Петин В. А.

77 ПРОЕКТОВ ДЛЯ ARDUINO

Москва, 2020

УДК 681.4:004.9Arduino
ББК 32.816с515+32.965с515
П29

Петин В.А.
П29 **77 проектов для Arduino. — М. ДМК Пресс. 2020. — 356 с.: ил.**

ISBN 978-5-97060-697-1

В книге представлено 77 экспериментов для Arduino — ценнейшего практического материала для обучения. Каждый эксперимент подразумевает поэтапное изучение электроники и программирования путем создания проектов на Ардуино. Процесс обучения от начала работы до готового устройства занимает не очень много времени. В рамках нашей программы обучения можно собрать полноценный проект. Например, игру «Змейка», домашнюю метеостанцию, WEB-опросник, бегущую строку, электронные часы с будильником, FM радио, электронный компас и многое другое!

Для сборки проектов не требуется паяльник, а порог вхождения в электронику очень легкий, что соответствует нашему девизу: «Arduino — это очень просто!»

Авторы книги использовали современные методики обучения. Книгу можно использовать, как методическое пособие. Она содержит подробные иллюстрации к каждому занятию. Теория полностью совмещена с практикой и излагается постепенно, от простого к сложному, не упуская всех деталей.

УДК 681.4:004.9Arduino
ББК 32.816с515+32.965с515

ISBN 978-5-97060-697-1

Содержание

Введение

Эта книга создавалась одновременно с набором «Лаборатория электроники и программирования. 77 проектов для Arduino». С этой книгой Вы освоите в теории, а с набором на практике основы программирования, конструирования электронных устройств и робототехники на основе контроллеров – плат Arduino и WiFi модулей ESP8266.

Arduino — это электронный контроллер и удобная платформа быстрой разработки электронных устройств для новичков и профессионалов. Платформа пользуется огромной популярностью во всем мире благодаря удобству и простоте языка программирования, а также открытой архитектуре и программному коду.

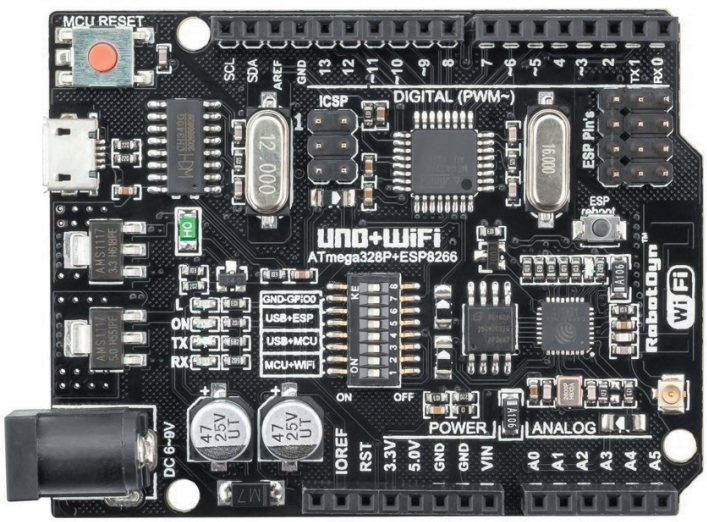


Рис. 1. Плата Arduino+WiFi от компании RobotDyn ESP8266

Появившиеся не так давно платы на основе WiFi модуля ESP8266 и представляющие собой полноценный 32 битный микроконтроллер ESP-8266EX со своим набором GPIO, в том числе SPI, UART, I2C, составляют на данный момент конкуренцию платам Arduino, учитывая низкую цену и возможность программировать устройства ESP8266 в среде Arduino IDE.

Основным элементом набора, является плата Arduino+WiFi (рис. 1), на которой интегрированы контроллер, совместимый с Arduino UNO R3 и WiFi-модуль ESP8266.

Arduino UNO и ESP8266 могут работать вместе или каждый в отдельности, необходимый режим можно установить с помощью находящихся на плате переключателей.

Процесс обучения программированию и конструированию будет проходить посредством создания проектов, начиная с простых, и заканчивая достаточно сложными, требующими достаточного уровня мастерства, которое будет повышаться от проекта к проекту.

В книге описаны, а в набор включены различные датчики, модули, средства отображения информации, источники питания. Для удобства подключения устройств к плате Arduino+WiFi в наборе присутсвует большая плата прототипирования и множество проводов.

Установка Arduino IDE

Установка программного обеспечения

Разработка собственных приложений на базе плат, совместимых с архитектурой Arduino, осуществляется в официальной бесплатной среде программирования Arduino IDE. Среда предназначена для написания, компиляции и загрузки собственных программ в память микроконтроллера, установленного на плате Arduino-совместимого устройства. Основой среды разработки является язык Processing/Wiring — это фактически обычный C++, дополненный простыми и понятными функциями для управления вводом/выводом на контактах. Существуют версии среды для операционых систем Windows, Mac OS и Linux.

При написании этой книги использовались версии Arduino IDE не ниже 1.6.5. Скачать Arduino IDE можно на официальном сайте **www.arduino.cc.**

Установка Arduino IDE в Windows

Отправляемся на страницу **https://www.arduino.cc/en/Main/OldSoftwareReleases#previous** (рис. 02), выбираем версию для операционной системы Windows и скачиваем архивный файл. Он занимает чуть более 80 Мбайт и содержит все необходимое, в том числе и драйверы. По окончании загрузки распаковываем скачанный файл в удобное для себя место.

Теперь необходимо установить драйверы. Подключаем Arduino к компьютеру. На контроллере должен загореться индикатор питания — зеленый светодиод. Windows начинает попытку установки драйвера, которая заканчивается сообщением **Программное обеспечение драйвера не было установлено.**

Открываем Диспетчер устройств. В составе устройств находим значок Arduino UNO — устройство отмечено восклицательным знаком. Щелкаем правой кнопкой мыши на значке Arduino UNO и в открывшемся окне выбираем пункт Обновить драйверы и далее пункт **Выполнить поиск драйверов на этом компьютере**. Указываем путь к драйверу — ту папку на компьютере, куда распаковывали скачанный архив. Пусть это будет папка drivers каталога установки Arduino — например, C:\arduino-1.6.5\drivers. Игнорируем все предупреждения Windows и получаем в результате сообщение **Обновление программного обеспечения**

Arduino 1.6.x, 1.5.x BETA

These packages are no longer supported by the development team.

1.8.5	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.8.4	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.8.3	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.8.2	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.8.1	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.8.0	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.13	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.12	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.11	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.10	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github

Рис. 02. *Страница загрузки всех версий Arduino IDE официального сайта Arduino*

для данного устройства завершено успешно. В заголовке окна будет указан и COM-порт, на который установлено устройство.

Осталось запустить среду разработки Arduino IDE (рис. 03). В списке доступных портов отображается название платы Arduino.

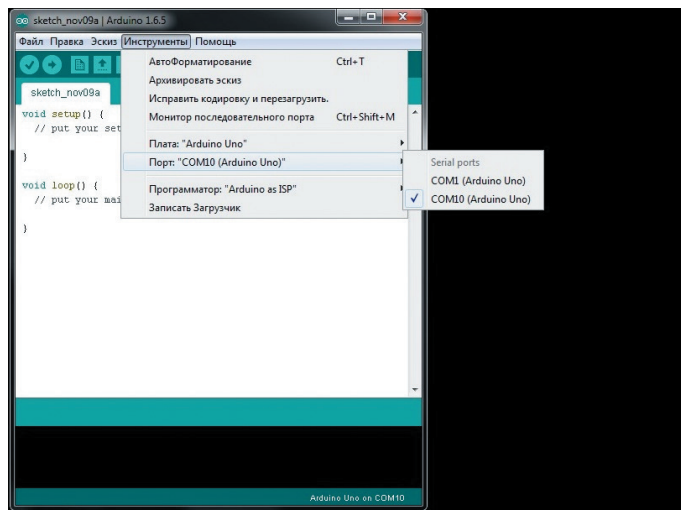


Рис. 03. Arduino IDE — среда разработки

Настройка среды Arduino IDE

Среда разработки Arduino состоит (рис. 04) из:

- редактора программного кода;
- области сообщений;
- окна вывода текста;
- панели инструментов с кнопками часто используемых команд;
- нескольких меню.

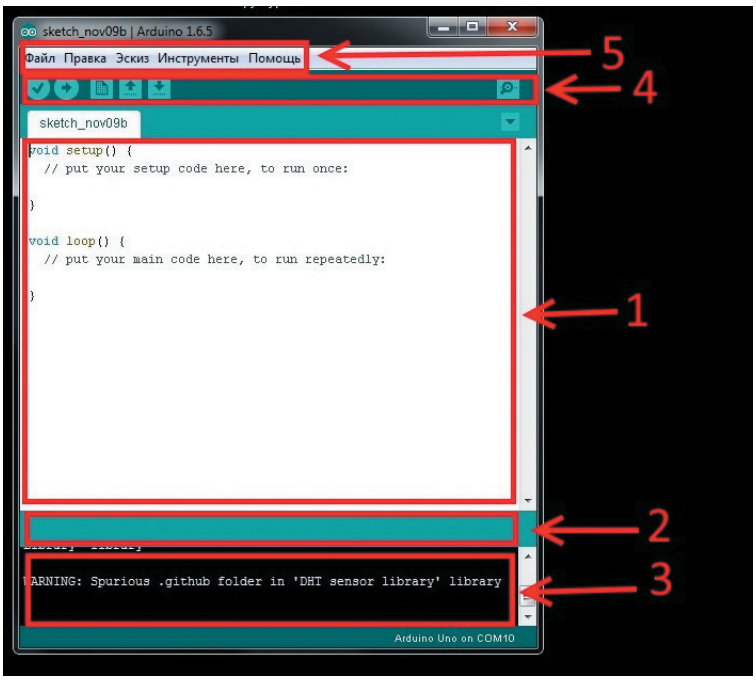


Рис. 04. Окно Arduino IDE

Программа, написанная в среде Arduino, носит название скетч. Скетч пишется в текстовом редакторе, который имеет цветовую подсветку создаваемого программного кода. Во время сохранения и экспорта проекта в области сообщений появляются пояснения и информация об ошибках. Окно вывода текста показывает сообщения Arduino, включающие полные отчеты об ошибках и другую информацию. Кнопки панели инструментов позволяют проверить и записать программу, создать, открыть и сохранить скетч, открыть мониторинг последовательной шины.

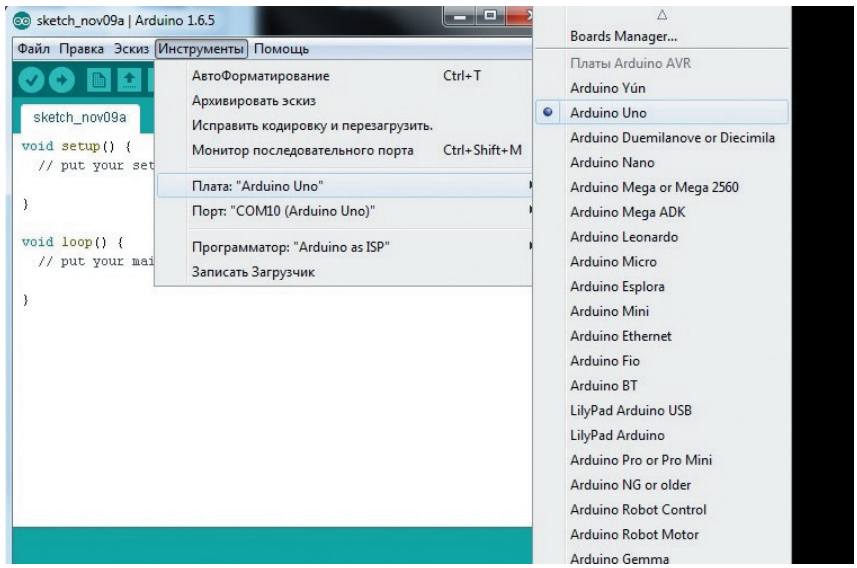


Рис. 05. Arduino IDE — выбор платы

Разрабатываемым скетчам дополнительная функциональность может быть добавлена с помощью библиотек, представляющих собой специальным образом оформленный программный код, реализующий некоторый функционал, который можно подключить к создаваемому проекту. Специализированных библиотек существует множество. Обычно библиотеки пишутся так, чтобы упростить решение той или иной задачи и скрыть от разработчика детали программно-аппаратной реализации. Среда Arduino IDE поставляется с набором стандартных библиотек: Serial, EEPROM, SPI, Wire и др. Они находятся в подкаталоге libraries каталога установки Arduino. Необходимые библиотеки могут быть также загружены с различных ресурсов. Папка библиотеки копируется в каталог стандартных библиотек (подкаталог libraries каталога установки Arduino). Внутри каталога с именем библиотеки находятся файлы *.cpp, *.h. Многие библиотеки снабжаются примерами, расположенными в папке examples. Если библиотека установлена правильно, то она появляется в меню **Эскиз | Импорт библиотек**. Выбор библиотеки в меню приведет к добавлению в исходный код строчки:

#include <имя библиотеки.h>

Эта директива подключает заголовочный файл с описанием объектов, функций и констант библиотеки, которые теперь могут быть использованы в проекте. Среда Arduino будет компилировать создаваемый проект вместе с указанной библиотекой.

Перед загрузкой скетча требуется задать необходимые параметры в меню **Инструменты | Плата** — как показано на рис. 05, и **Инструменты | Последовательный порт**— показано на рис. 3.

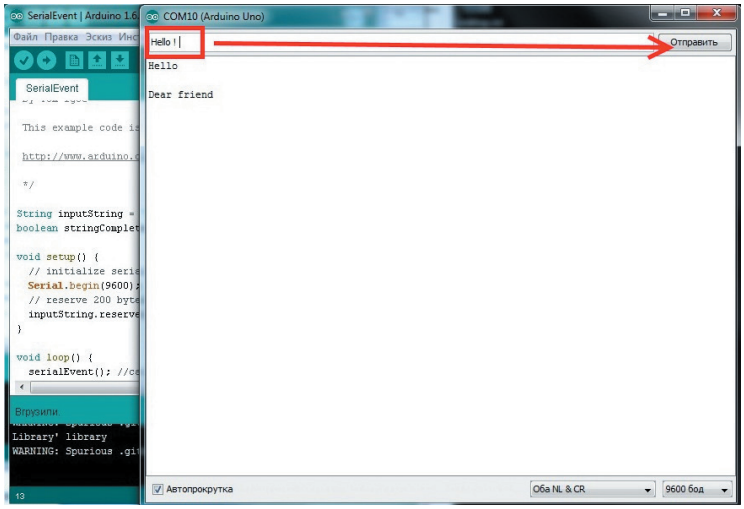


Рис. 06. Arduino IDE — монитор последовательного порта

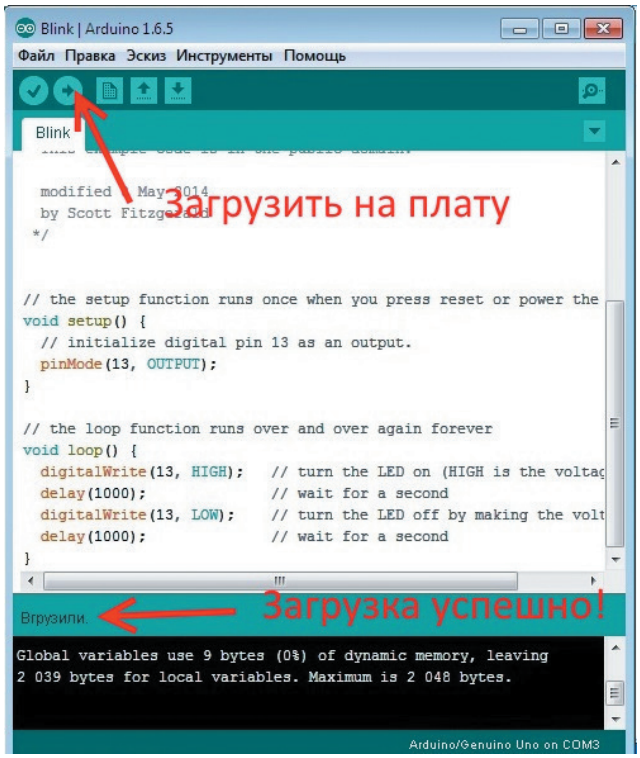


Рис. 07. v Загрузка скетча на плату Arduino

Современные платформы Arduino перезагружаются автоматически перед загрузкой. На старых платформах необходимо нажать кнопку перезагрузки. На большинстве плат во время процесса загрузки будут мигать светодиоды RX и TX.

При загрузке скетча используется загрузчик (bootloader) Arduino — небольшая программа, загружаемая в микроконтроллер на плате. Она позволяет загружать программный код без использования дополнительных аппаратных средств. Работа загрузчика распознается по миганию светодиода на цифровом выводе D13.

Монитор последовательного порта (Serial Monitor) отображает данные, посылаемые в платформу Arduino (плату USB или плату последовательной шины). Для отправки данных необходимо ввести в соответствующее поле текст и нажать кнопку **Послать** (Send) или клавишу <Enter> (рис. 06). Затем следует из выпадающего списка выбрать скорость передачи, соответствующую значению Serial.begin в скетче. На ОС Mac или Linux при подключении мониторинга последовательной шины платформа Arduino будет перезагружена (скетч начнется сначала).

