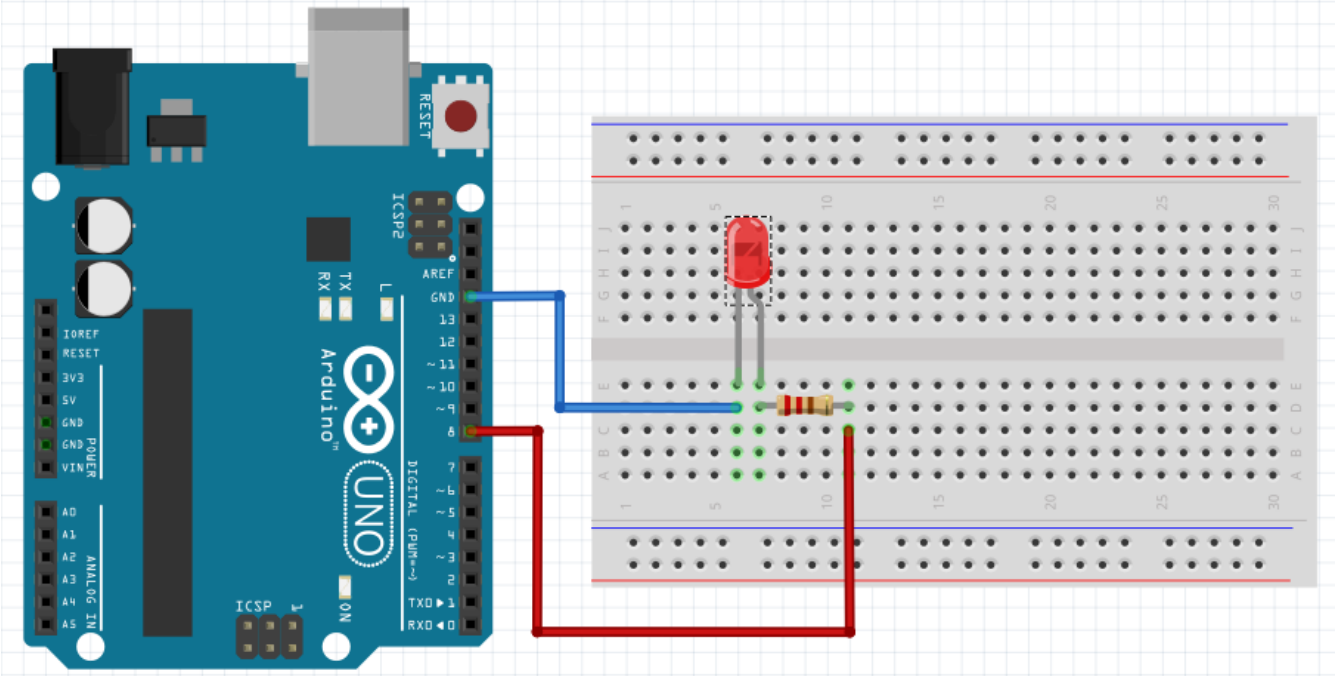


Рисунок 46 – Аппаратная схема для задачи № 0.1

Поворот элементов схемы осуществляется во всплывающем меню, которое вызывается нажатием правой кнопки по ним. Свойства объекта (цвет, координаты расположения, электрические параметры, размеры) настраиваются в меню 3. Для создания соединительного провода необходимо навести курсор мыши на отверстие макетной платы (выделится синим), зажать левую кнопку мыши и вести её в требуемую точку. Изгибы на проводах добавляются аналогично.

Таким образом, был выполнен первый пункт при решении задачи в самостоятельной работе (разработка аппаратной схемы). В отчёт переносится схема рисунка 46, а также её краткое описание.

Для сохранения схемы в формате «*.jpeg» необходимо нажать комбинацию клавиш «Alt+F», в появившемся меню перейти по вкладкам «Export\As

Image\JPG…» и выбрать место сохранения. Альтернативным вариантом возможно применение снимка экрана программы «Alt+Prt Scr» и его вставкой в документ Word комбинацией «Ctrl+V».

Если для проекта нет подходящих элементов в стандартных библиотеках, то необходимо их установить, предварительно загрузив с интернета, или взять у руководителя работы.

Для установки элементов необходимо нажать на меню компонентов 2 правой кнопкой мыши и в появившейся всплывающей панели выбрать «Import…». Затем указать путь к необходимому файлу с расширением «*.fzpz» («*.fzb», «*.fzbz») и он отобразится в программе.

Принципиальная схема системы

Fritzing при разработке аппаратной схемы автоматически составляет и принципиальную. Для её отображения в верхней области окна программы необходимо нажать на кнопку «Принципиальная схема», как показано на рисунке 47. От исполнителя работы требуется устранить недостатки построения (пересечения проводов, размещение элементов) и выбрать масштаб, как показано на рисунке 48.

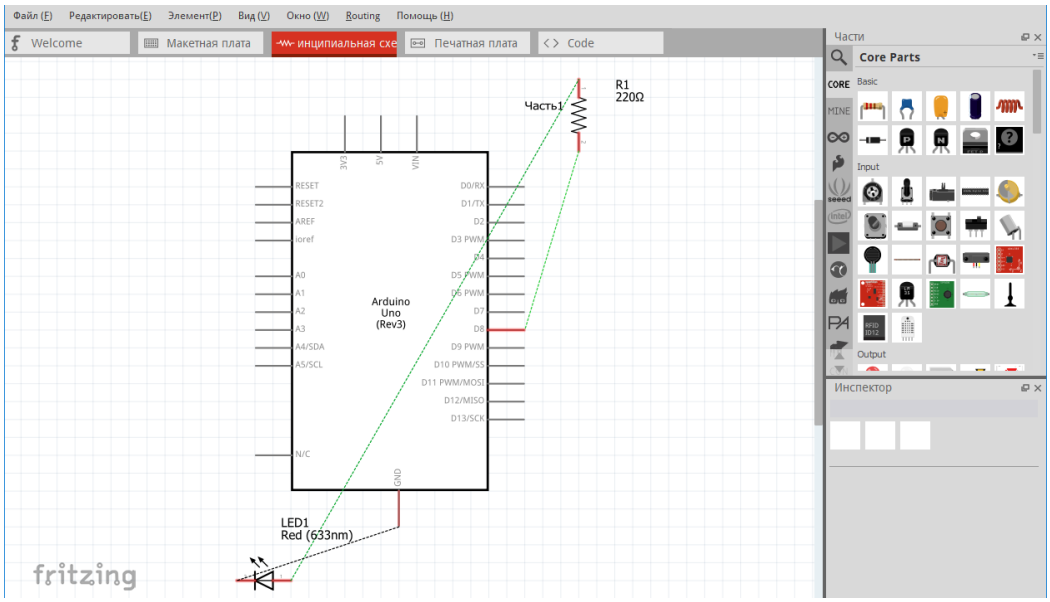


Рисунок 47 – Отображение принципиальной схемы во Fritzing

Автоматический подбор масштаба для схемы осуществляется нажатием комбинации клавиш «Ctrl + 0». Положения проводов и элементов меняется простым перемещением контрольных узлов с удерживанием левой кнопки мыши.

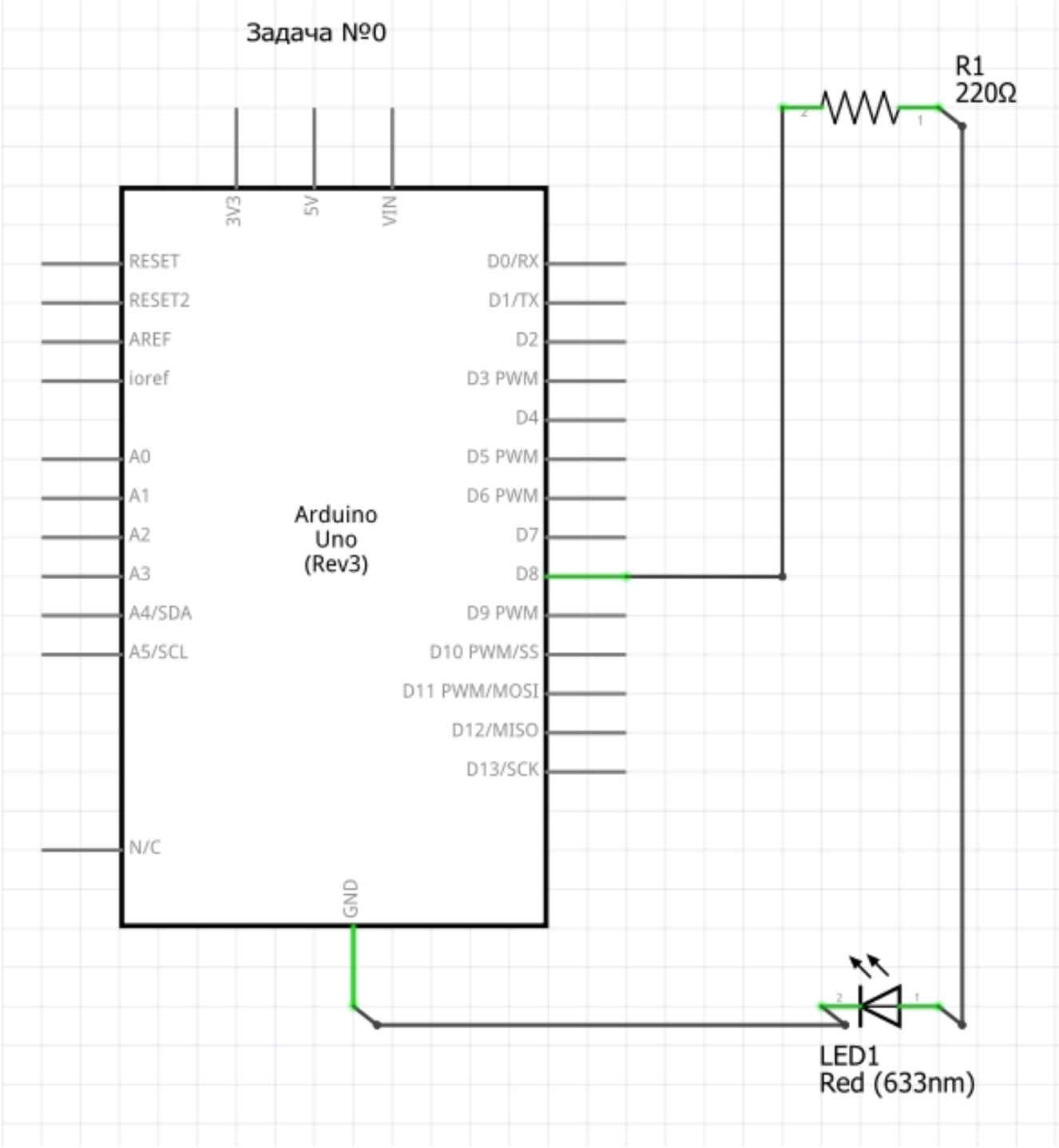


Рисунок 48 – Принципиальная схема для задачи № 0.1

В отчёт переносится только принципиальная схема.

Схема работы программного кода проектируемой системы

Подробная информация о том, как составляются схемы описано в разделе 5.2. Для написания программного кода, который удовлетворяет условиям рассмотренной задачи, построим алгоритм в графическом редакторе Microsoft Visio и представим его в общем виде на рисунке 49.

Рисунок 49 – Схема работы программы для задачи № 0.1

По заданию в рисунок могут быть добавлены комментарии. В отчёт вставляются визуальное представление алгоритма и краткие пояснения по составляющим его блокам loop() и setup().

Пример:

Представленную схему можно разделить на 2 основных этапа: объявление используемых переменных (целочисленных int) и портов (на выход OUTPUT); включение и выключение напряжения (HIGH / LOW) с задержкой.

Разработка программного кода

Разработка программного кода осуществляется в Arduino IDE, справочная информация по которой приведена в главе 4.

Для рассматриваемой работы программный код представлен на рисунке 50.

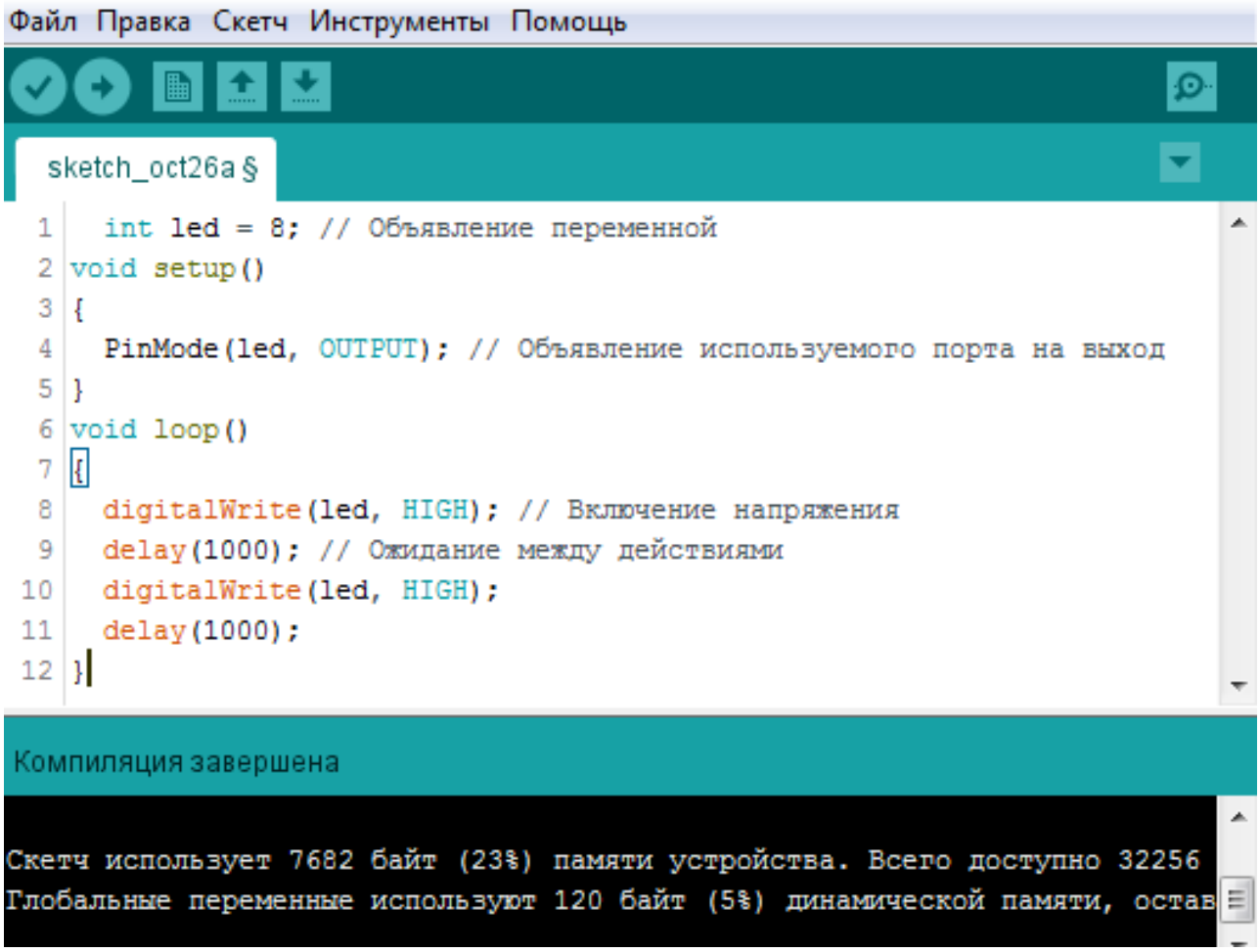


Рисунок 50 – Программный код для задачи № 0.1

В отчёт вставляется программный код в окне Arduino IDE. Обязательным условием является подтверждение на изображении успешной компиляции программы, как показано на рисунке 50.