Текст программы (скетч) пишется в окне редактора программного кода. В программе обязательно должны быть две записи, void setup() и void loop() (см. рис. 07) – это так называемые функции, первая выполняется единоразово, при подаче питания на Arduino, а вторая выполняется циклически до тех пор, пока присутствует питание микроконтроллера. В функцию setup() записываются различные настройки микроконтроллера для дальнейшей работы — например, это может быть конфигурация портов ввода/вывода, либо инициализация подключенного вами дисплея или датчика. Главное, что нужно запомнить, с этой функции начинается работа микроконтроллера и все, что в ней написано, выполняется только один раз. Функция loop() выполняется сразу же после функции setup(), и после этого микроконтроллер постоянно работает в ней.

Теперь загрузим на плату Arduino какой-нибудь скетч. Мы можем найти примеры скетчей в пункте меню Файл → Образцы, например **Файл → Образцы → Basics → Blink**. Для загрузки скетча на плату Arduino нажимаем на значок загрузки в панели инструментов (рис. 1.20) и в случае успешной загрузки скетча на плату в окне сообщений появится надпись **Вгрузили** (рис. 07).

Результат работы программы – мигание светодиода, подключенного к цифровому выводу 13.

Установка Arduino IDE для ESP8266

Arduino IDE для ESP8266 позволяет писать скетчи и загружать их одним кликом в ESP8266 в знакомой среде Arduino IDE. Рассмотрим установку Arduino IDE для ESP8266.

Сначала необходимо установить Arduino IDE с официального сайта версии не ниже 1.6.5. Запускаем Arduino IDE. Выбираем пункт **Файл → Настройки** и в поле

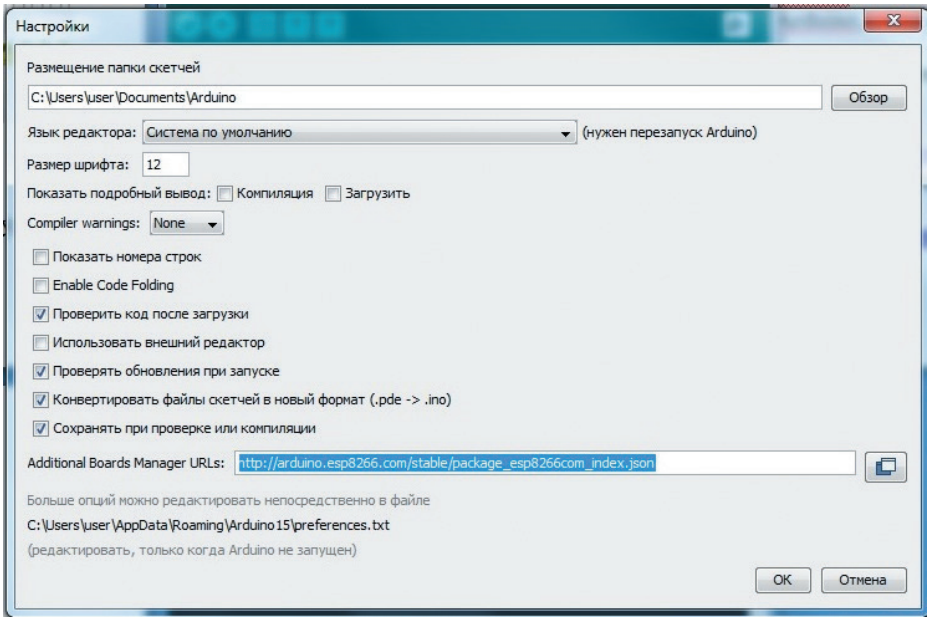


Рис. 08. Ввод адреса для скачивания Arduino IDE для ESP8266

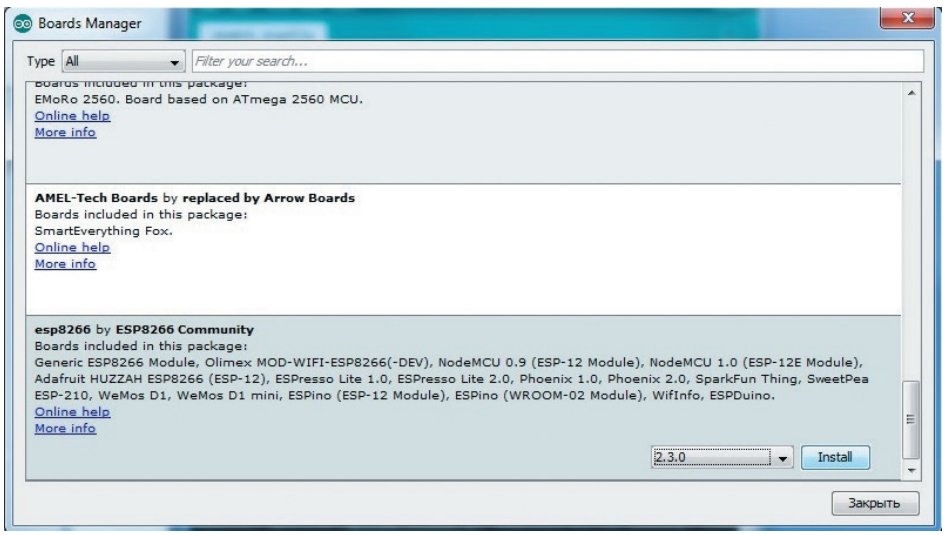


Рис. 09. Загрузка Arduino IDE для ESP8266

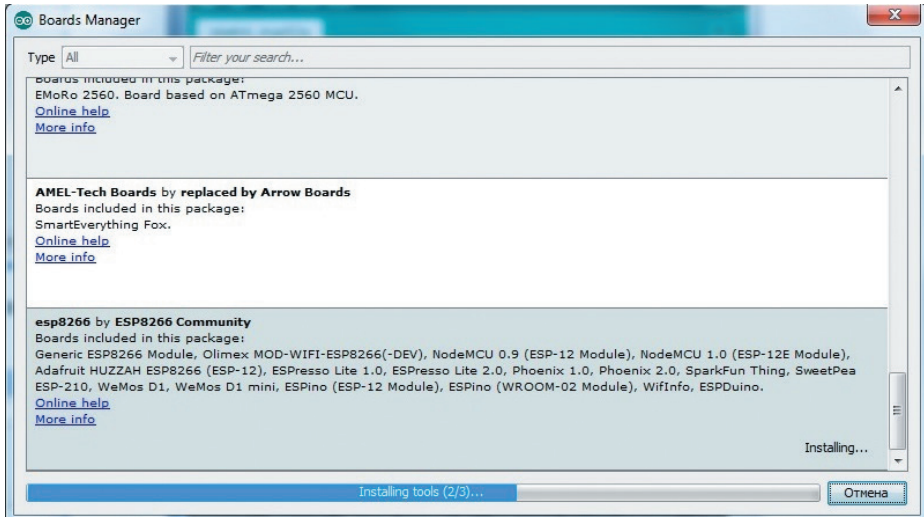


Рис. 010. Загрузка Arduino IDE для ESP8266

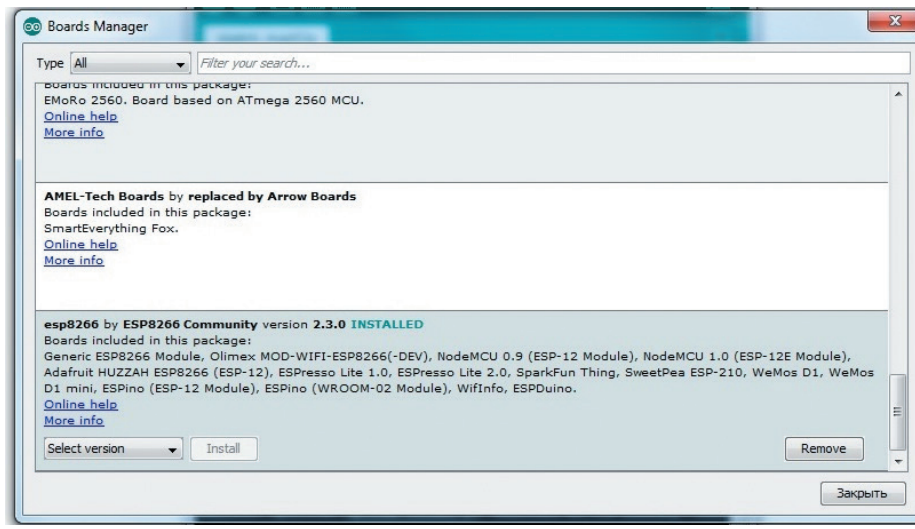


Рис. 011. Загрузка Arduino IDE для ESP8266

Additional Boards Manager URLs вводим **http://arduino.esp8266.com/stable/package_esp8266com_index.json.** Нажимаем **OK** (см. рис. 08).

Выбираем пункт **Инструменты → Плата → BoardsManager** и в списке ищем плату ESP8266. Выбираем этот пункт, версию и нажимаем на **Install** (см. рис. 09, 010, 011).

После загрузки программного обеспечения в списке плат (**Инструменты → Плата →**) появятся платы ESP8266 (см. рис. 012).

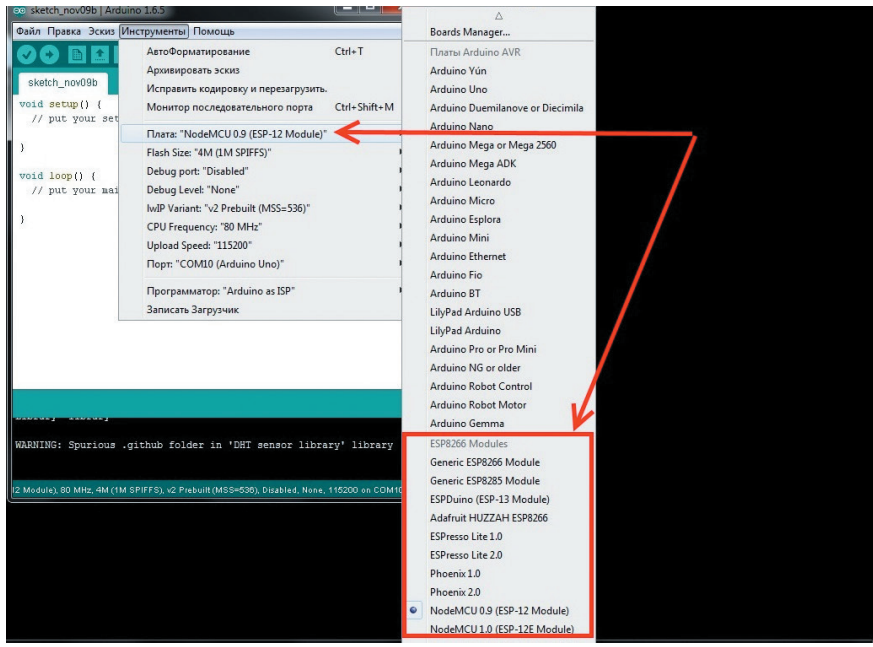


Рис. 012. Выбор платы ESP8266

Плата Arduino+WiFi

Рассмотрим более подробно основной элемент набора – плату Arduino+WiFi от компании RobotDyn (рис. 1). На данной плате интегрированы Arduino UNO R3 и WiFi-модуль ESP8266.

Arduino UNO и ESP8266 могут работать вместе или каждый в отдельности. Это решение для разработки новых проектов, требующих UNO и Wi-Fi. Через USB вы можете обновлять скетчи и прошивки как для ATmega328, так и для ESP8266, для этого на борту есть USB-serial конвертер CH340G. Необходимый режим можно установить с помощью находящихся на плате переключателей (рис. 013).

В таблице 1 представлены все возможные режимы работы платы и показаны все возможные положения переключателей на ней (переключатель 8 не используется).

Рис. 012. Переключатели на плате Arduino+WiFi от компании RobotDyn

РЕЖИМ	ПЕРЕКЛЮЧАТЕЛЬ						
	1	2	3	4	5	6	7
ATmega328 ←→ESP8266	*ON*	*ON*	*OFF*	*OFF*	*OFF*	*OFF*	*OFF*
USB←→ATmega328	*OFF*	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*
USB←→ ESP8266 (Обновление прошивки или эскиз)	*OFF*	*OFF*	*OFF*	*OFF*	*ON*	*ON*	*ON*
USB←→ ESP8266 (сообщение)	*OFF*	*OFF*	*OFF*	*OFF*	*ON*	*ON*	*OFF*
все независимые	*OFF*	*OFF*	*OFF*	*OFF*	*OFF*	*OFF*	*OFF*

Как видно из таблицы, если установить переключатели 3 и 4 в положение ON, а остальные в положение OFF, то мы получим обычную Arduino UNO. Чаще всего мы будем использовать этот режим.

Проводники и плата прототипирования

При сборке схем мы будем использовать беспаечную макетную плату (bread-board), которая позволяет осуществлять быстрый монтаж различных соединений и компонентов без необходимости использовать паяльник.

На рисунке 14 показан общий вид платы. Разным цветом обозначены шины-проводники. Синий цвет – это "-" схемы, красный – "+" , зеленый – это проводники, которые вы можете использовать по своему усмотрению для соединений частей электрической схемы, собираемой на макетной плате. Обратите внимание, что центральные отверстия соединены параллельными рядами поперек макетной платы, а не вдоль, в отличие от шин питания, которые размещены по краям макетных плат. На нашей макетной плате имеется несколько пар шин питания, что позволяет при необходимости подавать на плату разные напряжения, например, 5 В и 3,3 В. Группы поперечных проводников разделены широкой бороздкой. Благодаря этому углублению, на макетную плату можно устанавливать микросхемы в DIP-корпусах. Цифры и буквы на макетной плате нужны для того, чтобы вам легче было ориентироваться на плате.

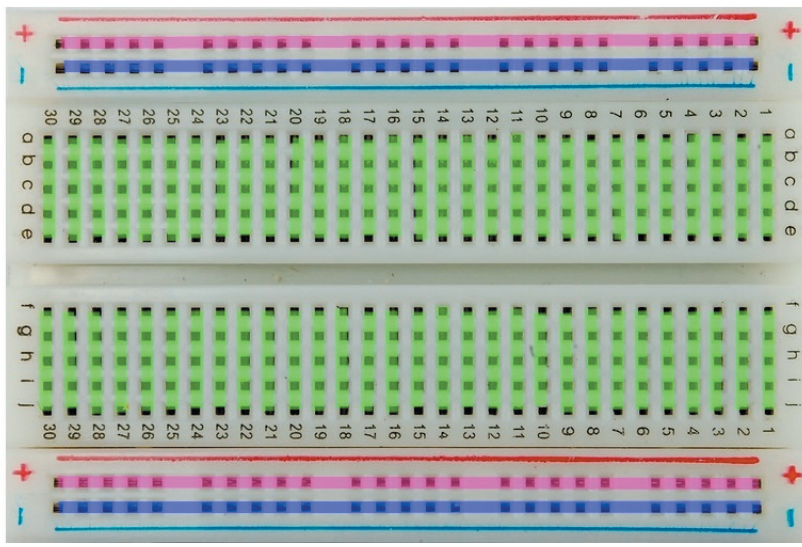


Рис. 014. Расположение шин контактов на макетной плате

Для сборки схем с помощью макетной платы, будем использовать проводники MM (папа-папа), MF (папа-мама) (рис. 15) и перемычки (рис. 16).

Рис. 015. Проводники MM, MF

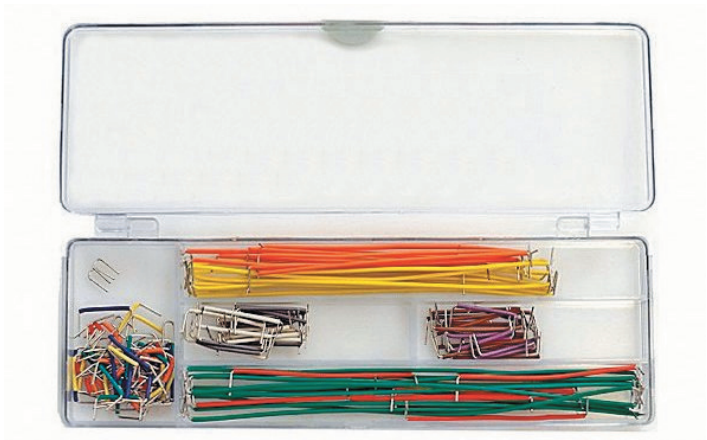


Рис. 016. Набор перемычек

Блоки питания

При сборке схем мы будем использовать беспаечную макетную плату (bread-board), которая позволяет осуществлять быстрый монтаж различных соединений и компонентов без необходимости использовать паяльник.

Рабочее напряжение плат Arduino 5 В. На плате установлен стабилизатор напряжения, поэтому на вход можно подавать питание с разных источников. Кроме этого, плату можно запитывать с USB - устройств. Источник питания выбирается автоматически. Рекомендуемое напряжение внешнего питания от 7 до 12 В. Максимальное напряжение 20 В, но значение выше 12 В с высокой долей вероятности быстро выведет плату из строя. Напряжение менее 7 В может привести к нестабильной работе, т.к. на входном каскаде может запросто теряться 1-2 В. Для подключения питания может использоваться встроенный разъем DC 2.1 мм или напрямую вход VIN для подключения источника с помощью проводов.