Вывод:

Согласно условию поставленной задачи, была разработаны аппаратная и принципиальная схемы, алгоритм работы и программный код. Основной задачей разработанной системы является непрерывное мигание светодиода. Выбраны время включения 1 секунда и время паузы 1 секунда.

Перечень вопросов к самостоятельной работе № 1

1. Куда расходуется энергия на резисторе при протекании через него тока?

2. Какими параметрами характеризуется любой резистор?

3. Почему одни резисторы маркируются четырьмя цветными полосками, а другие пятью?

4. Каких цветов будут обозначения полосок в маркировке резисторов номиналом 100 Ом, 130 кОм, 560 кОм, 4,7 МОм, 10 МОм.

5. Определите номинал резисторов по следующим маркировкам: четыре красных полоски; оранжевый, жёлтый, зелёный, оранжевый, красный; серый, фиолетовый, красный, жёлтый, золотой; два красных, чёрный и оранжевый; чёрный, красный, жёлтый и коричневый.

6. В каком направлении должен протекать ток через светодиод?

7. Вывод какого контакта светодиода длиннее?

8. Какие выводы имеет RGB-светодиод?

9. Почему тактовые кнопки не подключают к выводам микроконтроллера без дополнительных цепей?

10. Что такое подтягивающий резистор? Какие они бывают и для чего используются?

11. Как подключить кнопку к Arduino с использованием встроенных подтягивающих резисторов?

12. Как устроен делитель напряжения?

13. Чем потенциометр отличается от реостата?

14. Чем отличаются макетные платы?

15. Как при помощи макетной платы соединить последовательно и параллельно два элемента?

Самостоятельная работа № 2 «Схемы со сдвиговым регистром и индикаторами»

Оборудование: макетная плата, светодиоды, Arduino, соединительные провода, резисторы, ёмкостные сопротивления, сдвиговый регистр, шкальный индикатор, сегментный индикатор, 4-х разрядный семисегментный дисплей.

Сдвиговый регистр – это устройство, позволяющее хранить и выводить параллельно несколько бит информации, загружаемых последовательно друг за другом с помощью специального тактового сигнала. В проектах Arduino сдвиговый регистр часто используется для упрощения работы (увеличения цифровых выводов) с сегментными индикаторами и большим количеством светодиодов.

Название сдвиговый обусловлено принципом его работы. При добавлении нового бита, все остальные перемещаются (сдвигаются) в сторону. Рассмотрим подробно диаграмму, представленную на рисунке 51. Согласно начальному условию регистр заполнен восьмью битами. Добавим в него столько же новых (11011010). После совершения первых двух тактов, в начало помещаются два новых младших бита, а два старших теряются. На последнем шаге весь регистров будет занят новыми данными. Регистры можно соединять последовательно, тогда вытесненные старшие биты будут перемещены в начало следующего [1-4, 10].

Модель рассматриваемого в пособии сдвигового регистра 74HC595 (КР156ИР53 отечественный аналог) имеет объём памяти 8 бит. Схема подключения представлена на рисунке 52.

На рисунке 52 обозначены:

- выходы ячеек памяти Q0-Q7;
- питание VCC (5 В)
- общий провод GND;
- линия приема последовательных данных DS;

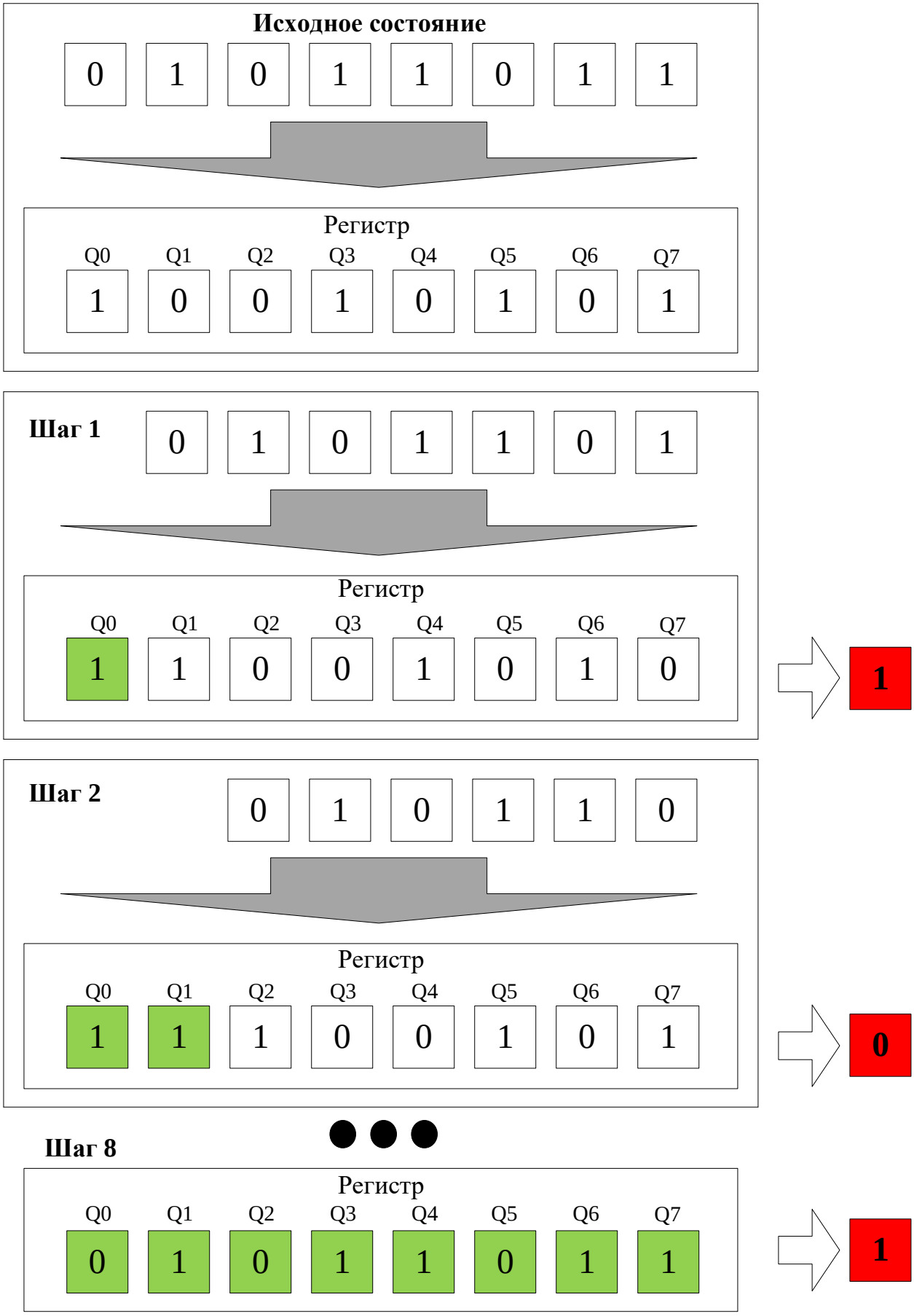


Рисунок 51 – Диаграмма работы сдвигового регистра

- импульсный сигнал ST_CP для передачи данных из внутреннего сдвигового регистра во внешний параллельный;

- тактовый сигнал SH_CP последовательной загрузки данных по входу DS;

- инверсный выход OE разрешения на вывод данных с внешнего параллельного регистра;

- вывод Q7’ предназначен для последовательного соединения со следующими регистрами.

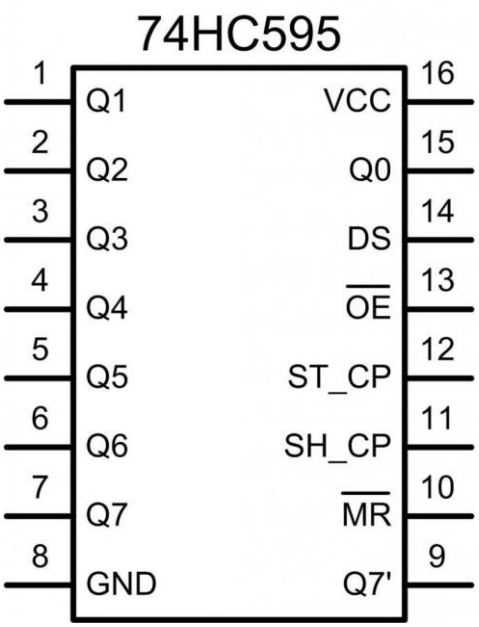


Рисунок 52 – Принципиальная схема сдвигового регистра 74HC595

Регистр очень чувствителен к импульсным помехам, поэтому питание должно быть зашунтировано конденсатором 0.1 мкФ, размещенным как можно ближе к соответствующим выводам микросхемы. В качестве нагрузки могут применяться светодиоды последовательно с резисторами (220 Ом).

Алгоритм работы с синхроимпульсами представлен на рисунке 53 в виде временных диаграмм. Загрузка последовательного внутреннего регистра осуществляется по фронту сигнала SH_CP. Информация со входа DS переписывается в самый младший бит, а остальные биты сдвигаются. Состояние выходов микросхемы в этот момент не меняется. После загрузки всех

необходимых битов загрузка данных из последовательного регистра в параллельный происходит по фронту импульса ST_CP, при этом изменение логических сигналов на выходе происходит одновременно.

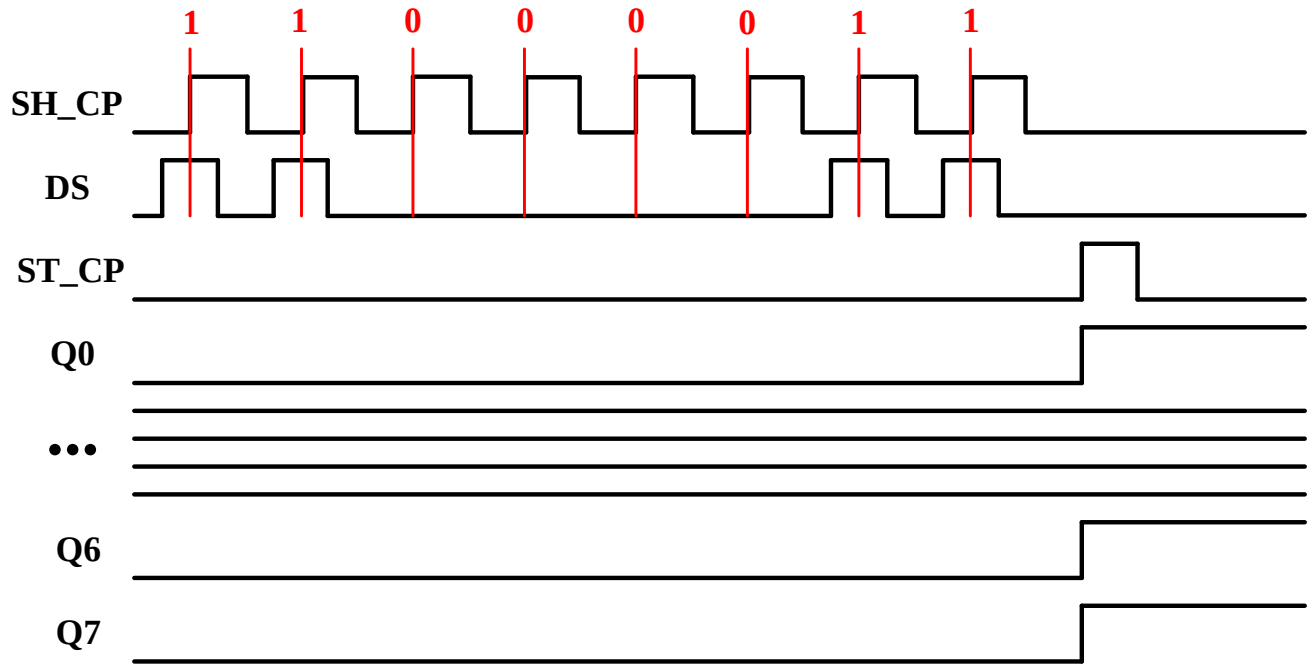


Рисунок 53 – Временные диаграммы работы синхроимпульсов

Порядок следования импульсов важен: на выводе DS устанавливается необходимый уровень сигнала, а затем подаётся импульс на SH_CP.

Сегментный индикатор – устройство вывода числовой информации на базе светодиодов.

Он состоит из набора светодиодов, расположенных так, что они вместе создают контуры цифр. Конструктивно сегментные индикаторы делятся на приборы с общим анодом или катодом: один из них соединён, а остальные выведены отдельно. Схема подключения представлена на рисунке 54.

Из рисунка видно, что все сегменты отмечены буквами латинского алфавита. Счёт ведут с верхнего по часовой стрелке, а точка обозначена как «dot». Общий катод или анод маркируется «com».

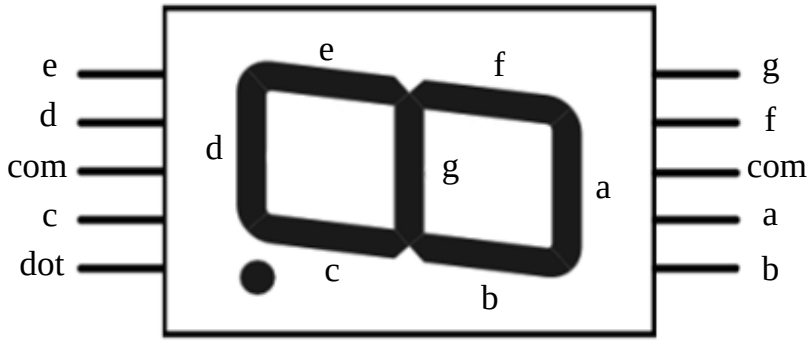


Рисунок 54 – Схема подключения сегментного индикатора

Достаточно распространённым альтернативным вариантом сегментному индикатору и сдвиговому регистру является LED индикатор TM1637, представленный на рисунке 55.

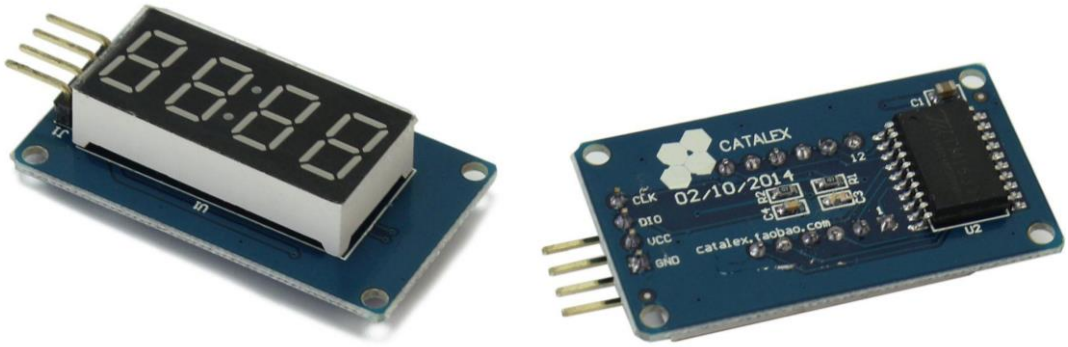


Рисунок 55 – LED индикатор TM1637

На плате представленного модуля расположен четырёхразрядный семисегментный дисплей на базе драйвера TM1637. Подключение осуществляется при помощи четырёх контактов: 5V/VCC, GND (питание), CLK и DIO (цифровой вход).