В качестве внешнего источника питания в наборе используется батарейный блок с двумя аккумуляторами 18650 емкостью 2200 мА/ч. Чтобы аккумуляторы служили долго, не допускайте разрядки их до напряжения ниже 3,7 В, и вовремя ставьте на зарядку!

Полный заряд будет выдавать напряжение 8,4 В. Большинство датчиков из набора требуют питание 5 В, которое мы будем брать с выхода +5В платы Arduino+WiFi. Однако в наборе присутствуют компоненты, требующие внешнего питания +5В. Для понижения напряжения с батарей будем использовать присутствующий в наборе преобразователь напряжения LM2596 с индикатором и подстроечным резистором.

При необходимости, можно использовать любой DC блок питания с подключением к разъему DC 2.1 мм.

Эксперимент 1.
Светодиодный маячок на 4 светодиодах

Здесь мы познакомимся с платой Arduino,
узнаем о режимах работы цифровых контактов,
научимся правильно подключать к Arduino светодиоды
и создадим первую программу
светодиодного маячка на 4 светодиодах.

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino + WiFi – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Светодиод красный - 4;
- ❑ Резистор 220 Ом – 4;
- ❑ Провода ММ – 5.

ПЕРЕКЛЮЧАТЕЛЬ						
1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Итак, установите переключатели на плате Arduino+WiFi следующим образом:

В результате получаем обычную Arduino UNO. Данное положение переключателей мы и будем использовать в большинстве экспериментов.

Arduino UNO представляет собой плату, с размещенными на ней компонентами, главным из которых является микроконтроллер Atmel AVR. Он является основной вычислительной системой этой платформы, поскольку именно для него и создается программное обеспечение, с помощью которого микроконтроллер взаимодействует с внешним миром посредством специальных портов ввода/вывода данных. Плата Arduino UNO имеет 14 цифровых вход/выходов (6 из которых могут использоваться как выходы ШИМ), 6 аналоговых входов, кварцевый генератор 16 МГц, разъем USB, силовой разъем, разъем ICSP и кнопку перезагрузки. Питание платы Arduino UNO при помощи адаптера AC/DC (рекомендуемое напряжение 7-12В), либо от компьютера посредством кабеля USB. Микроконтроллер ATmega328 располагает 32 кБ флэш-памяти, из которых 0.5 кБ используется для хранения загрузчика, а также 2 кБ ОЗУ (SRAM) и 1 Кб энергонезависимой памяти EEPROM.

Цифровые выводы на платах Arduino позволяют подключать к Arduino датчики, приводы и другие микросхемы. Изучение того, как использовать их, позволит вам использовать Arduino для выполнения практических полезных вещей.

Цифровые сигналы имеют только два отдельных значения: высокий (HIGH, 1) и низкий (LOW, 0) уровни. Вы можете использовать цифровые сигналы в ситуациях, где вход или выход будет принимать одно из этих двух значений. Поскольку цифровые выводы Arduino могут использоваться в качестве и входа, и выхода, сначала необходимо их настроить. Для настройки цифровых выводов в Arduino используется встроенная функция pimMode(), которая имеет следующий синтаксис:

pinMode(pin, mode)

где

- pin – номер вывода Arduino;
- mode – устанавливаемый режим для вывода pin:
  - INPUT – pin в режиме входа;
  - OUTPUT – pin в режиме выхода;
  - INPUT_PULLUP – в этом режиме к выводу подключается внутренний подтягивающий резистор 20 кОм, чтобы привести уровень на выводе к значению HIGH, если к нему ничего не подключено.

В данном эксперименте мы будем использовать выводы Arduino в режимы выходов (OUTPUT), для включения и выключения светодиодов. Светодиод – это полупроводниковый прибор, преобразующий электрический ток непосредственно в световое излучение. Цветовые характеристики светодиодов зависят от химического состава использованного в нем полупроводника. Светодиод излучает в узкой части спектра, его цвет чист, что особенно ценят дизайнеры. Светодиоды поляризованы, имеет значение, в каком направлении подключать их. Положительный вывод светодиода (более длинный) называется анодом, отрицательный – катодом. Как и все диоды, светодиоды позволяют току течь только в одном направлении – от анода к катоду. Поскольку ток протекает от положительного к отрицательному, анод светодиода должен быть подключен к цифровому сигналу, а катод должен быть подключен к земле.

Собираем схему согласно рис. 1.1.

В схеме подключения светодиодов к цифровым выходам мы используем ограничительный резистор номиналом 220 Ом. Рассмотрим, как подобрать ограничительный резистор и как будет влиять номинал резистора на яркость светодиода.

Самым главным уравнением для любого инженера-электрика является закон Ома. Закон Ома определяет отношения между напряжением, током и сопротивлением в цепи. Закон Ома определяется следующим образом:

$$V = I \times R,$$

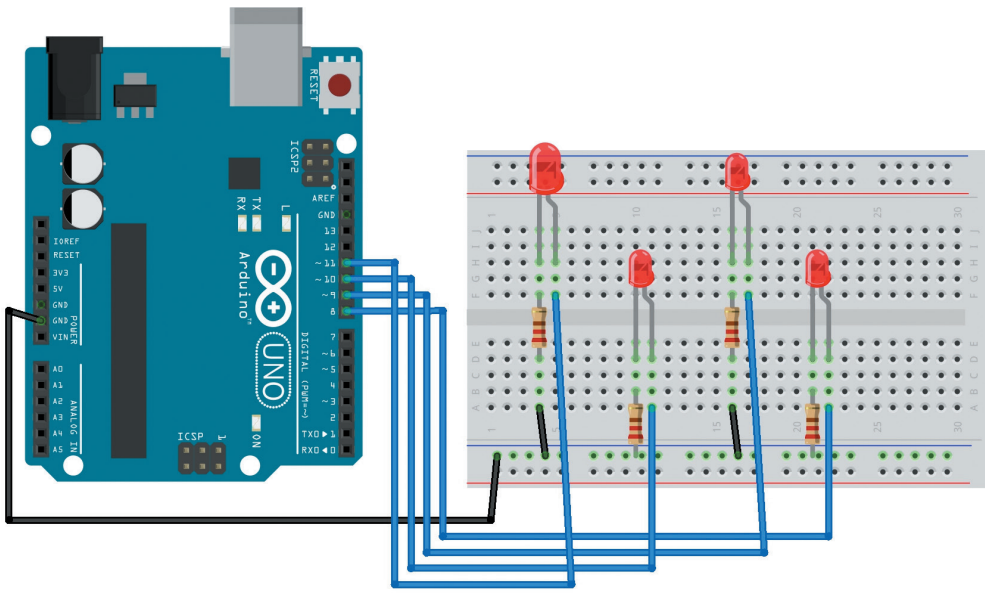


Рис. 1.1. Схема соединений для эксперимента

где V – напряжение в вольтах; I – ток в амперах; R – сопротивление в омах.

В электрической схеме каждый компонент имеет некоторое сопротивление, что снижает напряжение. Светодиоды имеют предопределенное падение напряжения на них и предназначены для работы в определенном значении тока. Чем больше ток через светодиод, тем ярче светодиод светится, до предельного значения. Для наиболее распространенных светодиодов максимальный ток составляет 20 мА. Обычное значение падения напряжения для светодиода – около 2 В. Напряжение питания 5 В должно упасть на светодиоде и резисторе, поскольку доля светодиода 2 В оставшиеся 3 В должны упасть на резисторе. Зная максимальное значение прямого тока через светодиод (20 мА), можете найти номинал резистора.

$$R = V/I = 3/0{,}02 = 150 \text{ Ом}$$

Таким образом, со значением резистора 150 Ом ток 20 мА протекает через резистор и светодиод. По мере увеличения значения сопротивления ток будет уменьшаться. 220 Ом немного более, чем 150 Ом, но все же позволяет светиться светодиоду достаточно ярко, и резистор такого номинала очень распространен.

Приступим к написанию программы (скетча).

Светодиоды должны одновременно мигать с определенной частотой. В процедуре setup() настроим режим работы контактов (пинов), к которым подключены светодиоды, как OUTPUT (выход)

```
void setup() {
   pinMode(8,OUTPUT);
   pinMode(9,OUTPUT);
   pinMode(10,OUTPUT);
   pinMode(11,OUTPUT);
}
```

В процедуре loop() сначала зажигаем все светодиоды (подаем на выводы 8,9,10,11 сигнал HIGH), ждем некоторое время (время "горения"), затем "тушим" светодиоды (подаем на выводы 8,9,10,11 сигнал LOW) и опять ждем некоторое время и т.д. по кругу. Для выполнения операции подождать "некоторое время" мы будем использовать встроенную Arduino функцию delay(). Эта функция просто останавливает выполнение программы на заданное в параметре количество миллисекунд (1000 миллисекунд в 1 секунде). Например:

delay(2000); // останавливает выполнение программы на 2000 мсек (

И весь код программы в листинге 1.1. Скачать данный скетч можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_01_01.

Листинг 1.1

```
void setup() {
   // настроить выводы 8, 9, 10, 11 Arduino как OUTPUT
   pinMode(8,OUTPUT);
   pinMode(9,OUTPUT);
   pinMode(10,OUTPUT);
   pinMode(11,OUTPUT);
}

void loop() {
  // включить светодиоды
  digitalWrite(8, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(11, HIGH);
  // пауза 1000 мсек (1 сек)
  delay(1000);
  // выключить светодиоды
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, LOW);
  // пауза 1000 мсек (1 сек)
  delay(1000);
}
```

Загрузим данный скетч на плату Arduino. Вы должны наблюдать включение/выключение светодиодов с частотой 2 секунды.

Эксперимент 2.
Бегущий огонек на 8-ми светодиодах

В данном эксперименте мы создадим "бегущий огонек" на ленте из 8-ми светодиодов, научимся создавать собственные функции для сокращения кода скетча.

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Светодиод – 8;
- ❑ Резистор 220 Ом – 8;
- ❑ Провода ММ – 5.

Переключатели на плате Arduino+WiFi установите следующим образом:

ПЕРЕКЛЮЧАТЕЛЬ						
1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Собираем схему согласно рис. 2.1

Приступим к написанию программы (скетча).

В процедуре setup() настроим режим работы контактов (пинов), к которым подключены светодиоды, как OUTPUT (выход)

```
void setup() {
   pinMode(5,OUTPUT);
   pinMode(6,OUTPUT);
   pinMode(7,OUTPUT);
   pinMode(8,OUTPUT);
   pinMode(9,OUTPUT);
   pinMode(10,OUTPUT);
   pinMode(11,OUTPUT);
   pinMode(12,OUTPUT);
}
```

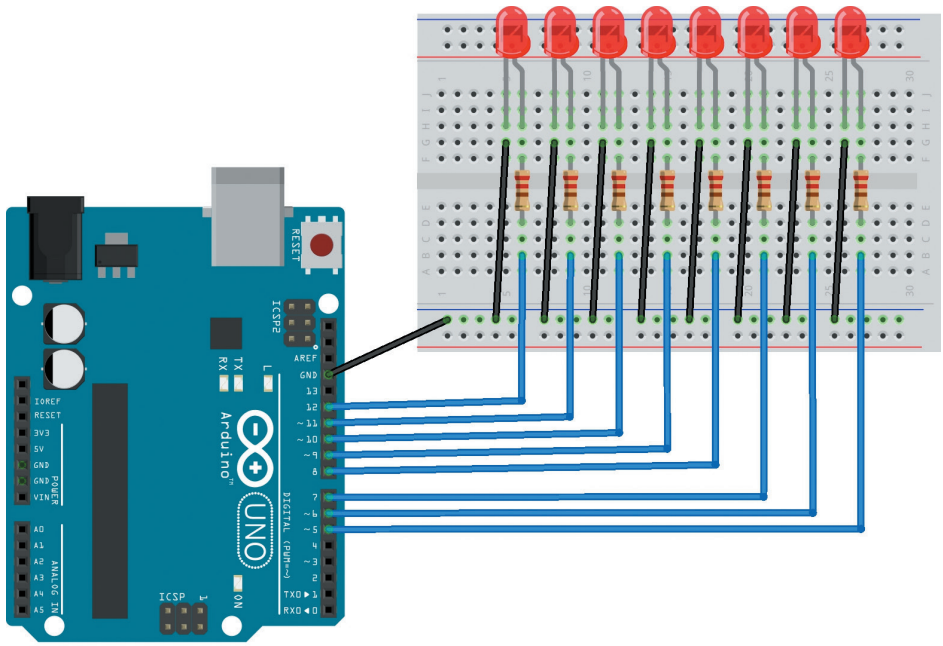


Рис. 2.1. Схема соединений для «бегущего огня» на 8 светодиодах

Алгоритм основного цикла следующий:

1. “гасим” все 8 светодиодов;
2. включаем первый светодиод;
3. включаем в скетче продолжительную паузу, чтобы увидеть горение одного светодиода;
4. повторяем шаги 1-3 для второго, третьего и далее до восьмого светодиода;
5. далее возврат в начало цикла loop().

Сокращенный код цикла loop() представлен в листинге 2.1.

Листинг 1.1

```
void loop() {
  // выключить 8 светодиодов
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, LOW);
  digitalWrite(12, LOW);
  // включить 1 светодиод
```

```
  digitalWrite(5, HIGH);
  // пауза 1000 мсек
  delay(1000);
  // выключить 8 светодиодов
  ….
  // включить 2 светодиод
  digitalWrite(6, HIGH);
  // пауза
  delay(1000);
  // выключить 8 светодиодов
  ….
  // включить 3 светодиод
  digitalWrite(7, HIGH);
  // пауза
  delay(1000);
  …….
  …….
  // выключить 8 светодиодов
  ….
  // включить 8 светодиод
  digitalWrite(12, HIGH);
  // пауза
  delay(1000);
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_02_01.**

Загрузим данный скетч на плату Arduino. Вы должны наблюдать бугущий огонек, перемещающийся по светодиодам с 1 по 8.

При том, что скетч работает верно, у него есть большой недостаток – он очень громоздкий. Рассмотрим, как можно код значительно уменьшить с помощью функций. Функция – это именованная последовательность операций. Причиной создания функции является необходимость выполнять одинаковое действие несколько раз. В цикле loop() восемь раз встречается совершенно одинаковый код для выключения 8 светодиодов. Создадим функцию ledoff() (выключить светодиоды). Функция создается за скобками функций setup(0 и loop(). Создание функции:

```
void ledsOff() {
     // код функции
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, LOW);
  digitalWrite(12, LOW);
}
```

И тогда цикл loop() нашего скетча сократится следующим образом (листинг 2.2)

Листинг 2.2

```
void loop() {
  // выключить 8 светодиодов
  ledsOff();
  // включить 1 светодиод
  digitalWrite(5, HIGH);
  // пауза 1000 мсек
  delay(1000);
  //
  ledsOff();
  digitalWrite(6, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(7, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(8, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(9, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(10, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(11, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(12, HIGH);
  delay(1000);
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_02_02.

Как видно, ввведение функции значительно сократило программу. Здесь мы рассмотрели простейший пример функции, в дальнейшем научимся передавать в функцию параметры и возвращать из функции в программу значения.

Эксперимент 3.
Бегущий огонек на 8 светодиодах – совершенствуем программу

Сегодня мы усовершествуем скетч "бегущего огня" на ленте из 8 светодиодов, научимся создавать константы и переменные и использовать их в программе.

В эксперименте мы будем использовать компоненты из эксперимента 2, но собрав схему соединений рис. 3.1.

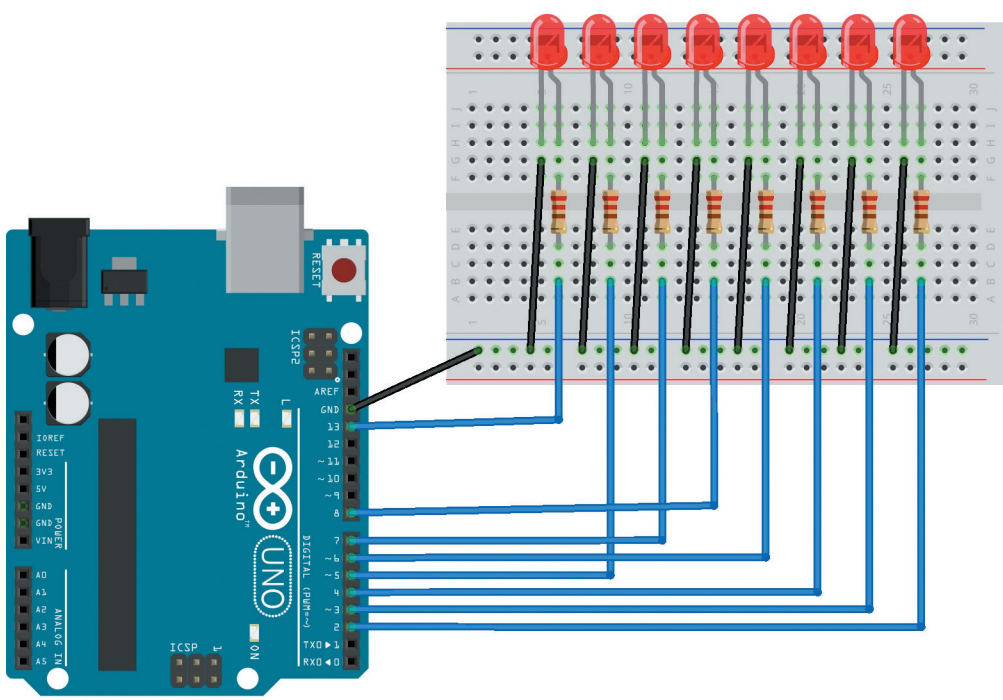


Рис. 3.1. Схема соединений для «бегущего огня» на 8 светодиодах

У нас есть работающий скетч **https://arduino-kit.ru/scetches/exp_02_02.**

Нам необходимо только изменить контакты для подключения светодиодов (например, с 5,6,7,8,9,10,11,12 на 2,3,4,6,8,7,5,13).

Можно сделать изменения в скетче, заменив значения 5,6,7,8,9,10,11,12 на 2,3,4,6,8,7,5,13 соответсвенно. Это 24 правки значений в скетче. В данном случае мы имеем дело с довольно простой программой и проделать изменения не так сложно. Но что если устройство довольно сложное и скетч расписан на сотни строк? В этом случае сделать все изменения правильно, не ошибиться и не пропустить ни одного изменения становится крайне сложно: мы люди и нам свойственно ошибаться из-за невнимательности.

Но можно поступить иначе. Можно каждому пину назначить понятное имя, которое потом используется для обращения:

```
#define LED1      2
#define LED2      3
#define LED3      4
#define LED4      6
#define LED5      8
#define LED6      7
#define LED7      5
#define LED8      13
```

И далее в скетче, например:

```
void setup() {
   pinMode(LED1,OUTPUT);
   pinMode(LED2,OUTPUT);
   pinMode(LED3,OUTPUT);
   pinMode(LED4,OUTPUT);
   pinMode(LED5,OUTPUT);
   pinMode(LED6,OUTPUT);
   pinMode(LED7,OUTPUT);
   pinMode(LED8,OUTPUT);
}
```

Конструкция #define называется макроопределением. Она говорит компилятору о том, что всякий раз, когда он видит указанное имя, стоит использовать на этом месте указанное значение.

Аналогично делаем изменения во всем скетче.

Скачать измененный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_03_01.**

Загрузим скетч на плату Arduino. Вы должны наблюдать бугущий огонек, перемещающийся по светодиодам с 1 по 8, подключенным по схеме на рисунке 3.1.

Все, теперь при изменении контактов подключения светодиодов (например, обратный переход к схеме соединений 2.1) достаточно изменить нужные значения в начале программы и не думать об изменениях в самом скетче.

Макроопределения хороши для именования значений, которые не могут измениться по ходу выполнения скетча. Но что делать, если какие-то параметры должны изменяться при выполнении программы? Для этого существуют переменные.

Переменная – это место хранения данных. Каждая переменная имеет имя, значение и тип. Например, объявление

int t = 1000;

создает переменную с именем t, значением 1000 и типом int (целое число в интервале -215 до 215). Затем, в программе имеется возможность обратиться к данной переменной через имя с целью работы с ее значением. Например:

t=t+100;

В качестве примера использования переменной напишем скетч «бегущего огня», в котором скорость движения (параметр в функции delay()) будет уменьшаться каждый цикл на 100 мсек до бесконечности. Изменение содержимого скетча показано в листинге 3.1.

Листинг 3.1.

```
int t=500;

void loop() {
  // выключить 8 светодиодов
  ledsOff();
  // включить 1 светодиод
  digitalWrite(LED1, HIGH);
  // пауза
  delay(t);
  //
  ledsOff();
  digitalWrite(LED2, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED3, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED4, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED5, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED6, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED7, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED8, HIGH);
  delay(t);
  t=t+100;
}
```

Эксперимент 4.
Десятисегментный линейный индикатор. Пульсирующая шкала

В данном эксперименте мы создадим "пульсирующую шкалу" на десятисегментном линейном индикаторе. При программировании познакомимся с массивами и научимся эффективно использовать массивы с помощью операторов for и if … else .

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Светодиод – 8;
- ❑ Резистор 220 Ом – 8;
- ❑ Провода ММ – 5.

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

10-разрядная линейная светодиодная шкала представляет собой сборку из 10 независимых светодиодов с катодами со стороны надписи на корпусе. Создадим на этой сборке проект "пульсирующей шкалы". Количество одновременно горящих светодиодов шкалы будет меняться от 0 до 8, затем опять с 0. Для подключения шкалы к Arduino будем использовать 10 цифровых выводов D3–D12. Схема соединений показана на рис. 4.1. Каждый из светодиодов шкалы выводом анода соединен с цифровым выводом Arduino, а катодом на землю через последовательно соединенный ограничивающий резистор 220 Ом.

Приступим к написанию программы (скетча). Определим константы для пинов подключения светодиодов в сборке и в процедуре setup() настроим режим работы контактов (пинов), к которым подключены светодиоды, как OUTPUT (выход) (см. листинг 4.1)

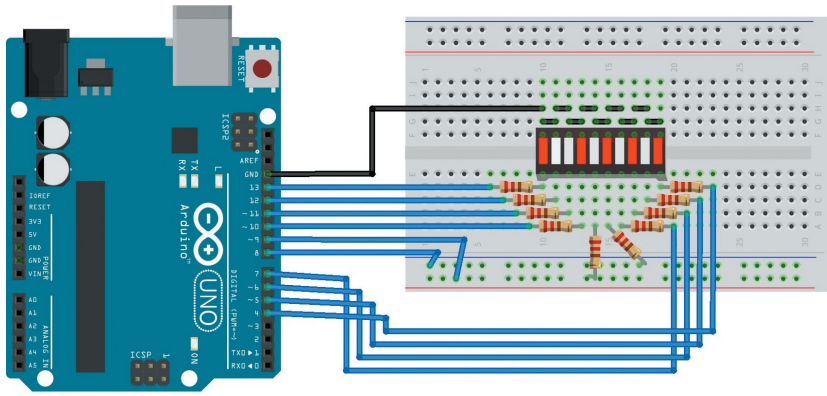


Рис. 4.1. Схема соединений для 10-разрядной линейной светодиодной шкалы

Листинг 4.1

```
#define LED1       13
#define LED2       12
#define LED3       11
#define LED4       10
#define LED5       9
#define LED6       8
#define LED7       7
#define LED7       6
#define LED8       5
#define LED8       4
void setup() {
   pinMode(LED1,OUTPUT);
   pinMode(LED2,OUTPUT);
   pinMode(LED3,OUTPUT);
   pinMode(LED4,OUTPUT);
   pinMode(LED5,OUTPUT);
   pinMode(LED6,OUTPUT);
   pinMode(LED7,OUTPUT);
   pinMode(LED8,OUTPUT);
   pinMode(LED9,OUTPUT);
   pinMode(LED10,OUTPUT);
}
```

И теперь самое время познакомиться с массивами. Массив - это набор переменных, доступ к которым осуществляется через их индекс. Рассмотрим варианты создания (объявления) массива.

Объявление массива без его инициализации:

```
int leds[10];         // массив из 10 элементов типа int
```

Можно объявить массив без непосредственного указания размера. Компилятор считает количество элементов и создает массив соответствующего размера:

```
int leds [] = {13,12,11,10,9,8,7,6,5,4};
```

Можно одновременно инициализировать и указать размер вашего массива:

```
int pin_leds [10] = {13,12,11,10,9,8,7,6,5,4};
```

Теперь рассмотрим доступ к элементам массива. Индексация в массивах начинается с нуля. То есть, первый элемент массива будет иметь порядковый номер 0. Таким образом:

```
pin_leds[0] = 13;
pin_leds[1] = 12;
```

Тогда код из листинга 4.1. можно записать в виде, представленном в листинге 4.2.

Листинг 4.2

```
int leds[] = {13,12,11,10,9,8,7,6,5,4};

void setup() {
   pinMode(leds[0],OUTPUT);
   pinMode(leds[1],OUTPUT);
   pinMode(leds[2],OUTPUT);
   pinMode(leds[3],OUTPUT);
   pinMode(leds[4],OUTPUT);
   pinMode(leds[5],OUTPUT);
   pinMode(leds[6],OUTPUT);
   pinMode(leds[7],OUTPUT);
   pinMode(leds[8],OUTPUT);
   pinMode(leds[9],OUTPUT);
}
```

Пока большого выигрыша в уменьшении написания кода мы не видим. Очень удобный метод работы с массивами - циклы. В этих случаях счетчик цикла используется для индексации каждого элемента массива. Конструкция for используется для повторения блока операторов, заключенных в фигурные скобки. Счетчик приращений обычно используется для приращения и завершения цикла. Пример 10-кратного выполнения операторов в цикле for:

```
int i=0;
for (; i < 10;) {
  // список операторов
  …..
  i=i+1;
}
```

или

```
for (int i=0; i < 10; i=i+1) {
  // список операторов
  …..
}
```

Оператор for подходит для любых повторяющихся действий и часто используется для перебора элементов массива. Например, код из листинга 4.2 будет выглядеть так:

```
int leds[] = {13,12,11,10,9,8,7,6,5,4};
void setup() {
   for (int i=0; i < 10; i=i+1) {
      pinMode(leds[i],OUTPUT);
   }
}
```

Теперь рассмотрим оператор if … else , который используется в сочетании с операторами сравнения, проверяет, достигнута ли истинность условия в if,

и выполняет блок операторов, заключенных в фигурные скобки, если условие не выполнено, выполняется блок операторов, заключенных в фигурные скобки else:

```
if(x<6) {
  // список операторов
   ……
}
else {
  // список операторов
   ……
}
```

Для включения n светодиодов из массива:

```
   for (int i=0;i<10; i=i+1) {
      if(i<3)
         {digitalWrite(leds[i],HIGH);}
      else
         {digitalWrite(leds[i],LOW);}
   }
```

И полный код программы показан в листинге 4.3. Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_04_03.

Листинг 4.3

```
// массив контактов подключения светодиодов шкалы
int leds[] = {13,12,11,10,9,8,7,6,5,4};
// количество горящих светодиодов
int n=0;

void setup() {
   // настроить выводы Arduino как OUTPUT
   for (int i=0;i < 10; i=i+1) {
      pinMode(leds[i],OUTPUT);
   }
}

void loop() {
  // выключить n светодиодов
   for (int i=0;i<10; i=i+1) {
      if(i<n)
         {digitalWrite(leds[i],HIGH);}
      else
         {digitalWrite(leds[i],LOW);}
   }
  // изменить n - приращение на 1
  // и остаток от деления на 11 (после 10 -> 0)
  n=(n+1)%11;
}
```

Загружаем скетч на плату Arduino и наблюдаем пульсирующую шкалу светодиодов.

Эксперимент 5.
Два светофора на перекрестке

В данном эксперименте мы создадим систему двух светофоров на светодиодах для перекрестка, продолжим работать с массивами и научимся создавать функции с передачей в них параметров.

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Светодиод красный – 2;
- ❑ Светодиод желтый – 2;
- ❑ Светодиод зеленый – 2;
- ❑ Резистор 220 Ом – 6;
- ❑ Провода ММ – 11.

Переключатели на плате Arduino+WiFi установите следующим образом:

Сегодня создадим систему двух светофоров на перекрестке. В качестве светофора используем 3 светодиода – зеленый, желтый и красный. Соберем схему соединений согласно рис. 5.1.

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Приступим к написанию программы (скетча). В эксперименте 4 мы уже использовали массивы, определим массив для контактов подключения светодиодов и сейчас:

```
int pinleds[]={7,8,9,10,11,12};
```

Определим список возможных состояний для двух светофоров, и для каждого состояния создадим массив:

```
// [green1,yellow1,red1,green2,yellow2,red2]
// 1) 1 светофор - зеленый, 2 светофор - красный
int leds1[]={HIGH,LOW,LOW,LOW,LOW, HIGH};
// 2) 1 светофор – мигает желтый, горит зеленый,
//  2 светофор - мигает желтый, горит красный
int leds2[]={HIGH,(int)blinkyellow,LOW,LOW,(int)blinkyellow,HIGH};
// 3) 1 светофор - красный, 2 светофор - зеленый
```

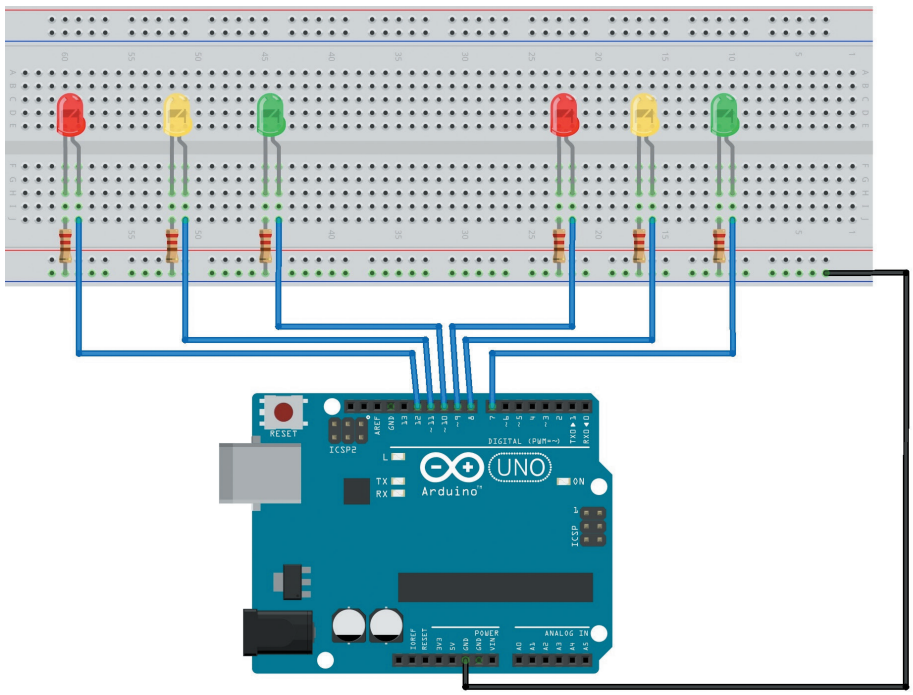


Рис. 5.1. Схема соединений для перекрестка на двух светофорах

```
   int leds3[]={LOW,LOW,HIGH,HIGH,LOW,LOW};
   // 4) 1 светофор - мигает желтый, горит красный,
   // 2 светофор - мигает желтый, горит зеленый
   int leds4[]={LOW,(int)blinkyellow,HIGH,HIGH,(int)blinkyellow,LOW};
```

Код для установки светофоров, например, в состояние 1, с использованием цикла for будет выглядеть следующим образом:

```
   for(int i=0;i<6;i++) {
      digitalWrite(pinleds[i], leds1[i]);
   }
```

Для второго состояния:

```
   for(int i=0;i<6;i++) {
      digitalWrite(pinleds[i], leds2[i]);
   }
```

По сути, мы видим повторяющийся код, отличие только в массивах состояний светодиодов leds1[], leds2[], а также leds3[], led4[] для 3 и 4 состояния. А для повторяющегося кода мы уже создавали функцию (см. Эксперимент 2). В данном случае мы также можем создать функцию, но только указывая ей с каким массивом состояний ей работать. Нужный массив передается в функцию и затем используется в ней. Записывается это так:

Вызов функции:

```
   trafficlight(leds1);
```

```
trafficlight(leds2);
trafficlight(leds3);
trafficlight(leds4);
```

Сама функция зажигания светодиодов светофора:

```
 void trafficlight(int leds[5]) {
    for(int i=0;i<6;i++) {
       digitalWrite(pinleds[i], leds[i]);
    }
 }
```

В первом вызове массив leds – это leds1, во втором – leds2, в третьем – leds3, в четвертом – leds4.

Теперь определим константы для времени "горения" красного, зеленого, и мигания желтого цвета, а также частоту мигания желтого цвета.

```
// время горения светодиодов в мсек
// зеленый
#define TIME_GREEN 15000
// красный
#define TIME_RED 15000
// желтый
#define TIME_YELLOW 5000
// период мигания желтого
#define TIME_BLINK 500
```

Переменную blinkyellow будем использовать для организации мигания желтого светодиода во время цикла, когда используется желтый светодиод.

```
boolean blinkyellow=true;
```

Переменная типа boolean имеет всего два значения true и false. Оператор

```
blinkyellow= !blinkyellow;
```

меняет значение переменной тина boolean на противоположное.

Код скетча для светофора представлен в листинге 5.1.