Существует библиотека (установка дополнительных библиотек описана в разделе 4.3) «tm1637.h», которую рекомендуется применять при написании программного кода, использующего данное устройство. Она позволяет реализовать следующие функции:

– выводить шестнадцатеричный знак в любое знакоместо;

– выводить сразу группу из 4-х шестнадцатеричных знаков;

– выводить десятичное целое число (от -999 до 9999);

– выводить десятичное число с плавающей точкой от (-999 до 9999);

– очищать дисплей;

– устанавливать яркость индикатора (от 0 до 7).

Задания к самостоятельной работе № 2

Варианты задач по второй самостоятельной работе указаны в таблице 11.

Таблица 11 – Варианты заданий для самостоятельной работы №2

Вариант	0	1	2	3	4	5	6	7	8	9
Задачи	1,6, 11	2,7, 12	3,8, 11	4,9, 12	5,10, 11	1,10, 12	2,9, 11	3,8, 12	4,7, 11	5,6, 12
Здание повышенной сложности	13, 14, 15									

В условиях всех задач предусмотрено использование сдвигового регистра модели 74HC595.

Задача № 1

Схема содержит восемь светодиодов (шкальный индикатор). Необходимо при помощи сдвигового регистра реализовать «бегущий огонь», используя функцию shiftOut ( ).

Задача № 2

Схема содержит восемь светодиодов (шкальный индикатор). Необходимо при помощи сдвигового регистра реализовать «бегущий огонь», используя функцию bitWrite ( ).

Задача № 3

Схема содержит восемь светодиодов (шкальный индикатор). Необходимо при помощи сдвигового регистра и ШИМ реализовать их одновременное плавное изменение свечения.

Задача № 4

Схема содержит сегментный индикатор, который подключён к микроконтроллеру через сдвиговый регистр. Необходимо, чтобы на нём последовательно отображались цифры от 0 до 9 с задержкой 2 секунды.

Задача № 5

Схема содержит потенциометр (подключён к аналоговому входу) и сегментный индикатор, который подключён к микроконтроллеру через сдвиговый регистр. Необходимо, чтобы на нём отображались цифры от 0 до 9 в зависимости от положения потенциометра.

Задача № 6

Написать программу, которая будет отображать на линейке светодиодов уровень напряжения на аналоговом входе, к которому подключен переменный резистор. При уровне 0 индикатор не светится. На уровне 512 светится 4 из 8 светодиодов. Все светодиоды светятся при уровне, равном 1023.

Задача № 7

Схема содержит LED индикатор TM1637 и тактовую кнопку. В исходном состоянии ничего не отображается. При первом нажатии выводится число 1234, при втором 4321, а при третьем индикатор выключается.

Задача № 8

Схема содержит LED индикатор TM1637. В исходном состоянии (при включении) на нём отображается число 0000. По истечении 10 секунд значение

меняется на L000, затем с интервалом 5 секунд на 0L00 / 00L0 / 000L и ещё через 10 секунд индикатор переходит в исходное состояние.

Задача № 9

Схема содержит LED индикатор TM1637 и RGB-светодиод. При её включении горит зелёный цвет и запускается таймер, который отображается на индикаторе. Когда значение времени достигает 10 секунд, цвет меняется на голубой, а через 30 секунд - на красный. Отсчёт останавливается и цифры начинают мигать в течение 10 секунд. Недостающие значения задержек задаются самостоятельно.

Задача № 10

Схема содержит LED индикатор TM1637 и две тактовые кнопки. В исходном состоянии отображается 0000. При первом нажатии на одну из кнопок запускается и отображается секундомер. Повторное нажатие его останавливает. Другая кнопка обнуляет значения.

Задача № 11

Соединить два регистра в цепочку, и подключить к ним два сегментных индикатора. Написать программу для Arduino, которая будет вести обратный отсчет от 99 до 0, с периодом 1 секунда.

Задача № 12

В схеме игры присутствуют две кнопки, и линейка светодиодов. В игре участвует два человека. Задача игрока – как можно чаще нажимать на свою кнопку. Чем чаще игрок нажимает кнопку, тем больше в его сторону перетягивается канат. Центр каната отображается горящим светодиодом на линейке. Чем больше превосходство одного игрока над другим, тем ближе включенный светодиод к соответствующему краю линейки. Игра заканчивается, когда активный светодиод достигает края линейки.

Задача № 13

Схема содержит четыре сегментных индикатора и сдвиговых регистра, индикатор TM1637, пять кнопок и RGB-светодиод. В исходном состоянии везде отображаются нули. Каждой кнопке соответствует соответствующий сегментный индикатор. При нажатии запускаются цикличные отсчёты от 0 до 9 с разной задержкой (определяется вариантом). Повторное нажатие переносит текущее значение на индикатор TM1637 (число переносится в такую же позицию, как и для семисегметного индикатора). Когда все четыре числа отображаются на LED индикаторе, то необходимо нажать на пятую кнопку. Если сумма двух первых чисел в левой части больше суммы двух правых, то светодиод должен загореться на 10 секунд зелёным цветом, если наоборот, то синим. В случае переноса не всех чисел и нажатии последней кнопки загорается красный цвет, и система переходит в исходное состояние. Значения задержек между изменениями чисел представлены в таблице 12.

Таблица 12 – Значения задержек между изменениями чисел в секундах

Вариант	0	1	2	3	4	5	6	7	8	9
1 индикатор	0,25	0,75	1,5	2,5	0,6	0,5	0,25	1,25	1,5	1
2 индикатор	0,5	0,25	2,5	1,5	0,5	0,25	0,5	0,25	1	0,75
3 индикатор	0,4	0,5	0,25	0,5	0,25	2,5	1	0,5	0,25	0,5
4 индикатор	1	1,1	0,5	0,25	0,15	4	2	1	0,5	0,25

Задача № 14

Схема содержит восемь светодиодов (шкальный индикатор). Необходимо при помощи сдвигового регистра и ШИМ реализовать различную яркость свечения в зависимости от варианта. Значения яркости по вариантам представлены в таблице 13.

Таблица 13 – Яркость в %

		Номера светодиодов слева направо									
Вариант		0	1	2	3	4	5	6	7	8	9
Яркость в %	5	1,2	–	7,8	–	7,8	–	–	3,4	–	7,8
	15	–	1,2	–	–	–	–	7,8	–	3,4	–
	30	7,8	–	1,2	5,6	5,6	5,6	–	7,8	–	3,4
	45	3,4	–	–	1,2	–	7,8	5,6	–	7,8	1,2
	60	–	3,4	5,6	7,8	1,2	3,4	–	5,6	1,2	–
	75	–	5,6	3,4	–	3,4	1,2	3,4	1,2	5,6	–
	100	5,6	7,8	–	3,4	–	–	1,2	–	–	5,6

Задача № 15

Схема содержит LED индикатор TM1637, два потенциометра и три тактовые кнопки. В исходном состоянии отображается 00.00. Один потенциометр должен менять две левые цифры от 00 до 99, аналогично второй – правые. Каждая из кнопок должна выполнять арифметические действия «+», «-» и «*» соответственно. Ответ выводится через 3 секунды, отображается 10 секунд и обнуляется. Обязательны к отображению отрицательные числа, получающиеся в результате вычитания. Если подряд нажаты два/три действия, то на экран выводится надпись NO, а затем он возвращается в исходное состояние.

Пример выполнения работы

Подробное оформление отчёта было рассмотрено в примере предыдущей работы. В этой и всех последующих будут приведены только аппаратная схема и программный код для выполнения условия задачи.

Задача № 0.2

Схема содержит восемь светодиодов. Необходимо при помощи сдвигового регистра реализовать их включение по определенной диаграмме.

Аппаратная схема представлена на рисунке 56.

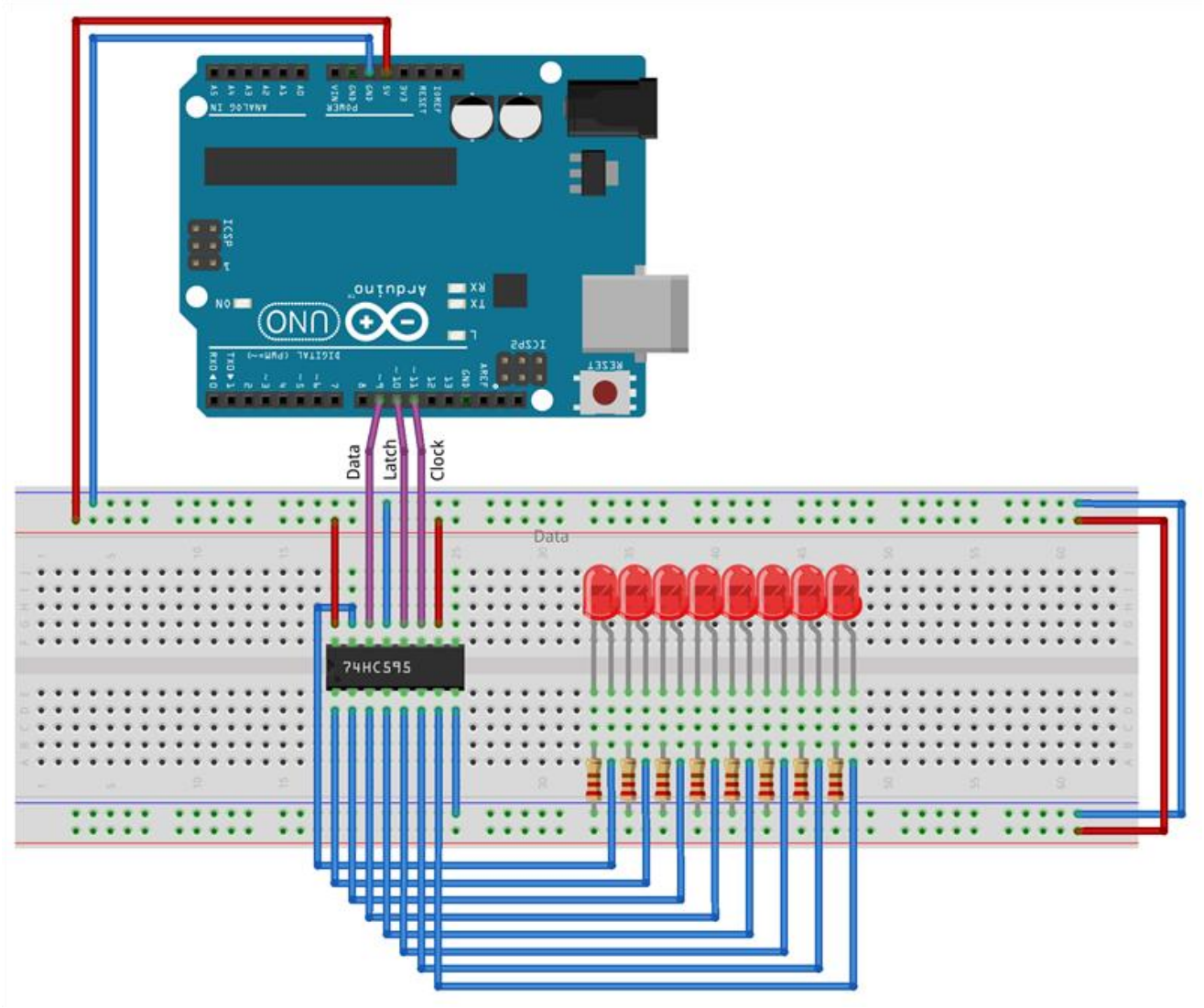


Рисунок 56 – Аппаратная схема к задаче № 0.2

Текст программного кода к задаче № 0.2 представлен на рисунке 57.

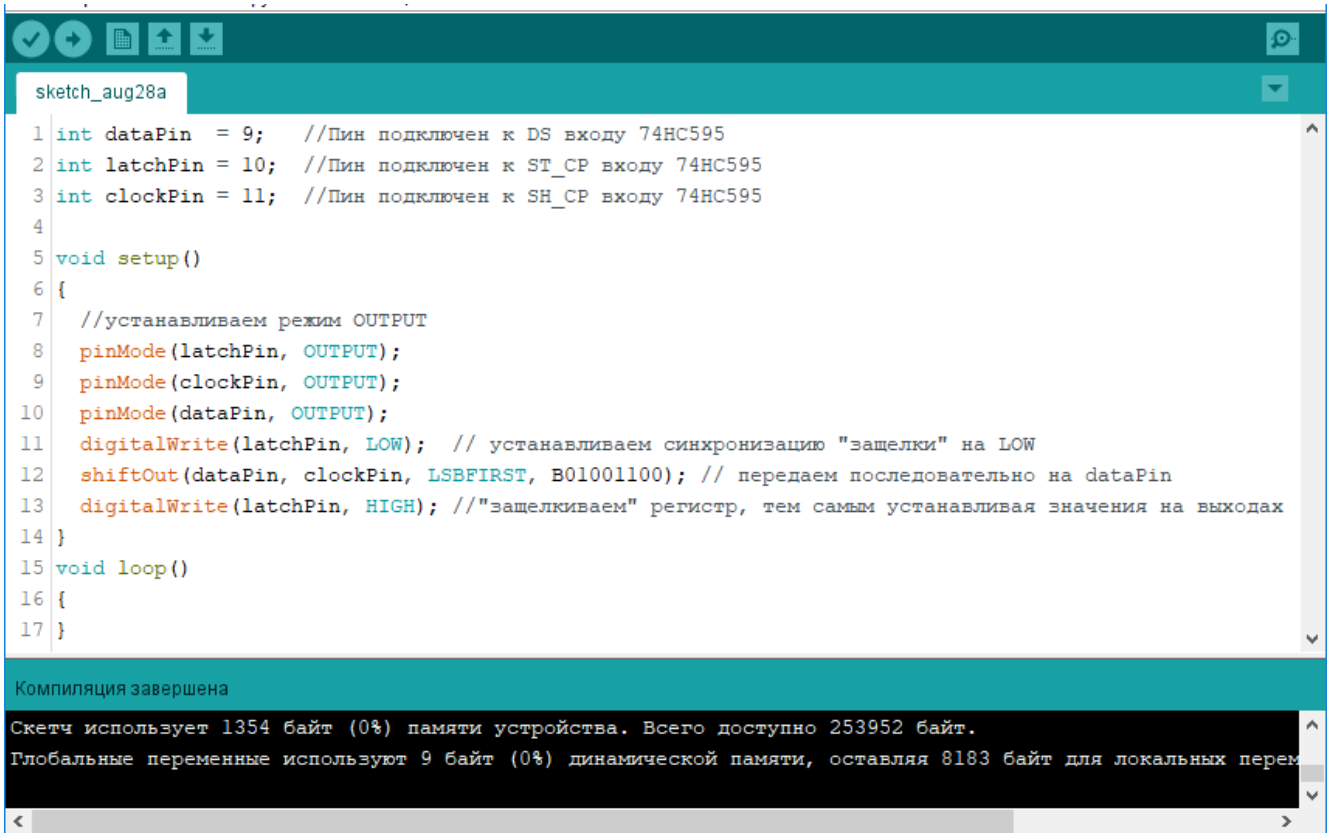


Рисунок 57 – Программный код для задачи № 0.2

При программировании Arduino значение байта можно представлять в двоичном коде. Для этого вначале пишется буква «B» (строка 12 на рисунке 56). Крайний левый бит является старшим, а крайне правый – младшим.