Листинг 5.1

```
// Выводы Arduino для подключения светодиодов
// [green1,yellow1,red1,green2,yellow2,red2]
int pinleds[6]={7,8,9,10,11,12};

// время горения светодиодов в мсек
// зеленый
#define TIME_GREEN 5000
// красный
#define TIME_RED 5000
// желтый
#define TIME_YELLOW 3000
// период мигания желтого
#define TIME_BLINK  300
// переменная blink для чередования мигания желтого
boolean blinkyellow=true;

void setup()
  {
```

```
  // настроить выводы Arduino как выходы
  // и потушить все светодиоды
  for(int i=0;i<6;i++)
     {
     pinMode(pinleds[i], OUTPUT);
     }
  }

void loop()
  {
  // 1 - зеленый, 2 - красный
  int leds1[]={HIGH,LOW,LOW,LOW,LOW, HIGH};
  trafficlight(leds1);
  delay(TIME_GREEN);
  blinkyellow=true;
  // 1 - желтый с зеленым, 2- желтый с красным
  for(int i=0;i<(TIME_YELLOW/TIME_BLINK);i++)
    {
     int leds2[]={HIGH,(int)blinkyellow,LOW,LOW,
                  (int)blinkyellow,HIGH};
     trafficlight(leds2);
     delay(TIME_BLINK);
     blinkyellow=!blinkyellow;
    }
  // 1 - красный, 2 - зеленый
  int leds3[]={LOW,LOW,HIGH,HIGH,LOW,LOW};
  trafficlight(leds3);
  delay(TIME_GREEN);
  blinkyellow=true;
  // 1- желтый с красным,  2 - желтый с зеленым
  //for(int i=0;i<(TIME_YELLOW/TIME_BLINK);i++)
  for(int i=0;i<TIME_YELLOW;i=i+TIME_BLINK)
    {
     int leds4[]={LOW,(int)blinkyellow,HIGH,HIGH,
                  (int)blinkyellow,LOW};
     trafficlight(leds4);
     delay(TIME_BLINK);
     blinkyellow=!blinkyellow;
    }
  }

 // функция зажигания светодиодов светофора
 void trafficlight(int statled[5])
   {
   for(int i=0;i<6;i++)
     {
     digitalWrite(pinleds[i],statled[i]);
     }
   }
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_05_01.**

Загрузим данный скетч на плату Arduino и проверяем работу светофоров.

Эксперимент 6.
Подключаем к Arduino кнопку

В данном эксперименте мы будем использовать цифровые контакты в качестве входов, и рассмотрим подключение кнопки к плате Arduino.

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Блок клавиатуры 4x4 – 1;
- ❑ Светодиод желтый – 2;
- ❑ Резистор 10 кОм – 1;
- ❑ Провода ММ – 3.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

В предыдущих экспериментах мы использовали цифровые контакты Arduino в качестве выходов. Сейчам рассмотрим использование этих контактов в качестве входов, позволит подключить, например, кнопку. Нажав на кнопку, мы подаем контроллеру сигнал, который затем приводит к каким-то действиям: включаются светодиоды, издаются звуки, запускаются моторы. Установка режима вывода, как входа, устанавливается с помощью функции pinMode().

```
pinMode(2,INPUT);          // вывод D2 установлен как выход
```

Кнопка – это достаточно простое устройство, замыкающее и размыкающее электрическую сеть. В наборе нет отдельной кнопки, но есть клавиатуры 4x4 (16 кнопок). Пока научимся работать с одной кнопкой.

Собираем схему согласно рис. 6.1

В этом случае мы работаем с кнопкой в левом верхнем углу. Один свободный проводник кнопки соединен с землей, другой – с цифровым выводом Arduino (рис. 1.28 ???). Но, это неправильно. В моменты, когда кнопка не замкнута, на цифровом

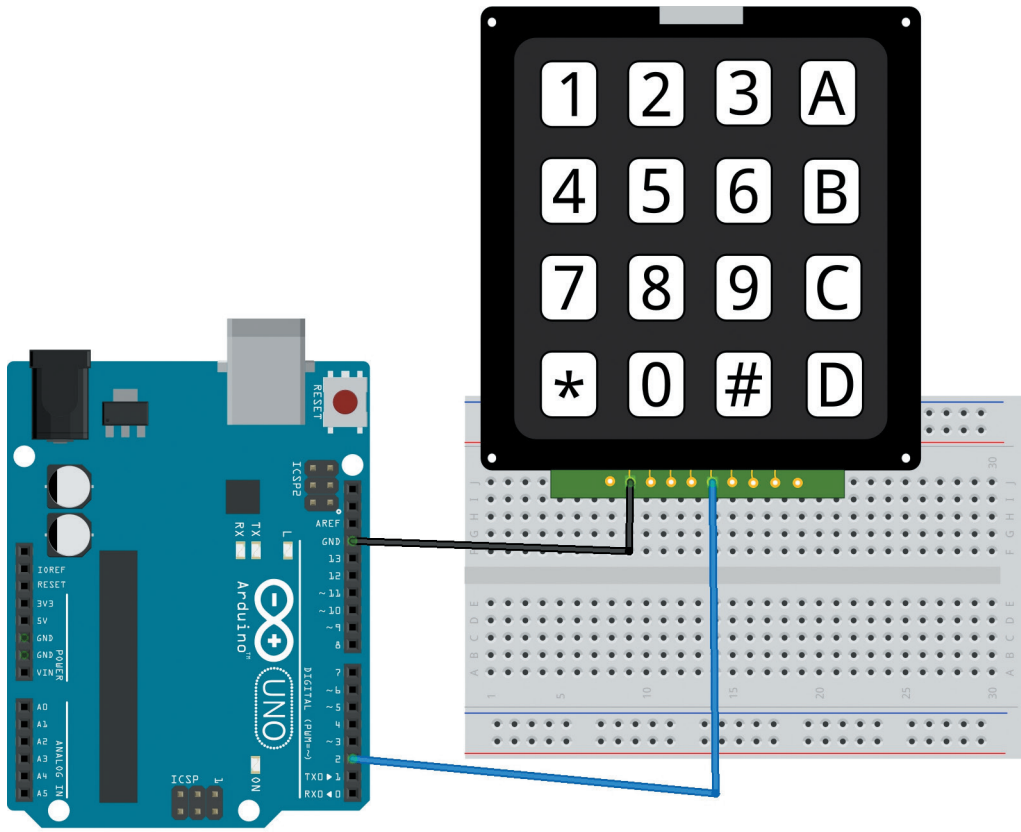


Рис. 6.1. Подсоединение к Arduino одной кнопки из клавиатуры 4x4

выводе Ардуино будут появляться электромагнитные наводки, и из-за этого возможны ложные срабатывания.

Чтобы избежать наводок, цифровой вывод обычно подключают через достаточно большой резистор (5-10 кОм) либо к питанию, либо к земле. В первом случае это называется “схема с подтягивающим резистором”, во втором – “схема со стягивающим резистором” (рис. 6.2). Резисторы в обеих схемах используются для установки “значения по умолчанию” из входного контакта. В схеме с подтягивающим резистором это HIGH, в схеме со стягивающим резистором – LOW.

Мы выше сказали, что схема на рисунке 6.1 неверная, но это только в том случае, если мы используем режим установки

```
pinMode(2,INPUT);           // вывод D2 установлен как выход
```

Все выводы платы имеют внутри микроконтроллера резисторы, подключенные к 5 В. Их можно программно включать или отключать от выводов. Сопротивление этих резисторов порядка 20-50 кОм. Программное включение осуществляется так

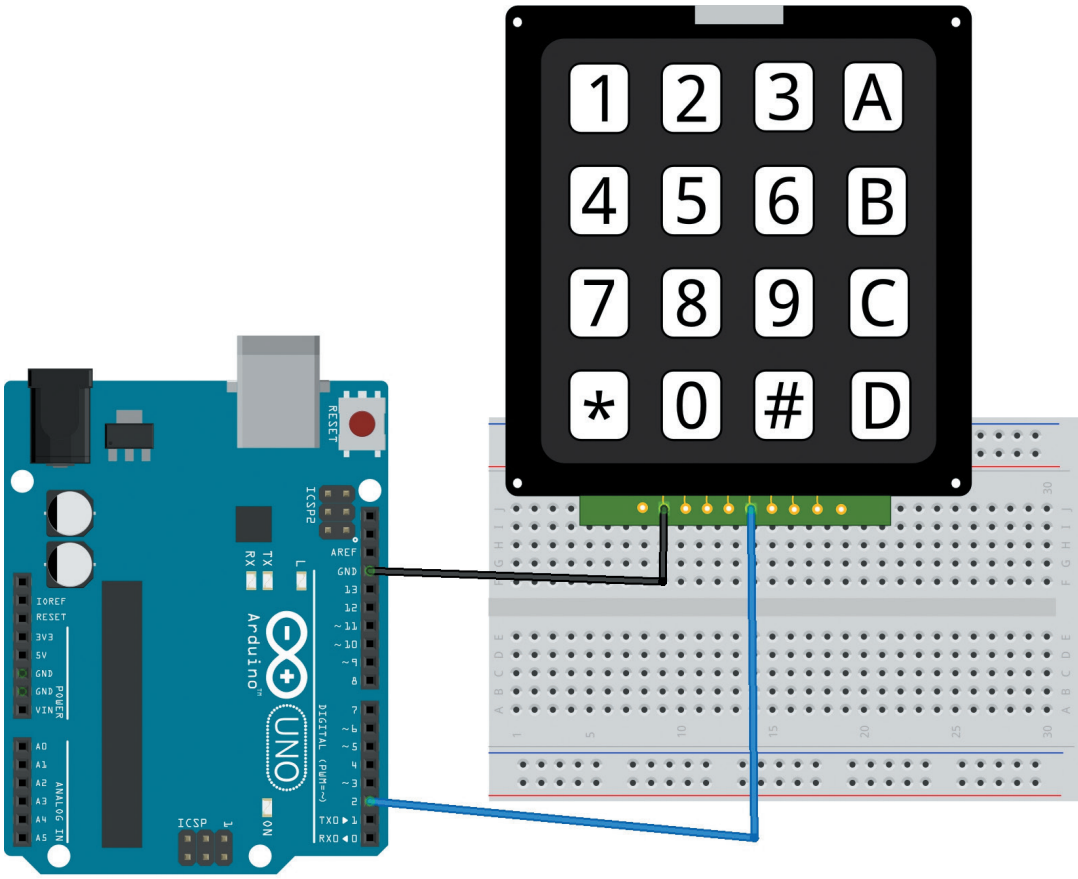


Рис. 6.2. Подсоединение кнопки к Arduino – схема со стягивающим резистором

pinMode(3, INPUT_PULLUP); // внутренний подтягивающий резистор 20кОм

И схема на рисунке 6.1 для подключения кнопки уже оказывается верной!

Теперь напишем программу, устанавливающую состояние светодиода, подключенного к выводу 13 (находится на плате Arduino) в зависимости от состояния кнопки (кнопка нажата – светодиод “горит”, кнопка отжата – светодиод “потушен”). В скетче (см. листинг 6.1) в setup() мы устанавливаем режимы выводов, а в loop() считываем состояние кнопки в переменную buttonState и передаем его на светодиод. Для схемы с подтягивающим резистором состояние переменной buttonState инвертируем, т.к. при нажатой кнопке низкое состояние сигнала, а светодиод светится при высоком. Для схемы со стягивающим резистором состояние переменной buttonState инвертировать не надо.

Листинг 6.1

```
// Контакт 13 для подключения светодиода
int LED=13;
// Контакт 2 для подключения кнопки
int BUTTON=2;
// переменная статуса кнопки buttonState
boolean buttonState;

void setup() {
  // определяем вывод LED (светодиод) как выход
  pinMode(LED, OUTPUT);
  // определяем вывод BUTTON (кнопка) как вход
  pinMode(BUTTON, INPUT_PULLUP);
}

void loop() {
  // считываем состояние BUTTON входа  и записываем в buttonState
  buttonState = digitalRead(BUTTON);
  // инверсия переменной buttonState
  // для схемы с подтягивающим резистором
  buttonState = ! buttonState;
  // записываем состояние из buttonState на выход LED (светодиод)
  digitalWrite(LED, buttonState);
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_06_01.**

Загрузим данный скетч на плату Arduino. Вы должны наблюдать включение светодиода на выводе 13 при нажатии кнопки, и выключение при отжатии кнопки.

Эксперимент 7.
Боремся с дребезгом контактов кнопки

В данном эксперименте мы будем использовать метод борьбы с дребезгом, который возникает при нажатии кнопки

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Блок клавиатуры 4x4 – 1;
- ❑ Резистор 10 кОм – 1;
- ❑ Провода MM – 5.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

В эксперименте 6 мы создали программу, устанавливающую состояние светодиода, подключенного к выводу 13 Arduino в зависимости от состояния кнопки (кнопка нажата – светодиод "горит", кнопка отжата – светодиод "потушен").

Насколько удобно держать кнопку постоянно нажатой для свечения светодиода? Гораздо удобнее иметь возможность нажать кнопку один раз, чтобы включить светодиод, и нажав его еще раз – выключить. Схема подключений согласно рис. 6.2. Загрузим на плату Arduino скетч переключения состояния светодиода при нажатии кнопки (листинг 7.1). Для отладки будем использовать вывод данных в последовательный порт. Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_07_01.**

Листинг 7.1

```
// Контакт 13 для подключения светодиода
int LED=13;
// Контакт 2 для подключения кнопки
int BUTTON=2;
// переменная статуса кнопки buttonState
boolean buttonState;
```

```
// переменная статуса кнопки предыдущая
boolean buttonStatePrev=LOW;
// переменная статуса светодиода
boolean ledState=LOW;

void setup() {
  // запуск последовательного порта
  Serial.begin(9600);
  // определяем вывод LED (светодиод) как выход
  pinMode(LED, OUTPUT);
  // определяем вывод BUTTON (кнопка) как вход
  pinMode(BUTTON, INPUT_PULLUP);
  // начальное состояние светодиода
  digitalWrite(LED, ledState);
}

void loop() {
  // считываем состояние BUTTON входа (кнопки)
  buttonState = digitalRead(BUTTON);
  // если нажатие с LOW на HIGH
  if(buttonState == HIGH && buttonStatePrev==LOW) {
     ledState = ! ledState;
     // записываем состояние из ledState на выход LED
     digitalWrite(LED, ledState);
     Serial.println(ledState);
  }
  buttonStatePrev = buttonState;
}
```

Нажимаем на кнопки и видим, что в последовательном порту при однократном нажатии кнопки происходит несколько изменений состояния кнопки (рис. 7.1), и соответственно, несколько переключений светодиода.

Почему же так происходит? Кнопки представляют собой механические устройства, с системой пружинного контакта. Когда Вы нажимаете на кнопку вниз, сигнал не просто меняется от низкого до высокого, он в течении нескольких миллисекунд меняет значение от одного до другого, прежде чем установится значение LOW. Рисунок 7.2. иллюстрирует отличие ожидаемого явления от реального.

Кнопка физически нажата в течении 25 мс. Вы могли бы предполагать, что можете сразу узнать о состоянии кнопки, считав значение с входа контакта, как показано на левом графике. Однако кнопка фактически возвращается вверх-вниз, пока значение не установится, как показано на правом графике. Теперь, зная, как ведет себя кнопка, Вы можете написать программное обеспечение кнопки с дребезгом, которое ищет изменение состояния кнопки, ожидает возврата, чтобы закончиться, и затем читает состояние переключателя снова. Эта логика программы может быть выражена следующим образом:

1) сохраняем предыдущее состояние кнопки и текущее состояние кнопки (при инициализации LOW);

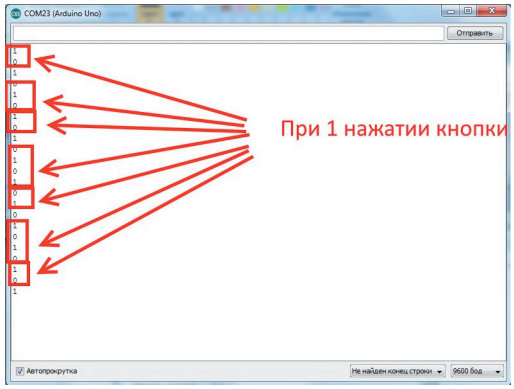


Рис. 7.1. Вывод данных отладки в монитор последовательного порта

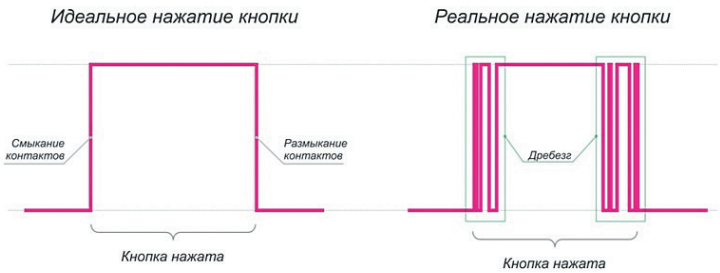


Рис. 7.2. Дребезг при нажатии кнопки

2) считываем текущее состояние кнопки;

3) если текущее состояние кнопки отличается от предыдущего состояния кнопки, ждем 5 мс, потому что кнопка, возможно, изменила состояние;

4) После 5 мс, считываем состояние кнопки и используем его в качестве текущего состояния кнопки;

5) если предыдущее состояние кнопки было LOW, а текущее состояние кнопки HIGH, переключаем состояние светодиода;

6) установливаем предыдущее состояние кнопки для текущего состояния кнопки;

7) возврат к шагу 2.

Составляем скетч по вышеприведенному алгоритму (см. листинг 1.8), загружаем на плату Arduino и проверяем. Однократное нажатие кнопки приводит к однократному изменению состояния светодиода.