Арифметика, включающая логические операции с переменными типа byte, в этой работе не рассматривается подробно, но рекомендуется к самостоятельному изучению.

Согласно тексту программного кода, представленного на рисунке 56, в функции loop( ) отсутствует цикличность исполнения, потому что сдвиговый регистр сохраняет заданное состояние при постоянном питании.

Задача № 0.3

Схема содержит шестнадцать светодиодов. Необходимо реализовать их управление при помощи двух последовательно подключенных сдвиговых регистров.

Аппаратная схема представлена на рисунке 58.

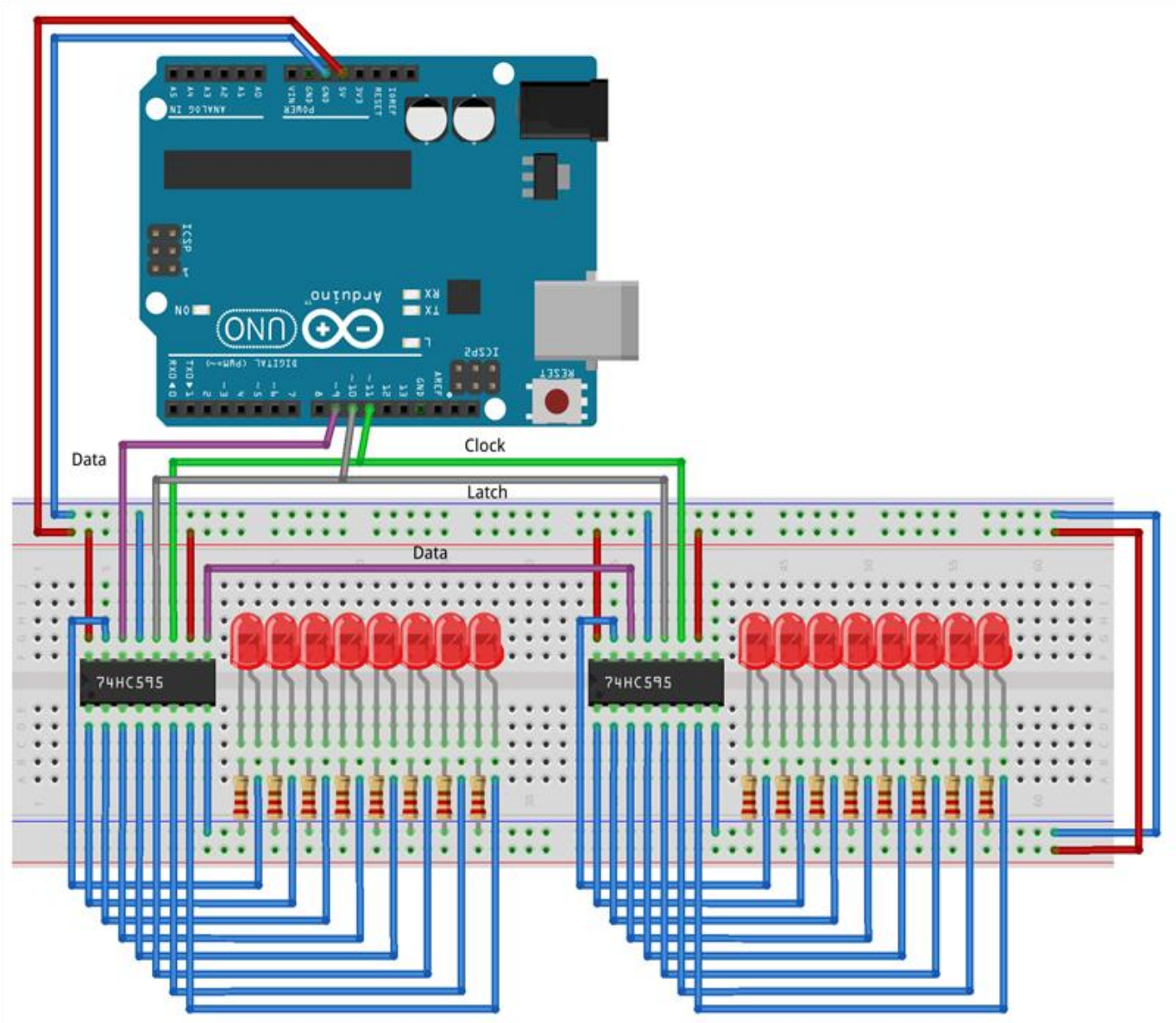


Рисунок 58 – Аппаратная схема к задаче № 0.3

Текст программного кода к задаче № 0.3 представлен на рисунке 59.

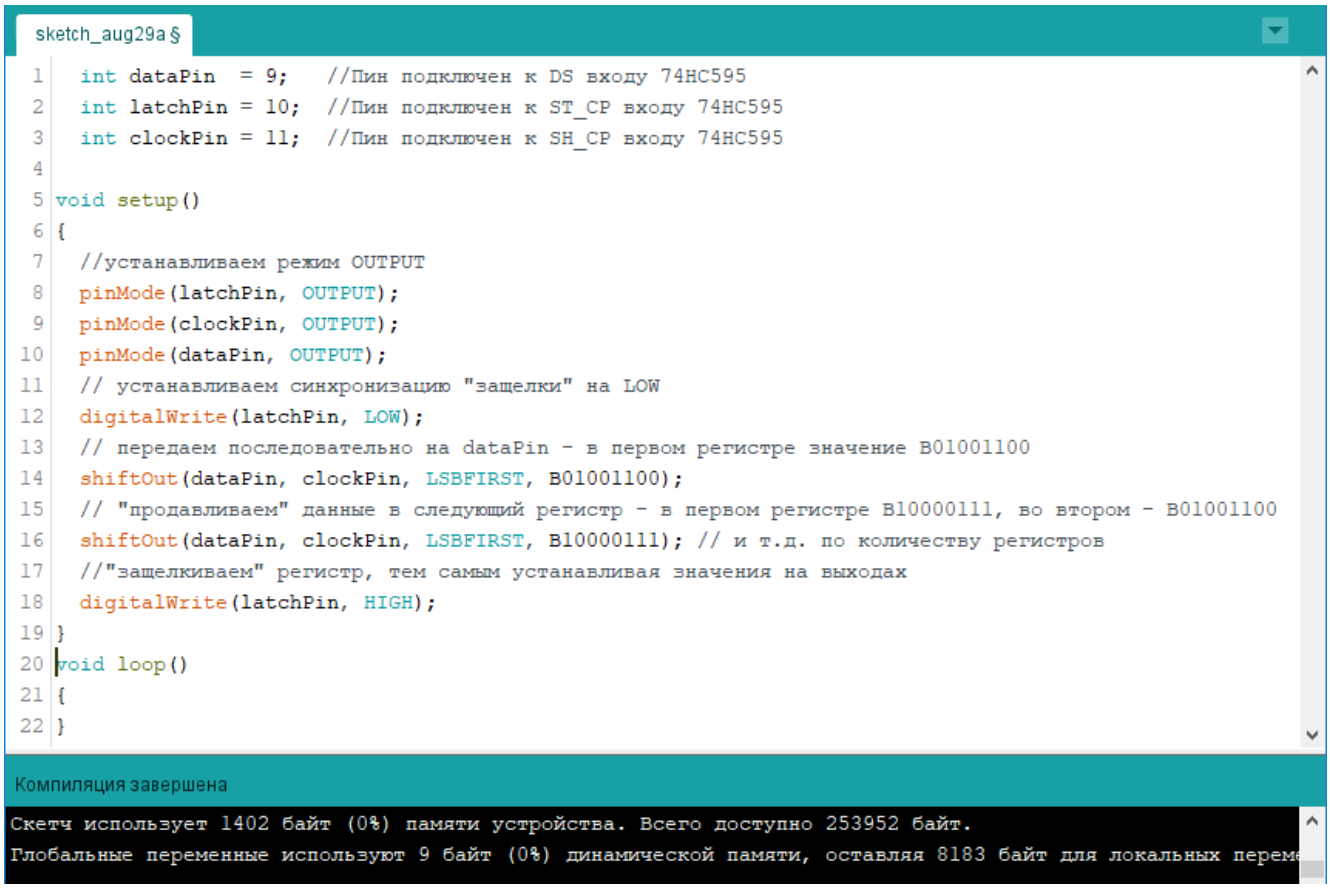


Рисунок 59 – Программный код для задачи № 0.3

Последовательно возможно подключить большое количество сдвиговых регистров, для этого используется его вывод 9 (Q7’), по которому старшие биты из внутреннего последовательного регистра «продавливаются» по мере их поступления. Выходы 11 (SH_CP) и 10 (ST_CP) подключаются параллельно и управляются синхронно. Таким образом к выходу последовательных данных Arduino подключается только один регистр 74HC595.

Вывод:

Применение сдвиговых регистров в системах световой индикации значительно экономит количество задействованных цифровых выводов, но для реализации более сложных алгоритмов управления необходимо обладать знаниями в области битовой арифметики.

Перечень вопросов к самостоятельной работе № 2

1. Какое устройство называют сдвиговым регистром?

2. Для чего предназначен сдвиговый регистр?

3. В чём заключается принцип работы сдвигового регистра?

4. Какие выводы имеет микросхема 74HC595?

5. Имеет ли микросхема 74HC595 аналог российского производства?

6. Сколько ячеек памяти у микросхемы 74HC595?

7. Как передаются биты при последовательном соединении регистров?

8. Объясните алгоритм работы с синхроимпульсами для 74HC595 на примере временных диаграмм.

9. Сколько выводов микроконтроллера занимает сдвиговый регистр?

10. Каким образом фильтруются высокочастотные помехи на регистре?

11. Почему сдвиговые регистры не соединяют параллельно?

12. Возможно ли регулирование ШИМ через сдвиговый регистр?

13. В чем отличия функций shiftOut ( ) и bitWrite ( ) при решении задач на мигание светодиодов, подключённых через сдвиговый регистр?

14. Как в программном коде обозначается начало байта?

15. Какие устройства индикации существуют?

16. Как конструктивно устроен сегментный индикатор?

17. Какие выводы имеет сегментный индикатор?

18. Как маркируется общий катод или анод сегментного индикатора?

19. Какое напряжение необходимо для работы LED индикатора TM1637?

20. В чём достоинства и недостатки применения в проекте LED индикатора TM1637?

21. Как конструктивно устроен LED индикатор TM1637?

22. Какие функции LED индикатора TM1637 реализованы в описанной библиотеке?

Самостоятельная работа № 3 «Схемы с светодиодными матрицами»

Оборудование: макетная плата, Arduino, соединительные провода, резисторы, кнопки, светодиодная матрица, сдвиговые регистры.

Матричный индикатор – это устройство, предназначенное для отображения символов и состоящее из групп светодиодов, расположенных в виде сетки.

В проектах Arduino наиболее распространённым матричным индикатором является светодиодная матрица, размером 8 на 8. Её конструктивное устройство представлено на рисунке 60. Большее разрешение получают составлением нескольких матриц. Все 64 светодиода разделены на 16 групп: 8 вертикальных (столбцов, колонок Column) с общим катодом и 8 горизонтальных (строк Row) с общим анодом.

Поставим задачу – зажечь светодиод R6C3, как показано на рисунке 61а. Тогда на ввод R6 необходимо подать высокий сигнал, а вывод C3 заземлить. Если последовательно зажечь аналогично R3C7, то строки пересекутся с колонками в четырёх местах и отобразится индикация, как показано на рисунке 61б. Поэтому придётся включать рассмотренные точки поочерёдно, но очень быстро (незаметно для человеческого глаза). Этот прием называется динамической индикацией.

Для управления светодиодной матрицей, с целью экономии выходов микроконтроллера можно использовать сдвиговые регистры: один для управления строками, другой столбцами.

Для того, чтобы соединить несколько светодиодных матриц необходимо использовать сдвиговые регистры и микросхемы ULN2003, которые вместе с резисторами и проводами создадут значительные «нагромождения» в системе. Для того, чтобы решить эту проблему применяют специализированные микросхемы для управления индикаторами. Одной из них является MAX7219, представленная на рисунке 62 в составе адаптированного модуля подключения матрицы 8 на 8 [17-18, 20].

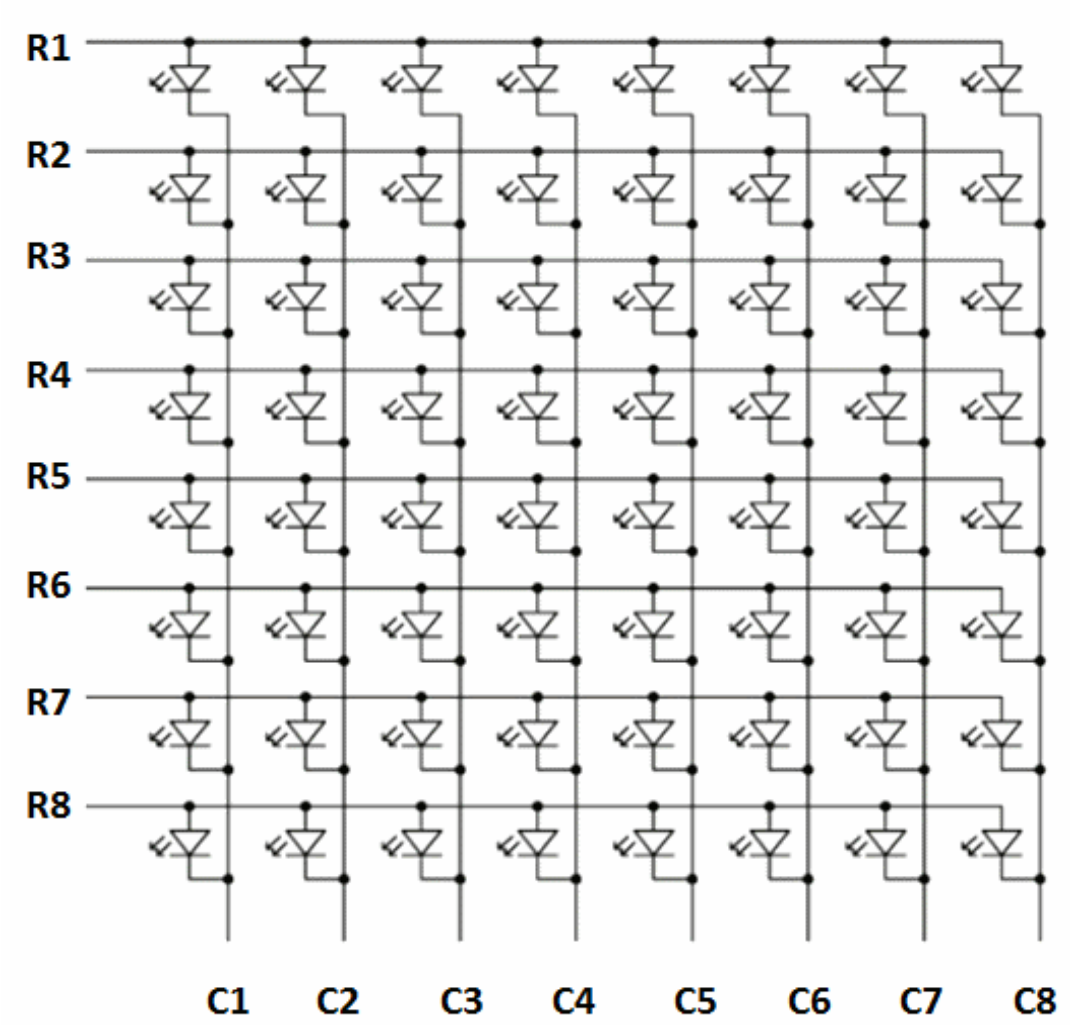
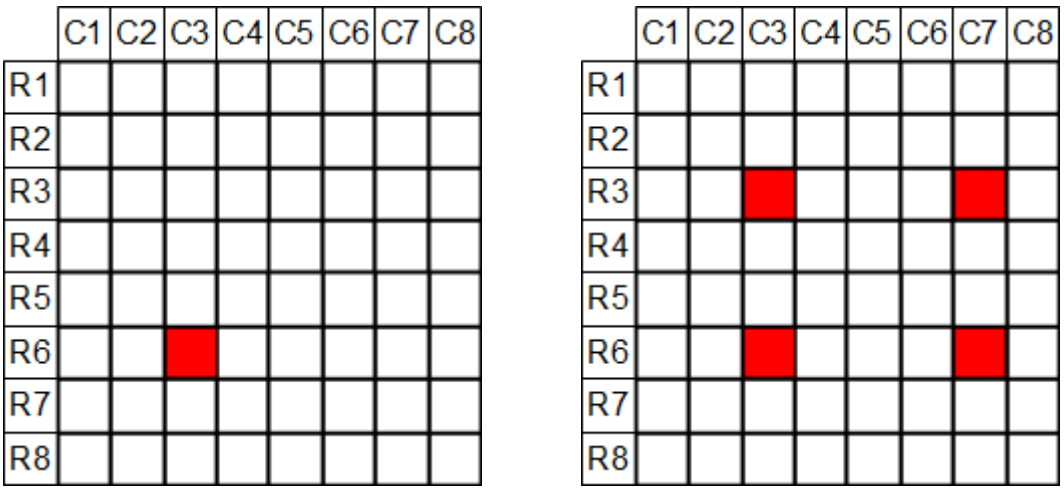


Рисунок 60 – Устройство светодиодной матрицы



а) б)

Рисунок 61 – Устройство индикации в светодиодной матрице

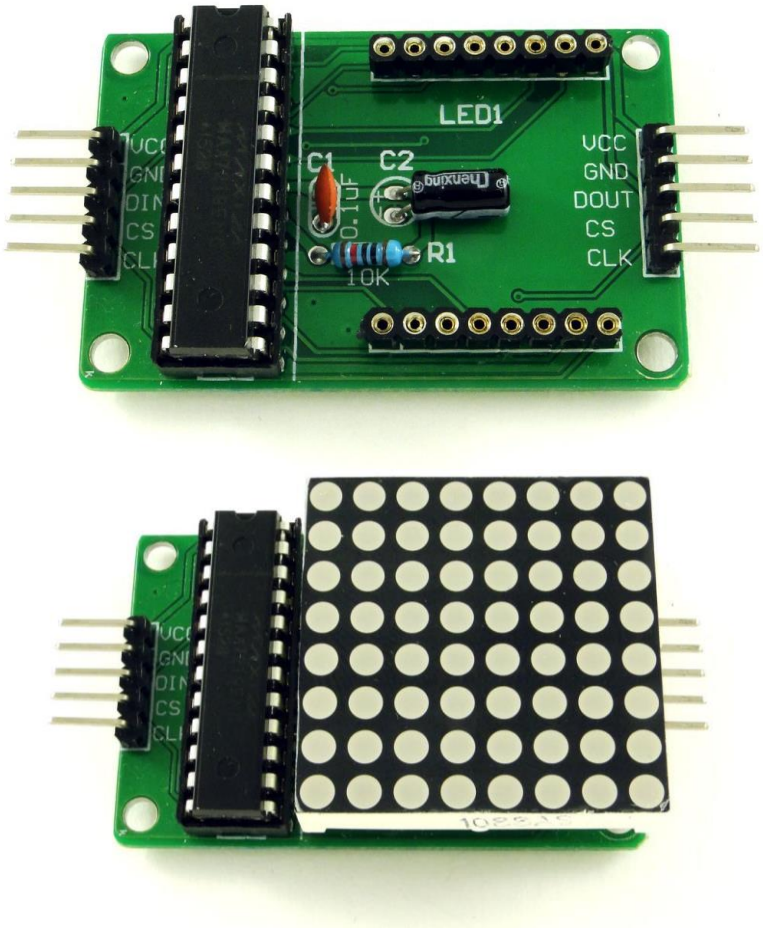


Рисунок 62 – Модуль для светодиодной матрицы с микросхемой MAX7219

Для размещения и закрепления нескольких модулей на одной поверхности, на них реализованы специальные отверстия и разъём под съёмный индикатор. Для соединения матриц в цепочку модуль содержит по пять выводов с двух сторон, для ввода и вывода данных. Рабочее напряжение 5 В. На входе и выходе идентичные контакты: питание VCC и GND, вход данных DIN, выбор модуля CS, синхроимпульс CLK.

Задания для самостоятельной работы №3

Варианты задач по третьей самостоятельной работе указаны в таблице 14.

Таблица 14 – Варианты заданий для самостоятельной работы №3

Вариант	0	1	2	3	4	5	6	7	8	9
Задачи	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,4	1,5	1,6
Здание повышенной сложности	9, 10									

Задача № 1

Схема содержит светодиодную матрицу 8 на 8, два сдвиговых регистра и транзисторную сборку ULN2003. Необходимо реализовать отображение цифры вашего варианта.

Задача № 2

Схема содержит светодиодную матрицу 8 на 8, два сдвиговых регистра и транзисторную сборку ULN2003. Необходимо реализовать переменное отображение цифры вашего варианта и первой буквы имени исполнителя. Временные задержки выбираются самостоятельно.

Задача № 3

Схема содержит светодиодную матрицу 8 на 8, два сдвиговых регистра и транзисторную сборку ULN2003. Необходимо реализовать анимацию смайлика. Задержки и тип смайлика выбираются самостоятельно и должны отличаться от приведённого в примере 0.5.

Задача № 4

Схема содержит четыре модуля на микросхеме MAX7219. Необходимо отобразить бегущую строку с именем и фамилией исполнителя.

Задача № 5

Схема содержит два модуля на микросхеме MAX7219 и потенциометр. При изменении напряжения на одной матрице должны отображаться цифры в диапазоне от 0 до 9, а на второй - соответствующая шкала.

Задача № 6

Схема содержит четыре модуля на микросхеме MAX7219. Необходимо реализовать часы реального времени с отображением часов и минут.

Задача № 7

Схема содержит два модуля на микросхеме MAX7219, три тактовые кнопки и RGB-светодиод. При нажатии на первую кнопку на одной из матриц загораются по одному светодиоду, аналогично заполняется вторая. В зависимости от того, какая матрица заполнится первой, такой цвет светодиода и должен загореться. Третья кнопка служит для обнуления матриц и гашения светодиода.

Задача № 8

Гипноз. Запрограммировать контроллер таким образом, чтобы на светодиодной матрице с периодом в 1 секунду появлялись концентрические окружности с постоянно увеличивающимся радиусом.

Задача № 9

Схема содержит пять модулей на микросхеме MAX7219, и четыре тактовых кнопки. Необходимо реализовать простейший калькулятор. Первая и третья кнопка выводят на соответствующие матрицы числовые значения от одного до десяти. Вторая кнопка отображает действия (сложение, вычитание и умножение). В четвертой матрице всегда горит знак равенства. Последняя кнопка выполняет выбранное математическое действие и отображает результат в пятой матрице. В случае, если какая-то матрица не заполнена, то вся система возвращается в исходное состояние.

Задача № 10

Игра змейка. Собрать из четырех светодиодных матриц дисплей разрешением 16×16 пикселей. Реализовать на Arduino такую известную игру, как «змейка» (или «питон»). В схему необходимо добавить четыре кнопки для управления направлением движения, а также зуммер для сигнализации события съедания яблок.

Пример выполнения работы

Задача № 0.4

Схема содержит модуль на микросхеме MAX7219. Необходимо реализовать мигание светодиода с заданными координатами (3,4).

Аппаратная схема представлена на рисунке 63.

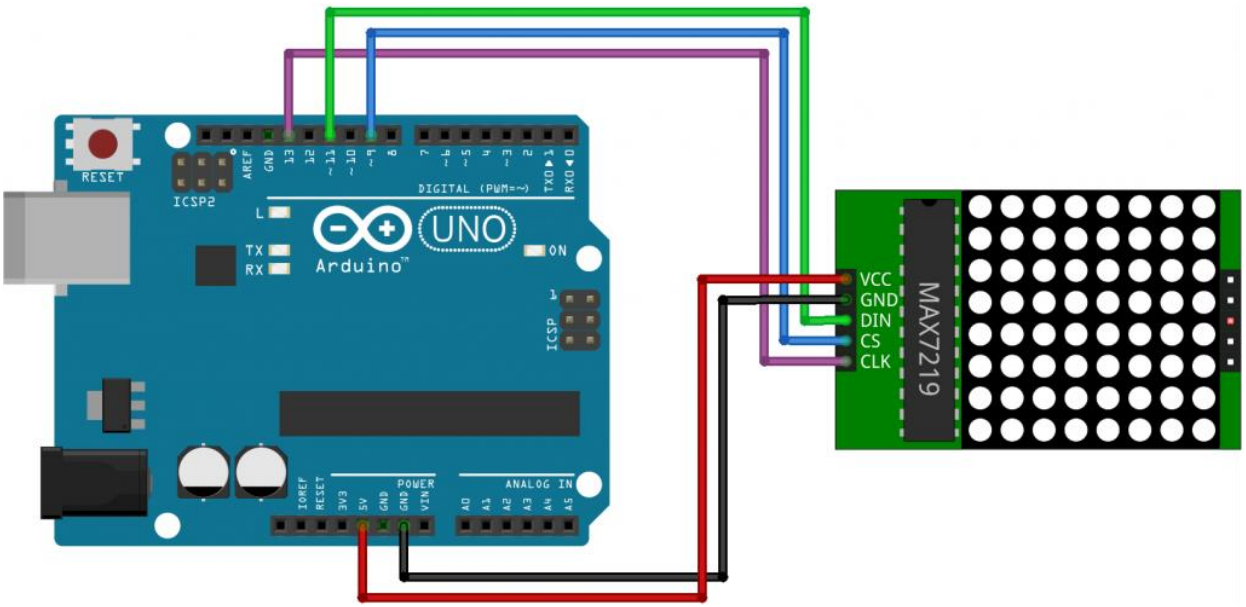


Рисунок 63 – Аппаратная схема для решения задачи № 0.4

Подключение матричного модуля к плате Arduino осуществляется соответственно по принципиальной схеме, представленной на рисунке 64 (VCC/5V, GND/GND, CIN/11, CS/9, CLK/13).

Для работы с используемым модулем применяется ряд библиотек, которые можно взять у руководителя работы или скачать в интернете: SPI, Adafruit_GFX и Max72xxPanel.

Текст программного кода к задаче №0.4 представлен на рисунке 65.

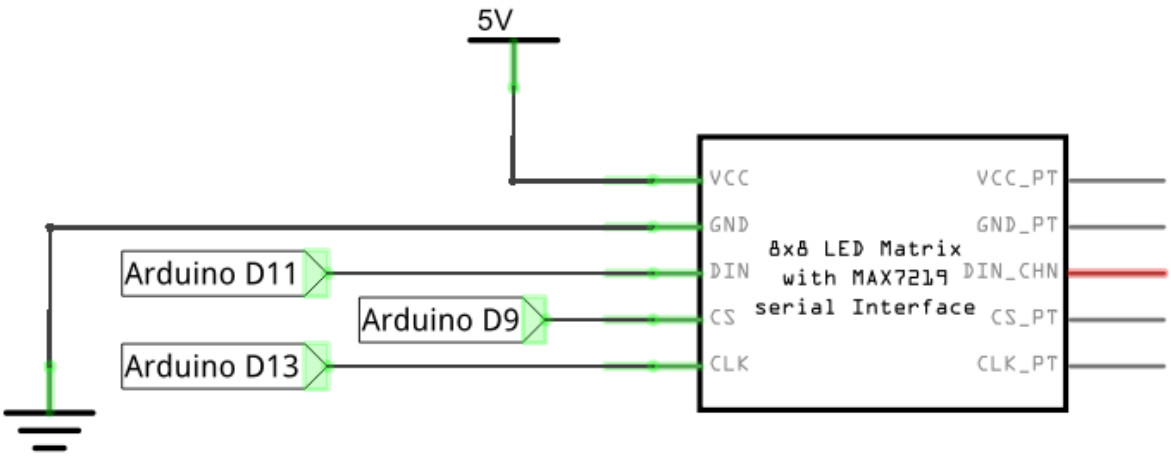


Рисунок 64 – Принципиальная схема подключения модуля MAX7219

sketch_sep09a

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 9;
int numberOfHorizontalDisplays = 1; // количество матриц по-горизонтали
int numberOfVerticalDisplays = 1; // количество матриц по-вертикали

Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays, numberOfVerticalDisplays);

void setup() {
    matrix.setIntensity(4); // яркость от 0 до 15
}

void loop() {
    matrix.drawPixel(3, 4, HIGH); // зажигаем пиксель с координатами {3,4}
    matrix.write(); // вывод всех пикселей на матрицу
    delay(300);
    matrix.drawPixel(3, 4, LOW); // гасим пиксель
    matrix.write();
    delay(300);
}
```

Компиляция завершена

```
Скетч использует 5882 байт (2%) памяти устройства. Всего доступно 253952 байт.
Глобальные переменные используют 81 байт (0%) динамической памяти, оставляя 8111 байт для локальных
```

Рисунок 65 – Программный код для задачи № 0.4

Включение и отключение пикселей осуществляется функцией drawPixel. При этом нужно вызывать и функцию write, иначе на матрице ничего не отобразится.

Задача № 0.5

Принципиальная схема аналогична использованной в задаче № 0.4. Необходимо отобразить смайлик, используя в программе восьмибайтный массив, каждый из которых будет отвечать за строку матрицы, а биты за столбец.

Текст программного кода к задаче №0.5 представлен на рисунке 66.

sketch_sep09b

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 9;
int numberOfHorizontalDisplays = 1; // количество матриц по-горизонтали
int numberOfVerticalDisplays = 1; // количество матриц по-вертикали

Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays, numberOfVerticalDisplays);

const byte data[8] = {
    0b00111100,
    0b01000010,
    0b10100101,
    0b10000001,
    0b10100101,
    0b10011001,
    0b01000010,
    0b00111100};

void setup() {
    matrix.setIntensity(7); // яркость от 0 до 15

    matrix.fillScreen(LOW); // очистка матрицы
    for ( int y = 0; y < 8; y++ ) {
        for ( int x = 0; x < 8; x++ ) {
            // зажигаем x-й пиксель в y-й строке
            matrix.drawPixel(x, y, data[y] & (1<<x));
        } }
    matrix.write(); // вывод всех пикселей на матрицу
}
void loop() {
}
```

Компиляция завершена

```
Скетч использует 5736 байт (2%) памяти устройства. Всего доступно 253952 байт.
Глобальные переменные используют 89 байт (1%) динамической памяти, оставляя 8103 байт для локальных
```

Рисунок 66 – Программный код для задачи № 0.5

Матрица 8x8 заполняется последовательно по пикселям. Для этого организовано 2 цикла, вложенных друг в друга. Первый цикл (строка 25) сканирует последовательно строки матрицы, которые фактически являются элементами массива data[8]. Вложенный цикл (строка 26) организует сканирование по пикселям по горизонтали. В строке 28 рисуется пиксель с координатами x, y со значением, соответствующим биту текущего значения массива data[y]. Значение бита определяется путем наложения «маски» (битовый сдвиг влево 1 на значение x (1<<x)) через побитовое И (&) на значение data[y]. После битового сдвига маска будет содержать только один бит, равный единице, но расположенный в позиции, соответствующей текущему пикселю матрицы.