Листинг 7.2.

```
// Контакт 13 для подключения светодиода
int LED=13;
// Контакт 2 для подключения кнопки
```

```
int BUTTON=2;
// Переменная для сохранения предыдущего состояния кнопки
boolean lastButton = LOW;
// Переменная для сохранения текущего состояния кнопки
boolean currentButton = LOW;
// Текущее состояние светодиода (включен/выключен)
boolean ledOn = false;

void setup() {
  // запуск последовательного порта
  Serial.begin(9600);
  // Сконфигурировать контакт светодиода как выход
  pinMode (LED, OUTPUT);
  // Сконфигурировать контакт кнопки как вход
  pinMode (BUTTON, INPUT);
}

void loop() {
  currentButton = debounce(lastButton);
  // если нажатие...
  if (lastButton == LOW && currentButton == HIGH)
    {
    // инвертировать значение состояния светодиода
    ledOn = !ledOn;
    Serial.println(ledOn);
    }
  lastButton = currentButton;
  // изменить статус состояния светодиода
  digitalWrite(LED, ledOn);
}

// Функция сглаживания дребезга
// Принимает в качестве аргумента предыдущее состояние кнопки,
// выдает фактическое.
boolean debounce(boolean last) {
  // Считать состояние кнопки
  boolean current = digitalRead(BUTTON);
  if (last != current)     // если изменилось...
    {
    // ждем 5мс
    delay(5);
    // считываем состояние кнопки
    current = digitalRead(BUTTON);
    // возвращаем состояние кнопки
    return current;
    }
}
```

Загрузим данный скетч на плату Arduino и проверяем правильность работы. Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_07_02.**

Эксперимент 8.
Подключаем несколько кнопок, управляем светодиодами

В данном эксперименте мы подключим несколько кнопок и будем управлять с их помощью несколькими светодиодами, при программировании будем использовать оператор switch … case.

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Блок клавиатуры 4x4 – 1;
- ❑ Светодиод красный – 1;
- ❑ Светодиод желтый – 1;
- ❑ Светодиод зеленый – 1;
- ❑ Резистор 10 кОм – 4;
- ❑ Резистор 220 Ом – 3;
- ❑ Провода ММ – 20.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Создадим проект использования нескольких кнопок с привязкой определенных действий по нажатию каждой кнопки.

Схема соединений для эксперимента приведена на рис. 8.1.

В данном случае используется 4 кнопки из первого ряда блока клавиатуры 4x4.

При нажатии на кнопки 1-3 включаем красный, желтый или зеленый светодиод, при нажатии на кнопку 4 выключаем все светодиоды.

Составим скетч перехода на определенный код при нажатии каждой кнопки. Создадим массивы для пинов подключения кнопок, предыдущих и текущих состояний кнопок.

```
int pinButtons[]={2,3,4,5};
int lastButtons[]={0,0,0,0};
int currentButtons[]={0,0,0,0};
```

Проверяем в цикле for наличие нажатия кнопок и вызываем функцию doButtons(), передавая в качестве аргумента индекс нажатой клавиши:

```
// проверка нажатия кнопок выбора программ
for(int i=0;i<4;i++)
 {
 // борьба с дребезгом
 currentButtons [i] = debounce(lastButtons [i],pinButtons [i]);
 // если нажатие...
 if (lastButtons [i] == 0 && currentButtons [i] == 1)
   {
   // для отладки
   Serial.println("click!");
   // функция перехода при нажатии кнопки
   doButtons(i);
   }
 lastButtons[i] = currentButtons[i];
 }
```

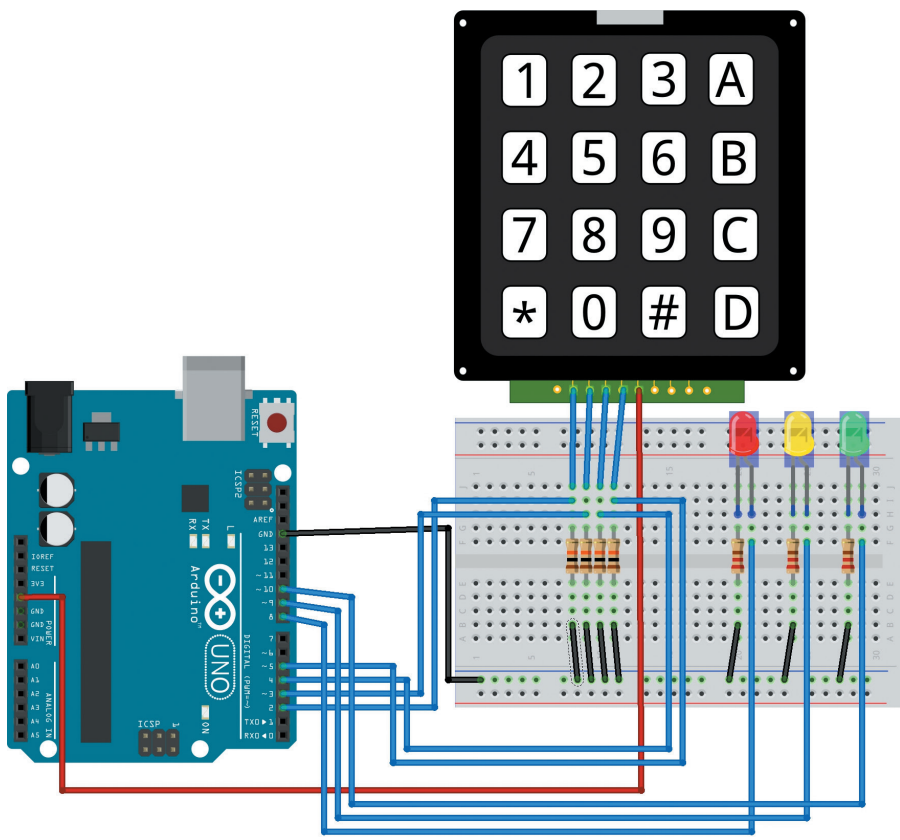


Рис. 8.1. Подсоединение 4 кнопок для управления светодиодами

В функции doButtons() мы будем делать переход в зависимости от нажатой клавиши, разных сообщений в монитор последовательного порта, используя конструкцию switch...case, корорая управляет процессом выполнения программы, позволяя задавать альтернативный код, который будет выполняться при разных условиях. Оператор switch сравнивает значение переменной со значением, определенном в операторах case. Когда найден оператор case, значение которого равно значению переменной, выполняется программный код в этом операторе.:

```
switch(x)
   {
   case 0:  // код 0
         break;
   case 1:  // код 1
         break;
   ……
   default:
         break;
   }
}
```

И весь скетч целиком показан в листинге 8.1.

Листинг 8.1.

```
// пины для подключения кнопок
int pinButtons[]={2,3,4,5};
// для сохранения предыдущих состояний кнопок
int lastButtons[]={0,0,0,0};
// для сохранения текущих состояний кнопок
int currentButtons[]={0,0,0,0};
// пины подключения светодиодов red,yellow,green
int pinLeds[]={8,9,10};

void setup() {
  // запуск последовательного порта
  Serial.begin(9600);
  // Сконфигурировать пины подключения кнопок как вход
  for(int i=0;i<4;i++) {
     pinMode (pinButtons[i], INPUT);
  }
  // Сконфигурировать пины подключения светодиодов
  // как выход и выключить
  for(int i=0;i<3;i++) {
     pinMode (pinLeds[i], OUTPUT);
     digitalWrite (pinLeds[i], LOW);
  }
}

void loop() {
```

```
  // проверка нажатия кнопок выбора программ
  for(int i=0;i<4;i++)
   {
   // борьба с дребезгом
   currentButtons [i] = debounce(lastButtons [i],pinButtons [i]);
   // если нажатие...
   if (lastButtons [i] == 0 && currentButtons [i] == 1)
     {
     // для отладки
     Serial.println("click!");
     // функция перехода при нажатии кнопки
     doButtons(i);
     }
   lastButtons[i] = currentButtons[i];
   }
}
// обработка клавиш выбора программ
void doButtons(int but)
   {
   switch(but)
      {
      case 0:  Serial.println("case for button 0»);
               digitalWrite(pinLeds[0],HIGH);
            break;
      case 1:  Serial.println("case for button 1»);
               digitalWrite(pinLeds[1],HIGH);
            break;
      case 2:  Serial.println("case for button 2»);
               digitalWrite(pinLeds[2],HIGH);
           break;
      case 3:  Serial.println("case for button 3»);
               for(int i=0;i<3;i++) {
                  digitalWrite (pinLeds[i], LOW);
               }
            break;
      default:
            break;
     }
  }
// Функция сглаживания дребезга
int debounce(int last,int pin1) {
  int current = digitalRead(pin1);
  if (last != current)
    {
    delay(5);
    current = digitalRead(pin1);
    return current;
    }
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_08_01.**

Загружаем скетч на плату Arduino и проверяем работу кнопок – в мониторе последовательного порта (рис. 8.2).

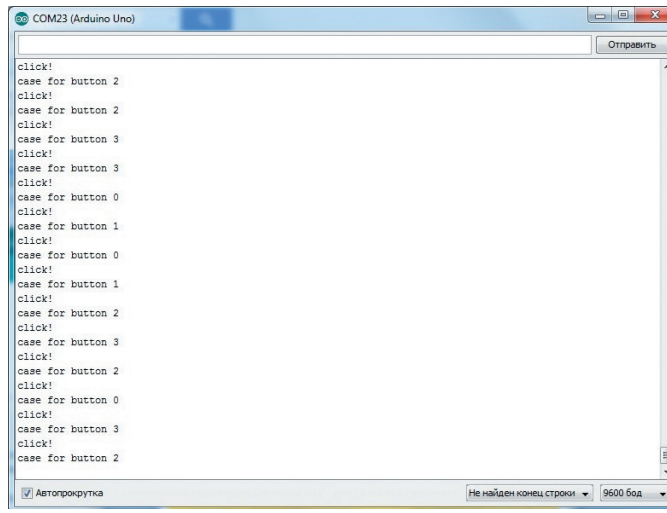


Рис. 8.2. Вывод данных работы скетч в монитор последовательного порта

Теперь мы можем создавать проекты, в которых будем делать выбор кода в зависимости от нажатия определенной кнопки. В следующем эксперименте мы изменим наш проект “бегущего огня”, добавив управление скоростью и направлением движения с помощью четырех кнопок.

Эксперимент 9.
delay() и millis() - управляем скоростью и направлением «бегущего огня» с помощью кнопок

В данном эксперименте мы будем управлять скоросью и напралением "бегущего огня" на светодиодах с помощью кнопок, научимся писать программы без использования функции delay().

В эксперименте мы будем использовать следующие компоненты:
- Плата Arduino UNO – 1;
- Кабель USB;
- Плата прототипирования – 1;
- Блок клавиатуры 4x4 – 1;
- Светодиод – 8;
- Резистор 10 кОм – 4;
- Резистор 220 Ом – 8;
- Провода ММ – 20.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Создадим проект установки скорости и направления движения "бегущего огня" на светодиодах посредством кнопок. Будем использовать 4 кнопки:
- направление движения слева-направо;
- направление движения справа-налево;
- увеличить скорость движения;
- уменьшить скорость движения.

Схема соединений показана на рис. 9.1. В данном случае используется 4 кнопки из первого ряда блока клавиатуры 4x4.

Казалось бы, с учетом тех знаний, что мы приобрели за предыдущие уроки, код для бегущего огня совсем простой, примерно так:

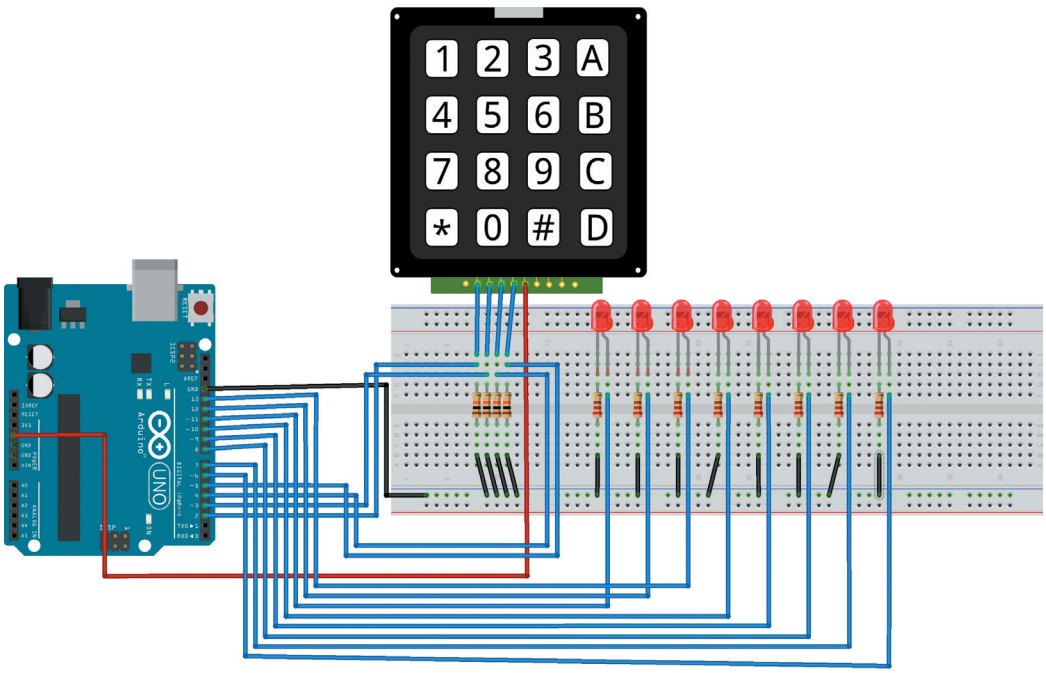


Рис. 9.1. Схема для управления светодиодами с помощью 4 кнопок

```
void loop() {
   // определение нажатия, если оно есть
   код
   // переключение светодиодов
   for (int i=0;i<8; i=i+1) {
      if(i==tekled)
         {digitalWrite(pinLeds[i],HIGH);}
      else
         {digitalWrite(pinLeds[i],LOW);}
   }
   delay(interval);
   tekled=(tekled+1)%8;
}
```

Но во время выполнения команды delay(interval) программа останавливается, и если в это время произойдет нажатие кнопки, программа его пропустит. Но решение есть – не использовать функцию delay()! Есть стандартная Arduino-функция millis(), которая возвращает количество миллисекунд, прошедшее с начала программы. И код:

```
void loop() {
   // определение нажатия, если оно есть
   код
```

```
    // переключение светодиодов
    if(millis()-t>=interval) {
      for (int i=0;i<8; i=i+1) {
        if(i==tekled)
          {digitalWrite(pinLeds[i],HIGH);}
        else
          {digitalWrite(pinLeds[i],LOW);}
      }
      tekled=(tekled+1)%8;
      t=millis();
    }
}
```

будет выполнять переключение следующего состояния светодиодв только один раз за интервал времени = interval, остальное время будет идти опрос кнопок и проверка условия if(millis()-t>=interval).

Нам только остается добавить обработку нажатий кнопок.

Для установки направления движения “бегущего огня” введем переменную dir (1 или -1) и с помощью нее будем устанавливать значение текущего “горящего” светодиода.

```
tekled=(tekled+dir)%8;
if(tekled<0) {
   tekled=7;
}
```

И весь скетч целиком показан в листинге 9.1.

Листинг 9.1.

```
// пины для подключения кнопок
int pinButtons[]={2,3,4,5};
// для сохранения предыдущих состояний кнопок
int lastButtons[]={0,0,0,0};
// для сохранения текущих состояний кнопок
int currentButtons[]={0,0,0,0};
// пины подключения светодиодов red,yellow,green
int pinLeds[]={6,7,8,9,10,11,12};
// скорость бегущего огня
unsigned long interval=1000;
unsigned long t=0;
// текущий горящий светодиод
int tekled=0;
// направлени бегущего огня
int dir=1;

void setup() {
  // Сконфигурировать пины подключения кнопок как вход
  for(int i=0;i<4;i++) {
```

```
      pinMode (pinButtons[i], INPUT);
  }
  // Сконфигурировать пины подключения светодиодов
  // как выход
  for(int i=0;i<8;i++) {
     pinMode (pinLeds[i], OUTPUT);
  }

}

void loop() {
  // проверка нажатия кнопок выбора программ
  for(int i=0;i<4;i++)
   {
   // борьба с дребезгом
   currentButtons [i] = debounce(lastButtons [i],pinButtons [i]);
   // если нажатие...
   if (lastButtons [i] == 0 && currentButtons [i] == 1)
     {
     // функция перехода при нажатии кнопки
     doButtons(i);
     }
   lastButtons[i] = currentButtons[i];
   }
   // переключения светодиодов
   if(millis()-t>=interval) {
     for (int i=0;i<8; i=i+1) {
       if(i==tekled)
         {digitalWrite(pinLeds[i],HIGH);}
       else
         {digitalWrite(pinLeds[i],LOW);}
     }
     tekled=(tekled+dir)%8;
     if(tekled<0) {
       tekled=7;
     }
     t=millis();
   }

}
// обработка клавиш выбора программ
void doButtons(int but)
   {
   switch(but)
      {
      case 0:  dir=1;
            break;
      case 1:  dir=-1;
```

```
            break;
      case 2:  Serial.println("case for button 2»);
               digitalWrite(pinLeds[2],HIGH);
           break;
      case 3:  Serial.println("case for button 3»);
               for(int i=0;i<3;i++) {
                  digitalWrite (pinLeds[i], LOW);
               }
            break;
      default:
            break;
     }
  }

// Функция сглаживания дребезга
int debounce(int last,int pin1) {
  int current = digitalRead(pin1);
  if (last != current)
    {
    delay(5);
    current = digitalRead(pin1);
    return current;
    }
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_09_01.**

Загружаем скетч на плату Arduino и проверяем работу.