В строке 30 матрица отображается.

Функция matrix setRotation позволяет менять ориентацию изображения на матрице. Она используется после функции setIntensity с соответствующими аргументами: первый – индекс матрицы, второй – количество поворотов на 90 градусов.

Задача № 0.6

Принципиальная схема аналогична использованной в задаче №0.4. Необходимо отобразить на матрице последовательно буквы «HELLO WORLD!».

Текст программного кода к задаче №0.6 представлен на рисунке 67.

В библиотеке Adafruit-GFX-Library размещены функции для работы с графикой и текстом (база латинских букв и символов в верхнем и нижнем регистрах). Для отображения символов используется функция drawChar (x, y, символ, цвет, фон, размер). Координаты указывают расположение верхнего левого угла символа, цвет указывается HIGH, фон LOW, а размер 1.

Задача № 0.7

Схема содержит шесть горизонтально соединенных модулей на микросхеме MAX7219. Необходимо реализовать бегущую строку «HELLO WORLD!».

Принципиальная схема представлена на рисунке 68.

sketch_oct26a

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 9;
int numberOfHorizontalDisplays = 1;
int numberOfVerticalDisplays = 1;

Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays, numberOfVerticalDisplays);

String tape = "HELLO WORLD!";
int wait = 20;

void setup() {
    matrix.setIntensity(7); // яркость от 0 до 15
}

void loop() {
    for ( int i = 0 ; i < tape.length(); i++ ) {
        matrix.fillScreen(LOW);
        matrix.drawChar(0, 0, tape[i], HIGH, LOW, 1);
        matrix.write();
        delay(wait);
    }
}
```

Компиляция завершена

```
Скетч использует 7682 байт (23%) памяти устройства. Всего доступно 32256 байт.
Глобальные переменные используют 120 байт (5%) динамической памяти, оставляя 1928 байт для локаль
```

Рисунок 67 – Программный код для задачи № 0.6

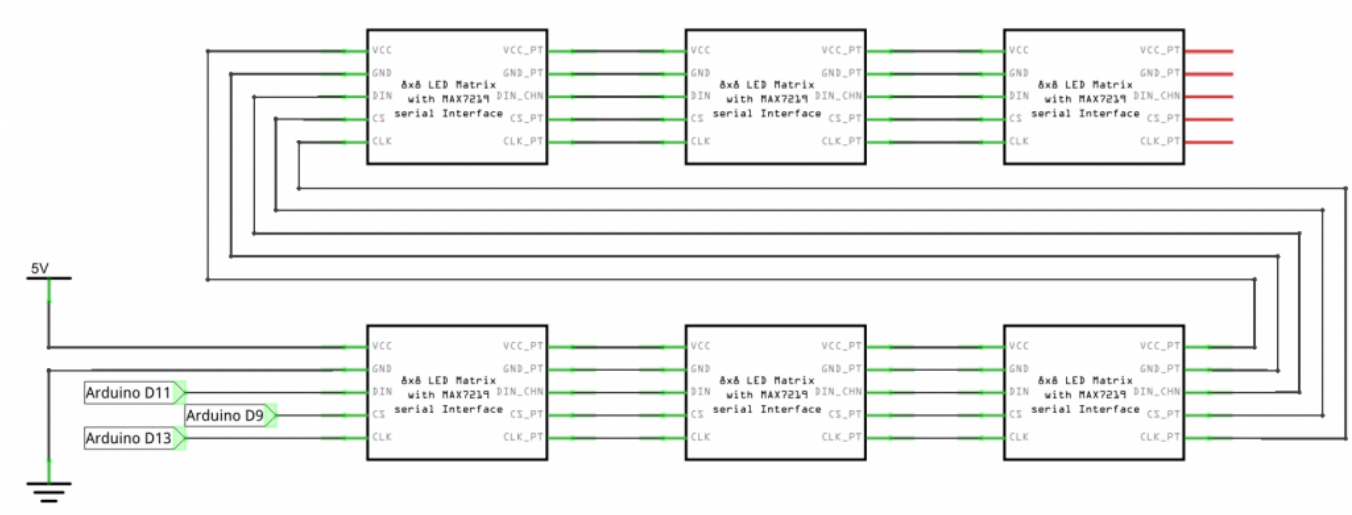


Рисунок 68 – Принципиальная схема для задачи № 0.7

Аппаратная схема аналогична представленной на рисунке 63, но содержит 6 последовательно подключённых модулей.

В основе реализации программного кода для решения данной задачи лежит сдвиг координат букв.

Пример программного кода представлен на рисунке 69.

sketch_sep10b

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 9;
int numberOfHorizontalDisplays = 6; // теперь у нас по-горизонтали 6 матриц
int numberOfVerticalDisplays = 1; // а по-вертикали, по-прежнему, одна

Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays, numberOfVerticalDisplays);

String tape = "HELLO WORLD!"; // текст, который будет плыть
int wait = 20; // время между крайними перемещениями букв

int spacer = 1; // расстояние между буквами
int width = 5 + spacer; // размер шрифта

void setup() {
    matrix.setIntensity(7); // яркость
}

void loop() {
    for ( int i = 0 ; i < width * tape.length() + matrix.width() - 1 - spacer; i++ ) {
        matrix.fillScreen(LOW);

        int letter = i / width;
        int x = (matrix.width() - 1) - i % width;
        int y = (matrix.height() - 8) / 2; // center the text vertically

        while ( x + width - spacer >= 0 && letter >= 0 ) {
            if ( letter < tape.length() ) {
                matrix.drawChar(x, y, tape[letter], HIGH, LOW, 1);
            }
            letter--;
            x -= width;
        }

        matrix.write();
        delay(wait);
    }}
```

Компиляция завершена

```
Скетч использует 6820 байт (2%) памяти устройства. Всего доступно 253952 байт.
Глобальные переменные используют 104 байт (1%) динамической памяти, оставляя 8088 байт для локальных
```

Рисунок 69 – Программный код для задачи № 0.7

В функции loop(), которая сама периодически выполняется, организованы два цикла. Первый с счетчиком i (строка 22) обеспечивает последовательный вывод всех букв строкового массива tape по столбцам матрицы (горизонтальным пикселям). Соответственно максимальное значение счетчика рассчитывается как длина текста в пикселях (width * tape.length()) плюс длина (ширина) пустой матрицы (matrix.width()).

Второй вложенный цикл выводит часть сообщения, которое в данный момент умещается на матрице.

Алгоритм построен таким образом, что бегущая строка выводится, начиная справа и двигается влево.

Очищается матрица (строка 23), затем рассчитывается текущая буква текста, которая будет отображаться на матрице в крайнем правом положении (строка 24 – целое от деления i на ширину буквы с пробелом). В строке 26 определяется координата X по матрице, где будет находиться эта буква. Эта координата зависит от счетчика i. С каждой новой буквой начальная координата равна последнему столбцу матрицы (matrix.width()-1) и постепенно уменьшается (сдвигается к началу) на ее ширину (остаток от деления i на width). Координата Y не изменяется и всегда равна нулю для однострочной матрицы (строка 27).

В цикле while последовательно выводятся все буквы - начиная с крайней правой до самой первой в тексте. Цикл будет продолжаться до тех пор, пока будут выполняться оба условия: текущая координата «x» вместе с шириной буквы больше или равно нулю и текущий индекс строкового массива больше или равно нулю (строка 29). Первое условие (x + width - spacer >= 0) обеспечивает вывод буквы в крайнем левом положении, даже если хотя бы часть ее помещается в матрице. Второе условие (letter >= 0) проверяет на начало строкового массива, и заканчивает цикл, если напечатали самую первую букву. Аналогично в строке 30 проверяется конечная буква строкового массива. В строке 31 печатаются текущие буквы в координатах (x, y). В строке 33 цикл переходит на следующую букву, стоящую ближе к началу, а в строке 34 рассчитывается ее координата левее на длину width.

Вся матрица выводится в строке 37, затем включается пауза общего цикла.

Вывод:

Разработка проектов Arduino со светодиодными матрицами значительно упрощается за счёт уже разработанных библиотек: SPI, Adafruit_GFX и Max72xxPanel. Для упрощения проекта и мобильности устройства рекомендуется использовать модули на микросхеме MAX7219.

Перечень вопросов к самостоятельной работе № 3

1. Что такое матричный индикатор?

2. Как устроен матричный индикатор?

3. Где применяются матричные индикаторы?

4. Возможно ли подключение нескольких матричных индикаторов для вывода цельной информации?

5. Как реализуется мигание одним светодиодом (пикселем) на матичном индикаторе?

6. Используются ли сдвиговые регистры для управления матричными индикаторами? Если да, то как?

7. Для чего в модуле подключения светодиодной матрицы используют микросхему MAX7219?

8. Какие выводы имеет модуль на микросхеме MAX7219?

9. Какие виды изображений возможно отобразить на матричном индикаторе?

10. Какие библиотеки Arduino рекомендуется использовать при работе с матричным индикатором?

11. Какая функция используется для включения и отключения пикселей на матричном индикаторе?

12. Какая функция используется для изменения ориентации изображения на матрице?

13. В чем особенность применения функции drawChar?

14. Как реализуется передача изображения на группу соединенных матричных индикаторов?

15. Возможно ли реализовать анимацию на матричном индикаторе? Если да, то как?

Самостоятельная работа № 4 «Схемы с LCD дисплеями»

Оборудование: макетная плата, Arduino, соединительные провода, резисторы, кнопки, LCD дисплей 1602, потенциометр, модуль I2C.

Чаще всего для отображения информации в проектах на Arduino используют LCD дисплеи. Для их работы не требуются сдвиговые регистры и глубокие знания в области математической логики. Самой распространённой моделью является 1602, представленный в общем виде на рисунке 70.

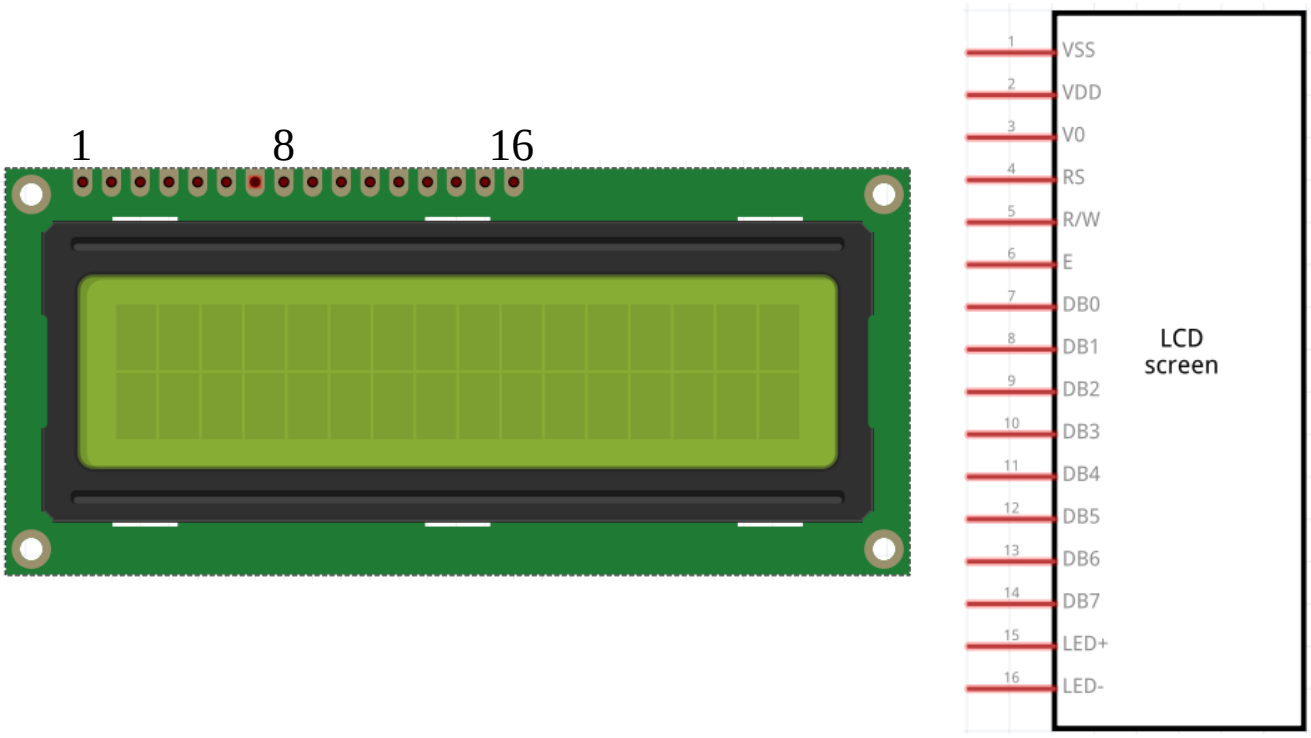


Рисунок 70 – LCD дисплей 1602 и его выводы

Обозначение выводов для LCD дисплея 1602 и соответствующие контакты при подключении к Arduino обозначены в таблице 15.

Для работы всех видов дисплеев с Arduino используются функции стандартной библиотеки LiquidCrystal.

Таблица 15 – Контакты LCD дисплея 1602 и их подключение к Arduino

Вывод	Обозначение	Описание	Arduino
1	GND	Общий вывод (земля)	GND
2	Vcc	Напряжение питания (3,3 – 5 В)	5V
3	Vo	Управление контрастностью	GND
4	RS	Выбор регистра ОЗУ/управления	12
5	R/W	Выбор режима записи или чтения	GND
6	E	Разрешение обращения к индикатору	11
7	DB0	Шина данных (режим 8 бит, младший)	–
8	DB1	Шина данных (режим 8 бит)	–
9	DB2	Шина данных (режим 8 бит)	–
10	DB3	Шина данных (режим 8)	–
11	DB4	Шина данных (режим 8 и 4 бит, младший)	5
12	DB5	Шина данных (режим 8 и 4 бит)	4
13	DB6	Шина данных (режим 8 и 4 бит)	3
14	DB7	Шина данных (режим 8 и 4 бит)	2
15	+LED	Плюс питания подсветки	5V
16	-LED	Минус питания подсветки	GND

Последовательная шина связи интегральных схем I^2C (Inter-Integrated Circuit) использует двухпроводное 8-ми битное соединение. Для соединения LCD дисплея 1602 и Arduino используется специальный модуль, который сокращает количество применяемых выводов до четырёх (два для передачи данных, два питания). В общем виде они представлены на рисунке 71.