Эксперимент 10.
Подключение 7-сегментного одноразрядного индикатора

В этом эксперименте мы рассмотрим работу с семисегментным светодиодным индикатором, который позволяет Arduino визуализировать цифры.

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Индикатор 7-сегментный одноразрядный – 1;
- ❑ Резистор 220 Ом – 8;
- ❑ Провода MM – 15.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Светодиодный семисегментный индикатор представляет собой группу светодиодов, расположенных в определенном порядке и объединенных конструктивно. Светодиодные контакты промаркированы метками от a до g (и дополнительно dp – для отображения десятичной точки), и один общий вывод, который определяет тип подключения индикатора (схема с общим анодом ОА, или общим катодом ОК). В наборе в наличие светодиодный семисегментный индикатор 5161AS – это индикатор с общим катодом ОК. Зажигая одновременно несколько светодиодов, можно формировать на индикаторе символы цифр.

Назначение контактов индикатора показано на рисунке 10.1.

Создадим проект вывода на семисегментный индикатор цифр от 0 до 9.

Схема подключения индикатора к плате Arduino показана на рис. 10.2. Обратите внимание – наличие ограничительных резисторов обязательно, в противном случае можно спалить светодиоды индикатора.

Теперь приступим к написанию скетча. Необходимо каждую секунду выводить на индикатор цифру от 0 до 9 и далее по кругу.

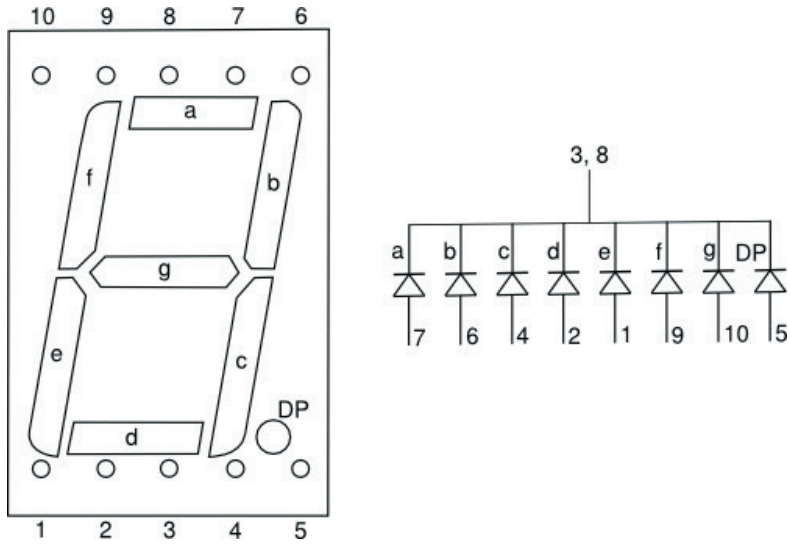


Рис. 10.1. Назначение контактов индикатора 5161AS

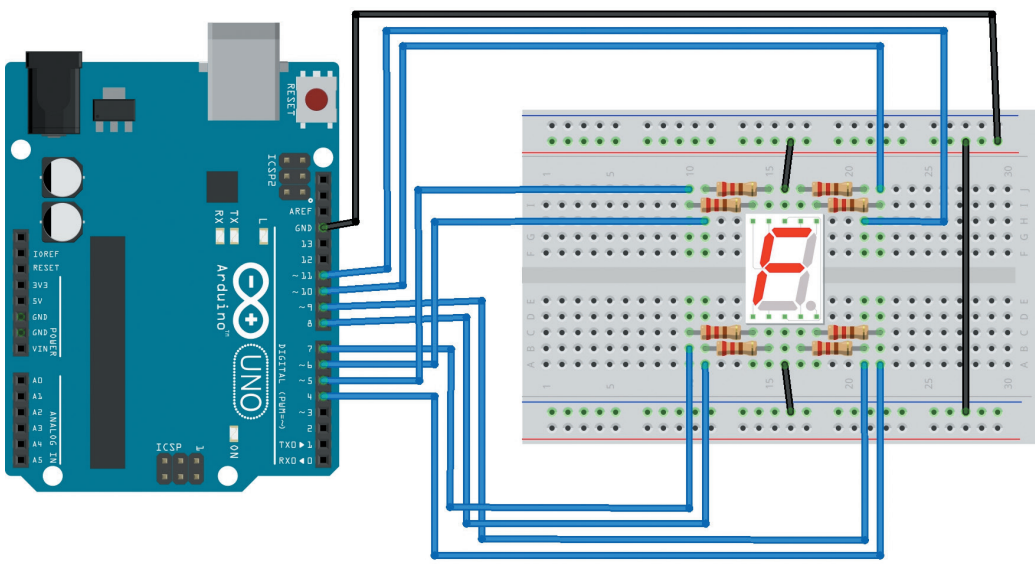


Рис. 10.2. Схема соединений для подключения индикатора к плате Arduino

Сначала определимся, в каком виде будем хранить данные для каждой цифры. Нам требуется определять каждую цифру с помощью 7 значений 0 или 1 (разряд десятичной точки не будем использовать). Самое экономное – отводить для хранения

ЦИФРА	СЕГМЕНТЫ ИНДИКАТОРА								ЧИСЛО В ДВОИЧНОЙ ФОРМЕ
	a – бит 7	b – бит 6	c – бит 5	d – бит 4	e – бит 3	f – бит 2	g – бит 1	dp – бит 0	
0	1	1	1	1	1	1	0	0	B11111100
1	0	1	1	0	0	0	0	0	B01100000
2	1	1	0	1	1	0	1	0	B11011010
3	1	1	1	1	0	0	1	0	B11110010
4	0	1	1	0	0	1	1	0	B01100110
5	1	0	1	1	0	1	1	0	B10110110
6	1	0	1	1	1	1	1	0	B10111110
7	1	1	1	0	0	0	0	0	B11100000
8	1	1	1	1	1	1	1	0	B11111110
9	1	1	1	1	0	1	1	0	B11110110

информации о каждой цифре 1 байт, который состоит из 8 бит (каждый бит – 0 или 1). Данные для каждой цифры представлены в таблице 10.1.

Т.е. для описания 10 цифр (от 0 до 9) нам потребуется массив

```
// значения для вывода цифр 0-9
byte numbers[10] = { B11111100,   // 0
                     B01100000,   // 1
                     B11011010,   // 2
                     B11110010,   // 3
                     B01100110,   // 4
                     B10110110,   // 5
                     B10111110,   // 6
                     B11100000,   // 7
                     B11111110,   // 8
                     B11100110    // 9
};
```

Теперь напишем подпрограмму setNumber(), отвечающую за вывод значений на выводы Arduino для индикации цифры. Будем использовать Arduino-функцию для bitRead(), которая возвращает состояние указанного бита числа (1 или 0), Нумерация начинается с младшего значащего бита (крайнего правого) с номером 0:

```
x=bitRead(B01001000,6);   // x=1 - шестой бит, начиная с нулевого
(крайнего справа)
```

И сама подпрограмма:

```
void setNumber(int num) {
   // пройти по всем битам
   for(int i=0;i<7;i++) {
      if(bitRead(numbers[num],7-i)==HIGH) // зажечь сегмент
         digitalWrite(pins[i],HIGH);
      else // потушить сегмент
         digitalWrite(pins[i],LOW);
   }
}
```

Каждую секунду (реализуем с помощью delay(1000)) мы будем выводить на индикатор новое число от 0 до 9. Полное содержимое скетча показано в листинге 10.1.

Листинг 10.1.

```
// список выводов Arduino для подключения к разрядам a-g
// семисегментного индикатора
int pins[7]={11,10,9,8,7,6,5,4};
// значения для вывода цифр 0-9
byte numbers[10] = { B11111100,   // 0
                     B01100000,   // 1
                     B11011010,   // 2
                     B11110010,   // 3
                     B01100110,   // 4
                     B10110110,   // 5
                     B10111110,   // 6
                     B11100000,   // 7
                     B11111110,   // 8
                     B11100110    // 9
};
// переменная для хранения значения текущей цифры
int number=0;

void setup() {
   // Сконфигурировать контакты как выходы
   for(int i=0;i<7;i++)
      pinMode(pins[i],OUTPUT);
}

void loop()  {
   setNumber(number);
   delay(1000);
   // следующая цифра
   number=(number+1)%10;
}

// функция вывода цифры на семисегментный индикатор
void setNumber(int num) {
   for(int i=0;i<7;i++) {
      if(bitRead(numbers[num],7-i)==HIGH) // зажечь сегмент
         digitalWrite(pins[i],HIGH);
      else  // потушить сегмент
         digitalWrite(pins[i],LOW);
   }
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_10_01.**

Загружаем скетч на плату Arduino, и смотрим на изменение цифр, выводимых на светодиодный индикатор.

Эксперимент 11.
Матрица 4-разрядная из 7-сегментных индикаторов

В этом эксперименте мы рассмотрим работу матрицы 4-разрядной из 7-сегментных индикаторов.

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino + WiFi – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ 4-разрядная матрица на семисегментных светодиодных индикаторах – 1;
- ❑ Резистор 220 Ом – 8;
- ❑ Провода MM – 15.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Светодиодный семисегментный индикатор, с которым мы работали в Эксперимете 10, используется редко, гораздо чаще используются сборки из 4 или 8 семисегментных индикаторов, на которые можно выводить какие-то осмысленные данные, например текущее время. Рассмотрим сборку из 4 семисегментных индикаторов, предназначенную для вывода 4 цифр с возможностью вывода десятичной точки. Схема 4-разрядной матрицы на 7-сегментных индикаторах показана на рис. 1.49.

Для вывода цифры необходимо зажечь нужные светодиоды на контактах A–G и DP и выбрать нужную матрицу подачей LOW на вывод 6, 8, 9 или 12.

Напишем скетч последовательного вывода цифр (0–9) на произвольный регистр матрицы. Схема подключения индикатора к плате Arduino показана на рис. 11.2.

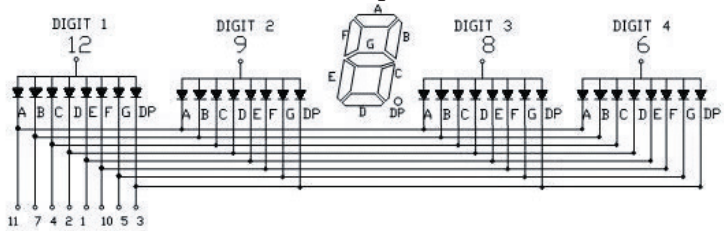


Рис. 11.1. Схема 4-разрядной матрицы на 7-сегментных индикаторах

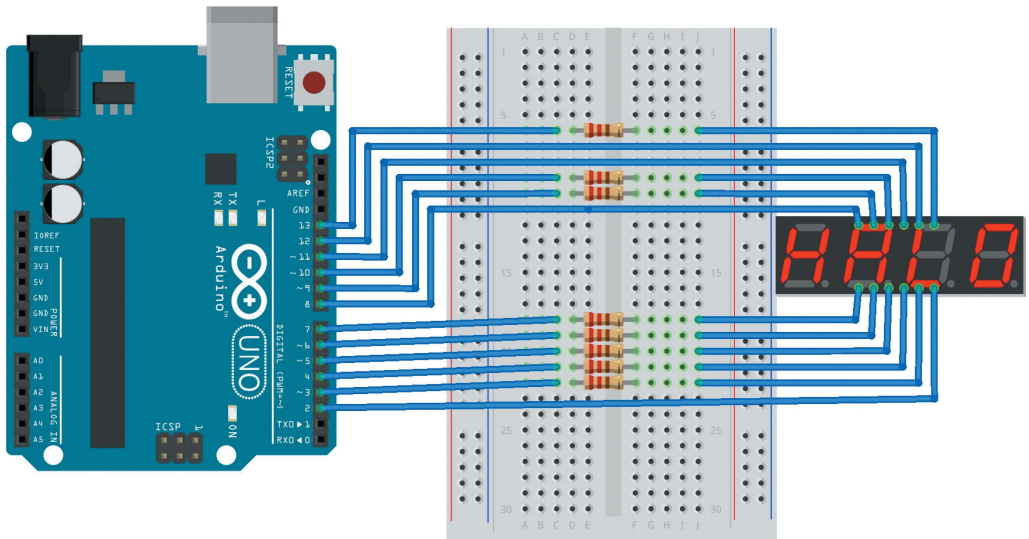


Рис. 11.2. Схема подключения 4-разрядной матрицы на 7-сегментных индикаторах

Обратите внимание – наличие ограничительных резисторов обязательно, в противном случае можно спалить светодиоды индикатора.

Приступим к написанию скетча. Для выбора случайного значения из диапазона будем использовать функцию random(). Функция random() выдает целое случайное значение из диапазона.

- ❑ random(max);
- ❑ random(min, max):
  - min: нижняя граница случайных значений, включительно. (опционально);
  - max: верхняя граница случайных значений, не включительно.

В массиве numbers[] хранятся значения, соответствующие данным для отображения цифр 0–9 (старший разряд байта соответствует метке сегмента A индикатора, а младший – сегменту G), в массиве pins[] – значения контактов для сегментов A–G и DP, в массиве pindigits[] – значения контактов для выбора разряда матрицы.

Каждую секунду (реализуем с помощью delay(1000)) мы будем выводить на индикатор случайное число из диапазона 0 – 9:

```
number=random(0,10);
```

Выбор матрицы, также выбираем случайно (0,1,2,3):

```
digit=random(0,4);
```

Содержимое скетча показано в листинге 11.1.

Листинг 11.1.

```
// список выводов Arduino для подключения к разрядам a-g
// семисегментного индикатора
int pins[8]={9,13,4,6,7,10,3,5};
```

```
// значения для вывода цифр 0-9
byte numbers[10] = { B11111100,   // 0
                     B01100000,   // 1
                     B11011010,   // 2
                     B11110010,   // 3
                     B01100110,   // 4
                     B10110110,   // 5
                     B10111110,   // 6
                     B11100000,   // 7
                     B11111110,   // 8
                     B11100110    // 9
};
// переменная для хранения значения текущей цифры
// семисегментного индикатора
int number=0;
// список выводов Arduino для выбора матрицы 0-3
int pindigits[4]={2,8,11,12};
// переменная для хранения текущего разряда
int digit=0;

void setup() {
   // Сконфигурировать контакты как выходы
   for(int i=0;i<8;i++)
      pinMode(pins[i],OUTPUT);
   for(int i=0;i<4;i++) {
      pinMode(pindigits[i],OUTPUT);
      digitalWrite(pindigits[i],HIGH);
   }
}

void loop() {
   number=random(0,10);
   // установить выводы a-g
   setNumber(number);
   // отключить все матрицы
   for(int i=0;i<4;i++)
      digitalWrite(pindigits[i],HIGH);
   // получить номер матрицы
   digit=random(0,4);
   // включить выбранную матрицу
   digitalWrite(pindigits[digit],LOW);
   // пауза 1 сек
   delay(1000);
}
// функция вывода цифры на семисегментный индикатор
void setNumber(int num) {
   for(int i=0;i<7;i++) {
      if(bitRead(numbers[num],7-i)==HIGH) // зажечь сегмент
         digitalWrite(pins[i],HIGH);
      else // потушить сегмент
         digitalWrite(pins[i],LOW);
   }
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_11_01.

Загружаем скетч на плату Arduino, и смотрим на вывод случайных цифр на случайную матрицу, меняющееся каждую секунду.

Эксперимент 12.
Секундомер на 4-разрядной матрице из 7-сегментных индикаторов

В этом эксперименте мы рассмотрим и создадим секундомер на 4-разрядной матрице из 7-сегментных индикаторов.

В эксперименте мы будем использовать следующие компоненты:

- Плата Arduino + WiFi – 1;
- Кабель USB;
- Плата прототипирования – 1;
- 4-разрядный матрица на семисегментных светодиодных индикаторах – 1;
- Резистор 220 Ом – 8;
- Блок клавиатуры 4x4 – 1;
- Резистор 10 кОм – 1.
- Провода ММ – 15.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

В эксперименте 11 мы рассмотрели вывод случайных цифр на случайную матрицу, меняющуюся каждую секунду. А как же нам выводить данные одновременно на все 4 индикатора, причем на каждый индикатор различные? Для этого будем использовать динамическую индикацию. При динамической индикации сегменты зажигаются по очереди. Наше зрение обладает свойством инерции, или персистенции. Это способность глаза соединять быстро сменяющиеся изображения в одно. Таким образом, чтобы человек видел на индикаторе четырехзначное число, вовсе необязательно зажигать все цифры разом. Достаточно в один момент времени включать только один отдельный индикатор. Переключение между соседними индикаторами должно происходить с большой частотой, чтобы получить эффект персистенции. Многие символьные и матричные светодиодные и газоразрядные индикаторы работают именно по такому принципу.