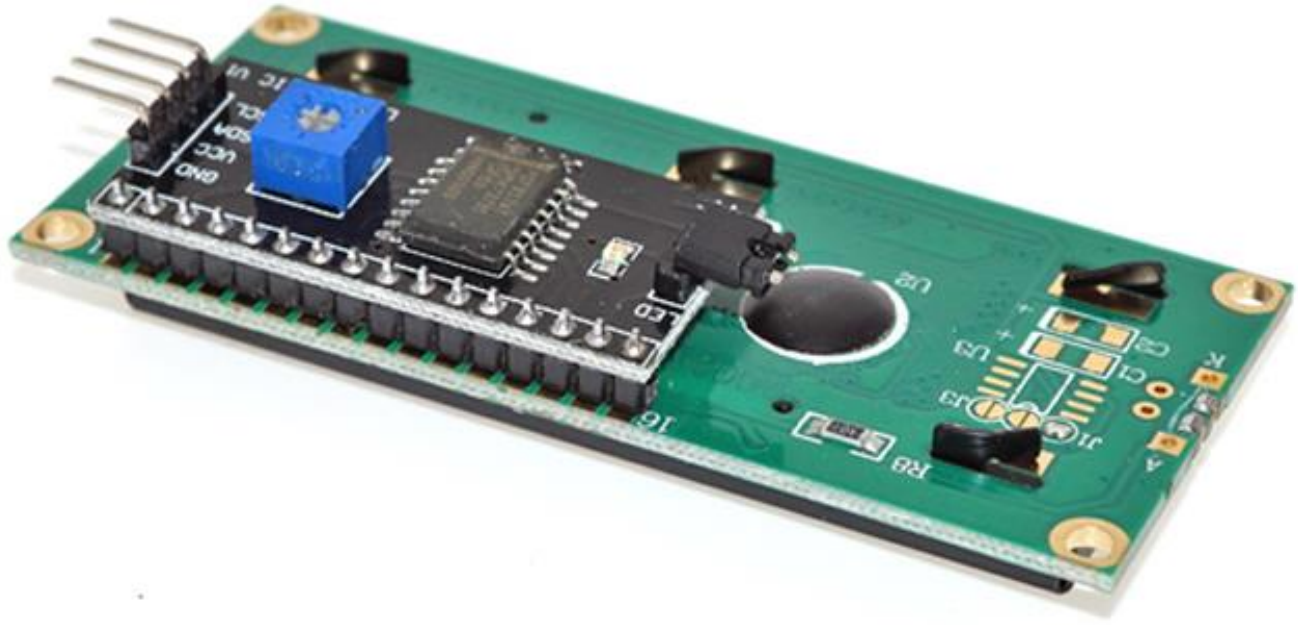


Рисунок 71 – LCD дисплей 1602 с модулем I^{2}C

В системе с I^{2}C одно устройство является ведущим (инициирует передачу данных по SDA и синхронизирует сигнал по SCL), остальные - ведомыми.

Для работы с представленной шиной, LCD дисплеем 1602 и Arduino существуют библиотеки LiquidCrystal_I2C и Wire.

Задания для выполнения работы № 4

Варианты задач по четвертой самостоятельной работе указаны в таблице 16.

Таблица 16 – Варианты заданий для самостоятельной работы № 4

Вариант	0	1	2	3	4	5	6	7	8	9
Задачи	1,6	2,7	3,8	4,6	5,7	1,8	2,6	3,7	4,8	5,1
Здание повышенной сложности	9,10									

Задача № 1

Схема содержит LCD дисплей 1602 и потенциометр для регулировки его яркости. Необходимо вывести в первой строке надпись «Arduino Good!», второй «Elvis Forever!».

Задача № 2

Схема содержит LCD дисплей 1602 и потенциометр для регулировки его яркости. Необходимо вывести текст, указанный в предыдущей задаче, в виде бегущей строки.

Задача № 3

Схема содержит LCD дисплей 1602 и потенциометр для регулировки его яркости. Необходимо построчно вывести ФИО разработчика на русском языке.

Задача № 4

Схема содержит LCD дисплей 1602 и два потенциометра. Один из них отвечает за яркость изображения. При изменении положения второго первая строка заполняется чёрными прямоугольниками (аналогично шкальному индикатору слева направо), а вторая наоборот осветляется (справа налево).

Задача № 5

Схема содержит LCD дисплей 1602, потенциометр для регулировки его яркости и 2 тактовые кнопки. Нажатие на первую кнопку отображает ФИО разработчика на английском языке, на вторую на русском.

Задача № 6

Схема содержит LCD дисплей 1602 с модулем I^2C и потенциометром для регулировки яркости. Необходимо в верхней строке отобразить надпись «Время», а по центру нижней непосредственно часы.

Задача № 7

Схема содержит LCD дисплей 1602 с модулем I^2C и потенциометром для регулировки яркости, а также три тактовые кнопки. При нажатии на первую в верхней строке отображается цифра 1 (каждое последующее нажатие дополняет строку на один символ), аналогично вторая кнопка в нижней строке отображает 2. Третья кнопка сбрасывает изображение в исходное состояние.

Задача № 8

Схема содержит LCD дисплей 1602 с модулем I^2C и потенциометром для регулировки яркости, а также две тактовые кнопки и матричный модуль. В исходном состоянии не отображается ничего. При нажатии на одну из кнопок на матричном модуле отображаются цифры от 0 до 3, на другую - они построчно переносятся на дисплей.

Задача № 9

Схема содержит LCD дисплей 1602 с модулем I^2C, два потенциометра (один отвечает за яркость) и RGB-светодиод. При изменении положения второго потенциометра должен изменяться цвет свечения светодиода (минимум 5 различных цветов), а его название должно отображаться на дисплее.

Задача № 10

Схема содержит LCD дисплей 1602 с модулем I^2C и потенциометром для регулировки яркости, а также тринадцать тактовых кнопок. Десять из них отвечают за ввод цифр от 0 до 9, одна перебирает последовательно математические действия (+, - ,*), вторая осуществляет расчёт, а третья обнуляет дисплей. В верхней строке посередине осуществляется запись выражения, а в нижней выводится ответ. В расчётах могут принимать участие только однозначные и двузначные числа.

Пример выполнения работы

Задача № 0.8

Схема состоит из LCD дисплея 1602, подключённого к Arduino, и потенциометра для изменения яркости. Необходимо отобразить в первой строке надпись «Hello world», а во второй «Do It Youself». Программный код и аппаратная схема представлены на рисунках 72 и 73.

helloWorld

```
// подключаем стандартную библиотеку LiquidCrystal
#include <LiquidCrystal.h>

// инициализируем объект-экран, передаём использованные
// для подключения контакты на Arduino в порядке:
// RS, E, DB4, DB5, DB6, DB7
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
    // устанавливаем размер (количество столбцов и строк) экрана
    lcd.begin(16, 2);
    // печатаем первую строку
    lcd.print("Hello world");
    // устанавливаем курсор в колонку 0, строку 1
    // на самом деле это вторая строка, т.к. нумерация начинается с нуля
    lcd.setCursor(0, 1);
    // печатаем вторую строку
    lcd.print("Do It Yourself");
}

void loop()
{
}
```

Компиляция завершена

```
Скетч использует 2182 байт (0%) памяти устройства. Всего доступно 253952 байт
Глобальные переменные используют 73 байт (0%) динамической памяти, оставляя 8
```

Рисунок 72 – Программный код для задачи № 0.8

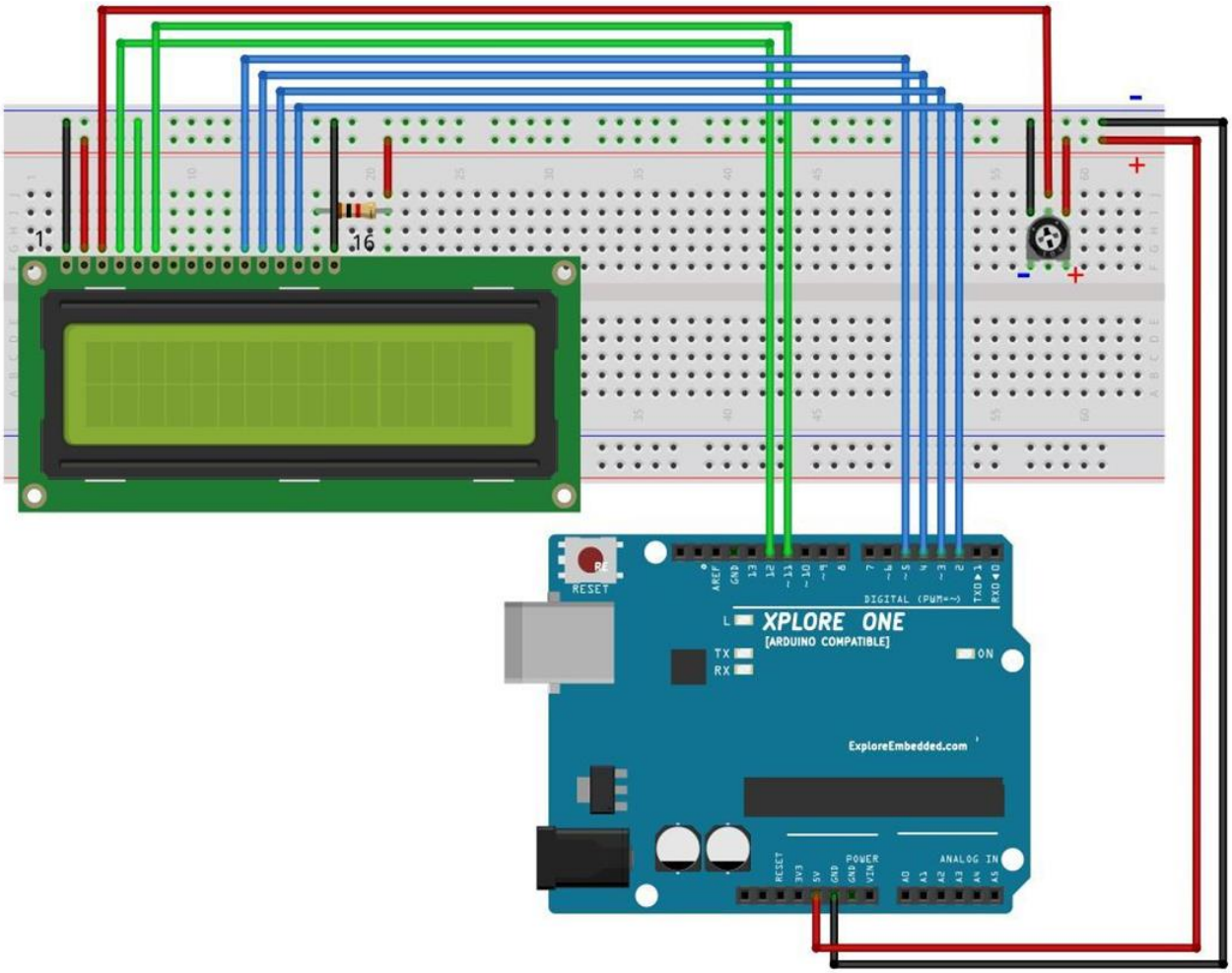


Рисунок 73 – Аппаратная схема для задачи № 0.8

В библиотеке LiquidCrystal_I2C.h содержится множество команд для управления LCD дисплеем по шине I²C, что значительно упрощает структуру скетча. Перед выводом данных на дисплей задаётся положение курсора при помощи команды setCursor(0,1), где 0 – номер символа в строке, 1 – номер строки.

Задача № 0.9

Схема состоит из LCD дисплея 1602, подключённого к Arduino по шине I²C, и потенциометра для изменения яркости. Необходимо отобразить в первой строке надпись «I LOVE», а во второй «ARDUINO». Программный код и аппаратная схема представлены на рисунках 74 и 75.

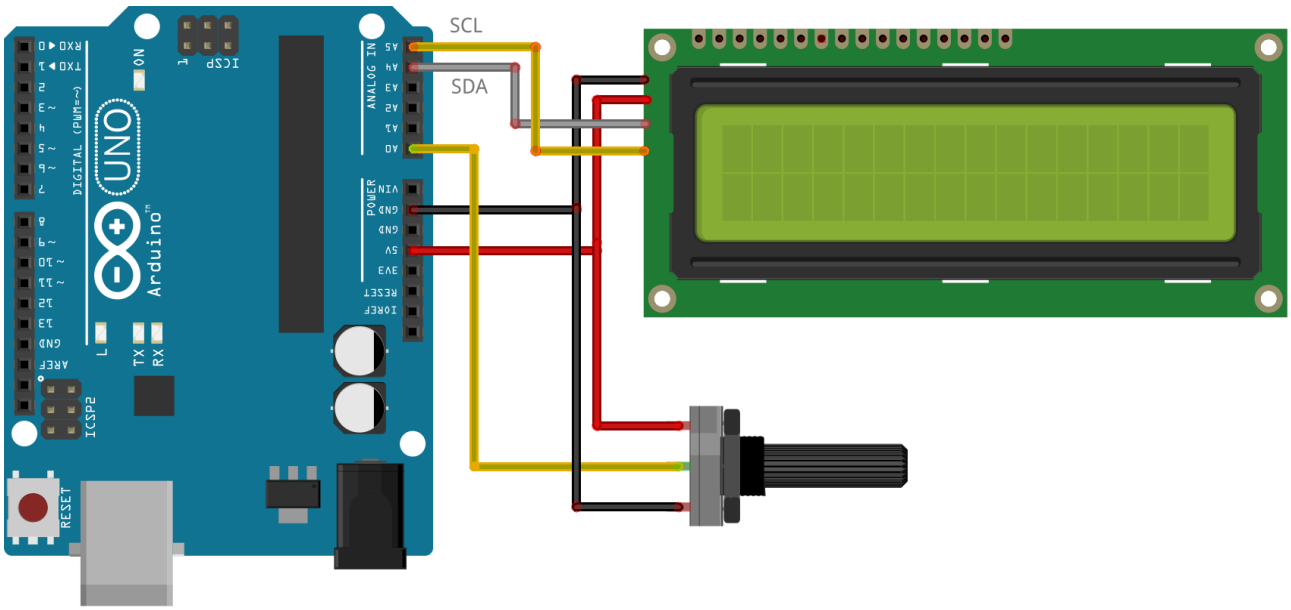


Рисунок 74 – Аппаратная схема для задачи № 0.9

lcd_i2c

```
#include <Wire.h> // библиотека для управления устройствами по I2C
#include <LiquidCrystal_I2C.h> // подключаем библиотеку для LCD 1602
LiquidCrystal_I2C lcd(0x27,20,2); // присваиваем имя lcd для дисплея 20x2

void setup() // процедура setup
{
  lcd.init(); // инициализация LCD дисплея
  lcd.backlight(); // включение подсветки дисплея

  lcd.setCursor(0,0); // ставим курсор на 1 символ первой строки
  lcd.print("I LOVE"); // печатаем сообщение на первой строке

  lcd.setCursor(0,1);  // ставим курсор на 1 символ второй строки
  lcd.print("ARDUINO"); // печатаем сообщение на второй строке
}

void loop() // процедура loop
{
}
```

Компиляция завершена

```
Скетч использует 3330 байт (1%) памяти устройства. Всего доступно 253952 байт.
Глобальные переменные используют 235 байт (2%) динамической памяти, оставляя 79
```

Рисунок 75 – Программный код для задачи № 0.9

Вывод:

Для вывода информации на LCD дисплей используются функции из библиотек LiquidCrystal, LiquidCrystal_I2C и Wire. Возможны подключения к Arduino напрямую, а также через шину I²C (экономит используемые выводы и упрощает код программы).

Перечень вопросов к самостоятельной работе № 4

1. Используются ли сдвиговые регистры при подключении LCD дисплеев?

2. Сколько выводов шины данных имеет LCD дисплей 1602?

3. Сколько выводов на LCD дисплее 1602 отводится под питание и заземление? Почему недостаточно двух выводов для реализации питания?

4. Какие библиотеки используются для работы с дисплеями и Arduino?

5. Какие способы подключения LCD дисплеев к плате Arduino используются чаще всего?

6. Какие достоинства имеет подключение LCD дисплея по последовательной шине связи I^2C по сравнению с шестнадцати контактным?

7. Какие недостатки имеет подключение LCD дисплея по последовательной шине связи I^2C по сравнению с шестнадцати контактным??

8. Какие существуют способы вывода латинских букв на LCD дисплей при помощи Arduino?

9. Какая команда используется для вывода курсора на дисплей?

10. Возможно ли использование нескольких дисплеев в проекте на базе Arduino Nano?

11. Какое максимальное количество LCD дисплеев возможно подключить к Arduino Mega?

12. Возможно ли последовательное соединение LCD дисплеев в проекте Arduino для вывода бегущей строки, как при использовании матричных модулей в задаче примера №0.7?

13. Какие визуальные параметры дисплеев можно регулировать при помощи Arduino? (не менее 5)

14. В чем будет заключаться принципиальное отличие программного кода Arduino при использовании различных моделей дисплеев (больших в размере, с разнообразной подсветкой)?

15. Возможно ли применение LCD дисплеев в проектах с Arduino для вывода информации с аналоговых и цифровых датчиков?

Список использованных источников

1. ArduinoKit. Мир электронного творчества, хобби и увлечений [Электронный ресурс]. – Режим доступа: http://arduinokit.ru/
2. Francis Perea, Arduino Essentials. – Published by Packt Publishing Ltd, Birmingham B3 2PB, UK, 2015. – P. 206. – ISBN 987-1-78439-856-9.
3. Блог о робототехнике, электронике и алгоритмах «РОБОТОША» [Электронный ресурс]. – Режим доступа: http://robotosha.ru/
4. Блог электроного магазина «Вольтик» [Электронный ресурс]. – Режим доступа: https://voltiq.ru/
5. Блум, Д. Изучаем Arduino: инструменты и методы технического волшебства. – Спб. БХВ-Петербург, 2015. – 336 с. – ISBN 987-5-9775-3585-4.
6. ГОСТ 16465-70 Сигналы радиотехнические измерительные. Термины и определения. – Издательство ФГУП «СТАНДАРТИНФОРМ», 2015. – 16 с.
7. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. – Издательство ФГУП «СТАНДАРТИНФОРМ», 2010. – 23 с.
8. ГОСТ Р ИСО 8373-2014. Роботы и робототехнические устройства. Термины и определения. – Издательство ФГУП «СТАНДАРТИНФОРМ». – 16 с.
9. Монк, С. Программируем Arduino. Профессиональная работа со скетчами. – СПб.: Питер, 2017. – 164 с. – ISBN 978-5-496-02385-6.
10. Образовательный проект хакспейса MakeItLab – РоботКласс [Электронный ресурс]. – Режим доступа: http://robotclass.ru/
11. Arduino в России: [офиц. сайт]. – Режим доступа: http://arduino.ru/
12. Arduino на Украине: [офиц. сайт]. – Режим доступа: https://doc.arduino.ua/
13. Компания производителя и разработчика Arduino: [офиц. сайт]. – Режим доступа: https://www.arduino.cc/
14. Фирма Амперка: [офиц. сайт]. – Режим доступа: http://amperka.ru/

15. Петин, В.А. Практическая энциклопедия Arduino / В.А. Петин, А.А. Биняковский. – М.: ДМК Пресс, 2017. – 152 с. – ISBN 987-5-97060-344-4

16. Петин, В.А. Проекты с использование контроллера Arduino / В.А. Петин. – Спб. БХВ-Петербург, 2015. – 464 с. – ISBN 987-5-9775-3550-2.

17. Уголок радиолюбителя: [сайт]. – Режим доступа: http://fornk.ru/

18. СМИ Сайт-ПАЯЛЬНИК: [Электронный ресурс]. – Режим доступа: http://cxem.net/arduino/

19. Соммер, У. Программирование микроконтроллерных плат Arduino/Freeduino / У. Соммер. – Спб. БХВ-Петербург, 2012. – 256 с. – ISBN 987-5-9775-0727-1.

20. Технический кружок ROBOTEXHИКА [Электронный ресурс]. – Режим доступа: http://xn--18-6kcdusowgbt1a4b.xn--p1ai/

Приложение А
(справочное)

Синтаксис языка C/C++

А.1 Переменные и константы

А.1.1 Типы

При выполнении программы всегда есть необходимость хранить какие-нибудь данные. Чаще всего данные хранятся либо в оперативной памяти микроконтроллера, либо непосредственно в программе. В первом случае имеется возможность изменять значения этих данных в процессе выполнения программы. Такие ячейки называют «переменные». Во втором случае данные фиксированы и не могут быть изменены, поэтому их называют «константы».

Пользователь сам определяет, какие типы ячеек следует использовать в программе согласно логическому построению будущей программы. Если данные не будут изменяться, то их следует оформить как константы. Тем самым высвобождаются ресурсы оперативной памяти для других данных. Примером константы может быть число Pi, которое определено значением 3.14159265…

Как переменные, так и константы могут содержать различные типы данных. От этого зависит и количество байт, необходимых для их хранения. Название, размерность и объявления на языке C++ наиболее часто употребляемых типов данных в Arduino IDE представлены в таблице А.1.

Выбор того или иного типа переменной или константы осуществляется самим программистом исходя из поставленной задачи и характера данных, оперируемых программой.

А.1.2 Записи (структуры)

Для удобства восприятия многочисленных переменных их можно логически группировать в некую структуру с единым именем. Чаще всего группируются переменные, относящиеся к одному конкретному объекту, действию или событию. Пример на языке C++:

Таблица А.1 – Типы переменных

Тип	Размер байт	C++ (объявление)
Целое без знака 0…255	1	`byte i;`
Целое без знака 0…65 535	2	`unsigned int a;`
Целое без знака 0… 4 294 967 295	4	`unsigned long b;`
Целое -128…127	1	`char i;`
Целое -32 768 .. 32 767	2	`int a;`
Целое -2 147 483648 … 2 147 483 647	4	`long b;`
С плавающей запятой (десятичные дроби, 7 знаков) $\pm 1.18 \cdot 10^{-38}$… $\pm 3.4 \cdot 10^{+38}$	4	`float c;`
Логический false (Нет)…true (Да)	1	`boolean f;`
Символ ‘0’, ‘a’, ‘D’, …	1	`char s;` `s = 'a'; // пример`
Строка символов	N	`char s[];` `char s[] = "aBc"; // пример`

```
typedef           // Служебное слово для обозначения
                  // пользовательского типа
struct            // Служебное слово, обозначающее
                  // начало структуры.
  {
    int s;        // 1-я переменная
    long v;       // 2-я переменная
    boolean a;    // 3-я переменная
  } tmyRec;       // Имя пользовательского типа tRec
tmyRec myREC;     // Определение переменной myREC типа tmyRec
```

В приведенном выше примере имя типа похоже на имя определяемой переменной за исключением буквы «t» в начале (от первой буквы служебного слова «Type»). Строго говоря, это требование не обязательное, так как имя типа и имя переменной могут совершенно не совпадать. Но для облегчения восприятия программы, когда это возможно, следует пользоваться данным правилом.

В самой программе доступ к той или иной переменной в записи (структуре) осуществляется следующим образом на языке C++:

```
myREC.s = 0;      // Имя записи и переменной в этой записи
myREC.s = myREC.s + myREC.v * 0.01;
```

А.1.3 Массивы

Иногда для решения поставленной задачи необходимо обработать последовательность переменных одного и того же типа (массивы). Например, для обработки статистических данных, поступающих с какого-нибудь датчика, эти данные удобно расположить последовательно друг за другом. Позиция данных в этой последовательности будет определять время их считывания.

Типы данных, которые могут быть помещены в массивы, различны. Это могут быть и стандартные типы (таблица А.1), и определенные пользователем, в том числе и записи (структуры). На языке C++:

```
Int S[10];                // Определение массива с
                          // именем S, состоящего из
                          // 10 элементов целого типа
                          // с индексами с 0 по 9
tmyREC myREC1 [20];       // Определение массива с
                          // именем myREC1, состоящего из
                          // 10 элементов типа tmyREC
                          // с индексами с 0 по 9
```

Доступ к конкретному элементу массива осуществляется с помощью индекса. На языке C++:

```
S[2] = 10;                  // Элементу массива S с индексом 2
                            // присваивается значение 10
myREC1[i].v = 1.5;          // Переменной v записи myREC1 с
                            // индексом по значению переменной i
                            // присваивается значение 1.5
```

Индекс массива всегда начинается с 0. Поэтому максимальный индекс массива равен его длине минус 1.

А.1.4 Локальные и глобальные переменные

Программа в общем случае состоит из набора процедур и функций, вызываемых при заранее определенных событиях, или пользователем при написании программы. В свою очередь каждую процедуру либо функцию можно рассматривать как отдельную программу со своим именем, своими переменными и константами. При этом имеет смысл разграничить области действия (и существования) некоторых переменных только в пределах этих подпрограмм. Например, если в конкретной подпрограмме используется переменная «i» в качестве счетчика цикла, то ее лучше объявить как локальную, действующую только в пределах этой подпрограммы. Тем самым экономится ОЗУ микроконтроллера, так как переменная уничтожается при выходе из подпрограммы. С другой стороны, это же имя «i» можно будет использовать для объявления другой локальной переменной в другой подпрограмме. Если требуется хранение данных, доступных из любой подпрограммы пользовательской программы, то такие переменные объявляются глобальными и, соответственно, такое имя уже считается зарезервированным и недоступным для других переменных.

Пример объявления глобальных и локальных переменных и констант в программе на языке C++:

```
float Rev;                    // Объявление глоб. переменной Rev
                              //   вещественного типа
                              // Она будет доступной из любого места
                              //   программы

const float Pi = 3.14159265; // Объявление глобальной константы
                              //     типа с плавающей запятой

void loop()                   // Объявление функции loop()
{
  int I;                      // Объявление локальной переменной I
                              // Она будет доступной только внутри
                              //   функции loop()
  for (I=0;I<10;I++)
  {
    Rev = Mul(I*Pi, Rev);  // вызов пользовательской функции Mul
  };
};

float Mul(float A, B)         // Объявление пользовательской функции
                              //   Mul
{
  float I;                    // Объявление локальной переменной I
                              // Она будет доступной только внутри
                              //   функции Mul(). И никакого отношения
                              //  не имеет к переменной I в функции
                              //  loop()
  I = A * A;
  return I * B;               // Это произведение будет результатом
                              // выполнения функции
};
```

А.2 Указатели

Существует определенный класс переменных, называемых указателями. Аналогом указателя можно назвать оглавление в книге. Для того, чтобы найти определенную информацию сначала читатель по оглавлению находит название главы, параграфа, подпункта. Затем по номеру страницы переходит непосредственно к тексту.

Указатель есть переменная, которая указывает, в каком месте памяти искать информацию. По-другому - указатель хранит адрес в памяти, по которому находится начало информации. Таким образом, указателями удобно пользоваться для доступа к массивам однотипных данных.

На языке С++ работа с указателями осуществляется следующим образом:

```
float Polozenie (void)
{
  int i[4];        // Объявление массива i целого типа
  int *k;          // объявление указателя k на целый тип данных

  k = &i[0];       // Определение адреса переменной «i[0]»
  *k = 10;         // доступ к «i[0]» через указатель «k»
  k += 2;          // перемещение указателя вперед на 2 длины
                   // переменной целого типа, то есть доступ к
                   // другой переменной через указатель «k»
  *k = 12;         // доступ к «i[2]» через указатель «k»
    ……
};
```

А.3 Команды

Для выполнения каких-либо действий над данными используются команды. Правильное написание команд, необходимые сопутствующие слова и порядок представления данных называют синтаксисом.

Список наиболее часто употребляемых команд, назначение и различие в синтаксисах на языке С++ приведены в таблице А.2.

Таблица А.2 – Список команд С++

N п/п	С++	
1	Команда присвоения	
1	2	3

	`A = 10;`
	`B=B+1;`
	`B++;`
	`B += 1;`
	`D = D-10;`
	`D -= 10;`
		Переменной А присвоить 10
		К В прибавить 1 и сохранить в В
		Переменную В увеличить на 1
		От D вычесть 10 и сохранить в D

2	Арифметические команды	

	`+`
	`-`
	`/`
	`*`
	`(int)` AAA/BBB
	`%`
		сложение
		вычитание
		деление
		умножение
		целая часть от деления целого на целое
		остаток от деления целого на целое

	`B=D+C;` `A=(int) D/N;`	К D прибавить С и сохранить в В Результат деления D на N преобразовать в целый тип и сохранить в А

Продолжение таблицы А.2

3	Логические команды	
1	2	3

	`&&`
	`||`
	`!`
		логическое И
		логическое ИЛИ
		логическое отрицание

	`(!((A && B) || (C && D)))`	Пример логического выражения
4	Команды сравнения	

	`>`
	`<`
	`==`
	`>=`
	`<=`
	`!=`
		больше
		меньше
		равно
		больше либо равно
		меньше либо равно
		не равно

	`(A == B)`
	`((D != C) &&`
	`(N > 1))`
		Пример логического выражения

5	Битовые команды	

	`&`
	`|`
	`^`
	`<<`
	`>>`
		побитовое И
		побитовое ИЛИ
		побитовое исключающее ИЛИ
		битовый сдвиг влево
		битовый сдвиг вправо

	`A=11;`
	`B=21;`
	`C=A | B;`
	`C=C << 1;`
		Пример использование битовых команд
		
		// C равно 31
		// C равно 62

6	Условие (сравнение, ветвление)	

	`if (`*условие*`)`
	`{`
	`// когда условие`
	`// выполняется`
	`};`
		Логическое условие всегда пишется в круглых скобках

	`if (`*условие*`)`
	`{`
	`// когда условие`
	`// выполняется`
	`}`
	`else`
	`{`
	`// когда условие`
	`// не выполняется`
	`};`
		Логическое условие всегда пишется в круглых скобках
		
		точка с запятой НЕ ставится

Продолжение таблицы А.2

1	2	3
7	Выбор (переключатель)	

	switch (*Переменная выбора*)
	{
	  case *выбор1*:
	  // при равенстве
	  // переменной выбора
	  // выбор1
	  break;
	  case *выбор2*:
	  // при равенстве
	  // переменной выбора
	  // выбор2
	  break;
	  //...
	  default:
	  // если ничего
	  // не подошло**};**
		Если переключатель может иметь несколько значений, а не два как для if
		
		break обязателен, иначе программа будет выполняться далее, то есть команды при «выбор2»

8	Безусловный переход	
	goto *Метка;*	Метка должна где-то быть внутри функции
9	Цикл For	

	for (*Var*=*Min; Var*<*Max; Var*++)
	{
	  // тело цикла с увеличивающимся счетчиком
	};
		Min – начальное значение
		Max = конечное значение
		Var++ - как изменяется счетчик цикла

10	Цикл While	

	while (*условие выхода*)
	{
	  // тело цикла
	};
		В теле цикла должна изменяться некая переменная, по которой будет выполняться условие выход из цикла