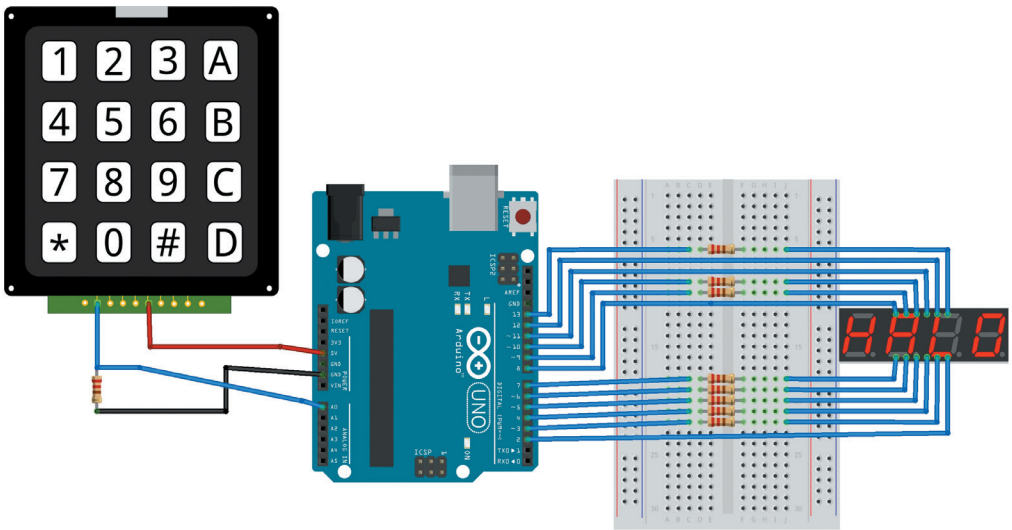


Рис. 12.1. Схема соединений для секундомера на 4-разрядной матрице на семисегментных светодиодных индикаторах

Создадим секундомер с точностью до 0.1 секунды, запускаемый и отключаемый по кнопке.

Схема подключений показана на рис. 12.1.

Приступаем к написанию скетча. Кнопку подсоединяем к выводу 14(A0), который будем использовать, как цифровой вход. При нажатии кнопки поочередно устанавливается режим работы скетча:

```
// изменение режима (1- секундомер)
mode=1-mode;
```

Для режима работы секундомера каждые 100 мсек происходит инкремент счетчика number (общее количество десятых долей секунд). На первую матрицу выводим количество сотен секунд, на вторую – десятков секунд, на третью – секунд, а на четвертую – десятых долей секунд. Значение цифры для текущего разряда получаем следующим образом

```
number1=number;
for(int i=0;i<4;i++) {
   number2=number1%10;
   number1=number1/10;
   setNumber(number2,i);
   for(int j=0;j<4;j++)
      {digitalWrite(pindigits[j],HIGH);}
   digitalWrite(pindigits[i],LOW);
   delay(1);
}
```

Процедуру вывода цифры на индикатор setNumber() немного изменяем, вводя для третьей матрицы вывод десятичной точки:

```
if(dig==1) // десятичная точка для второго разряда (третья матрица)
   digitalWrite(pins[7],HIGH);
```

Содержимое скетча показано в листинге 12.1.

Листинг 12.1.

```
// список выводов Arduino для подключения к разрядам a-g
// семисегментного индикатора
int pins[8]={9,13,4,6,7,10,3,5};
// значения для вывода цифр 0-9
byte numbers[10] = { B11111100, B01100000, B11011010,
B11110010, B01100110, B10110110,
B10111110, B11100000, B11111110,
B11110110};
// переменная для хранения и обработки текущего значения
// семисегментного индикатора
int number=0;
int number1=0;
int number2=0;
// список выводов Arduino для выбора матрицы 0-3
int pindigits[4]={2,8,11,12};
// переменная для хранения текущего разряда
int digit=0;
// для отмеривания 100 мс
unsigned long millis1=0;
// режим 1 - секундомер работает
int mode=0;
// Контакт 14(A0) для подключения кнопки
const int BUTTON=14;
// Переменная для сохранения текущего состояния кнопки
int tekButton = LOW;
// Переменная для сохранения предыдущего состояния кнопки
int prevButton = LOW;

void setup() {
   // Сконфигурировать контакт кнопки как вход
   pinMode (BUTTON, INPUT);
   // Сконфигурировать контакты как выходы
   for(int i=0;i<8;i++)
      pinMode(pins[i],OUTPUT);
   // выключить все контакты выбора матриц
   for(int i=0;i<4;i++) {
      pinMode(pindigits[i],OUTPUT);
      digitalWrite(pindigits[i],HIGH);
   }
}

void loop() {
   tekButton = debounce(prevButton);
```

```
    // если нажатие...
    if (prevButton == LOW && tekButton == HIGH) {
       // изменение режима
       mode=1-mode;
       if(mode==1)
          number=0;
    }
    if(millis()-millis1>=100 && mode==1) {
       millis1=millis1+100;
       number=number+1;
       if(number==10000)
          number=0;
    }
    number1=number;
    for(int i=0;i<4;i++) {
       number2=number1%10;
       number1=number1/10;
       setNumber(number2,i);
       for(int j=0;j<4;j++)
          digitalWrite(pindigits[j],HIGH);
       digitalWrite(pindigits[i],LOW);
       delay(1);
    }
}
// функция вывода цифры на семисегментный индикатор
void setNumber(int num,int dig) {
    for(int i=0;i<8;i++) {
       if(bitRead(numbers[num],7-i)==HIGH) // зажечь сегмент
          digitalWrite(pins[i],HIGH);
       else // потушить сегмент
          digitalWrite(pins[i],LOW);
    }
    if(dig==1) // десятичная точка для второго разряда
       digitalWrite(pins[7],HIGH);
}
// Функция сглаживания дребезга.
boolean debounce(boolean last) {
    boolean current = digitalRead(BUTTON);
    if (last != current) {
       delay(5); // ждем 5 мс
       current = digitalRead(BUTTON);
       // возвращаем состояние кнопки
       return current;
    }
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_12_01**. Загружаем скетч на плату Arduino, и проверяем работу секундомера.

Эксперимент 13.
Аналоговые входы Arduino. Подключение потенциометра

В этом эксперименте познакомимся с аналоговыми входами Arduino, подключим к Arduino простейший аналоговый датчик - потенциометр, и научимся измерять его сопротивление.

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino + WiFi – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Потенциометр 2.2 кОм– 1;
- ❑ Потенциометр 10 кОм – 1;
- ❑ Провода ММ – 8.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

В предыдущих экспериментах мы работали с цифровыми выводами Arduino, которые могут работать в режиме входов для считывания внешних цифровых сигналов и выходов для выдачи цифровых данных. Цифровые данные имеют только два состояние HIGH и LOW. Но мир вокруг вас является аналоговым, и большинство наблюдаемых особенностей в окружающем мире всегда будет иметь аналоговый характер. Мир предполагает бесконечное число возможных состояний.

На рисунке 13.1. показывано, как аналоговые и цифровые сигналы отличаются друг от друга. Слева прямоугольная волна, которая варьируется только между двумя значениями: 0 и 5 вольт. Справа часть косинусоидальной волны. Несмотря на то, что его границы все еще 0 и 5 вольт, сигнал берет бесконечное число значений между теми двумя напряжениями.

Компьютерная система никогда не может осуществлять измерение с бесконечным числом десятичных разрядов для аналогового значения, потому что память и производительность компьютера является конечным значением. Если это так,

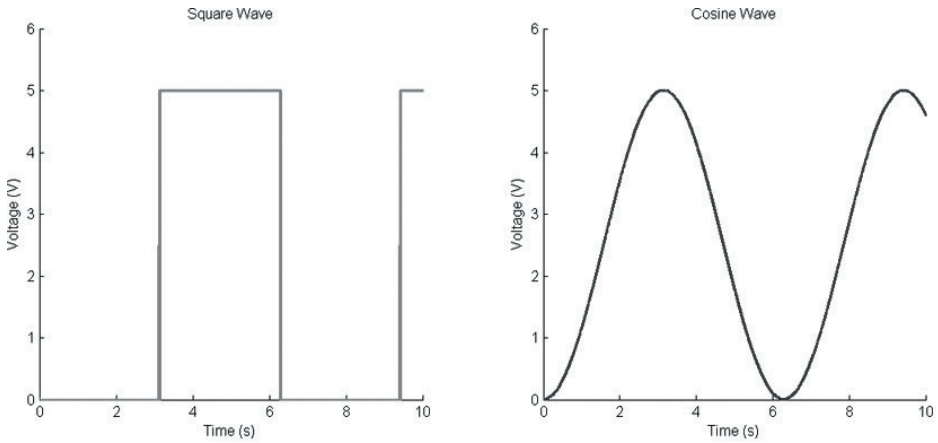


Рис. 13.1. График значений для аналоговых и цифровых сигналов

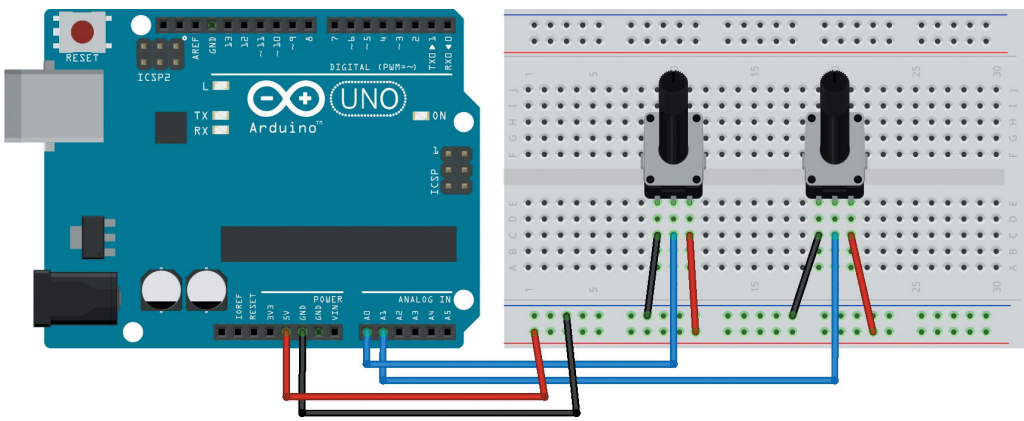


Рис. 13.2. Подключение потенциометров к плате Arduino

как соединить интерфейс цифрового Arduino с аналоговым реальным миром? С помощью аналого-цифрового преобразователя (АЦП), который может преобразовать аналоговые значения в цифровые представления с конечной точностью.

Каждый из аналоговых контактов Arduino (для Arduino Uno это A0 – A5) содержит 10-разрядный АЦП для аналоговых преобразований. 10-разрядный означает, что АЦП может разделить аналоговый сигнал на 210 различных значений (210=1024 – от 0 до 1023). Опорное напряжение определяет максимальное напряжение, его значение соответствует значению 1023 АЦП.

Самый простой аналоговый датчик, с которого можно получить данные – это потенциометр. Потенциометры являются переменными делителями напряжения, и выглядят часто как ручки. Они бывают разных размеров и форм, но все имеют три вывода.

В наборе есть два потенциометра (2.2 кОм и 10 кОм). Подключим их к плате Arduino согласно схеме соединений на рис. 13.2.

Для считывания данных с аналогового порта в Arduino есть функция analogRead(), которая возвращает аналоговое значение с аналогового входа Arduino, например:

```
int val=analogRead(A0);
```

Загрузим на плату Arduino скетч из листинга 13.1 – вывод в монитор последовательного порта значений с потенциометра, подключенного к аналоговому входу A0.

Листинг 13.1.

```
// пин подключения среднего вывода потенциометра
#define PIN_ANALOG A0
// переменная для сохранения значения потенциометра
int val;

void setup() {
   // запуск последовательного порта
   Serial.begin(9600);
}
void loop() {
  // считать значение с аналогового порта
  val=analogRead(PIN_ANALOG);
  // вывести последовательный порт
  Serial.print("val=");
  Serial.println(val);
  delay(500);
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_13_01. Загрузим скетч на плату, откроем монитор последовательного порта и покрутим ручку потенциометра, значения будут меняться в промежутке от 0 до 1023 (см. рис. 13.3).

Проблем нет, если мы хотим получить реальные значения потенциометра в Ом, просто составляем пропорцию:

```
1023 – 10кОм
val – x кОм
x=val*10/1023
```

Скетч для измерения текущих значений для двух потенциометров (2.2 кОм и 10 кОм) и вывода их значений в Ом в последовательный порт показан в листинге 13.2.

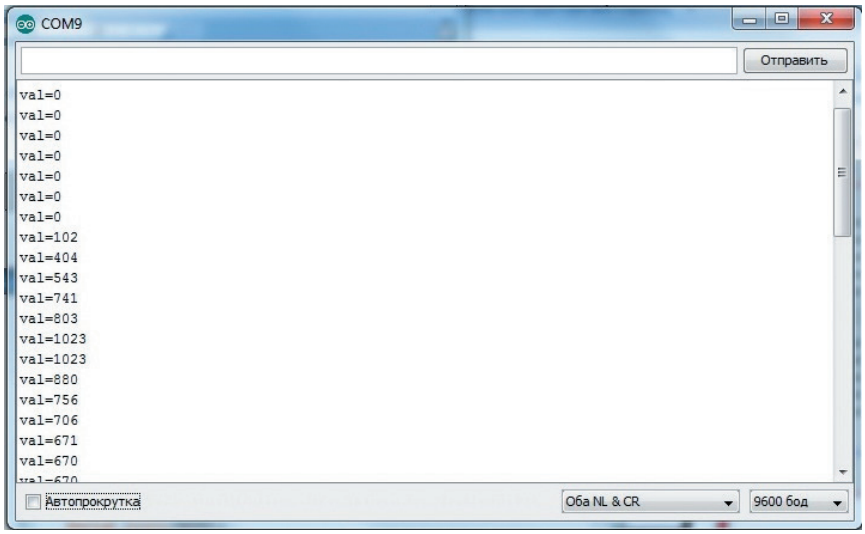


Рис. 13.3. Вывод данных с аналогового входа в монитор последовательного порта

Листинг 13.2.

```
// пины подключения средних выводов потенциометров
int pinPot1=A0;
int pinPot2=A1;
// переменная для сохранения значения потенциометра
int val1, val2;
float val1Om, val2Om;

void setup() {
   // запуск последовательного порта
   Serial.begin(9600);
}
void loop() {
  // считать значение с аналоговых портов
  val1=analogRead(pinPot1);
  val2=analogRead(pinPot2);
  // переводим в Ом
  val1Om=2200.00*val1/1023;
  val2Om=10000.00*val2/1023;
  // вывести последовательный порт
  Serial.print("R1=");
  Serial.print(val1Om);
  Serial.print(" Om");
  Serial.print("   R2=");
  Serial.print(val2Om);
  Serial.println(" Om");
  delay(500);
}
```

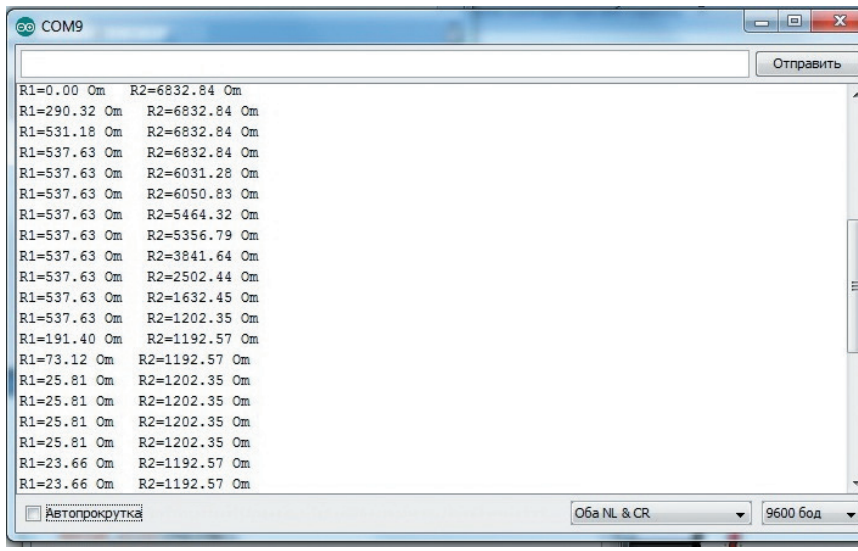


Рис. 13.4. Вывод показаний потенциометров в монитор последовательного порта

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_13_02**. Загрузим скетч на плату, и смотрим в мониторе последовательного порта значения сопротивления в Ом для двух потенциометров (см. рис. 13.4).

Эксперимент 14.
Использование потенциометра потенциометра в качестве регулятора показаний светодиодной шкалы

В этом эксперименте научимся масштабировать аналоговые данные с помощью функции map и выводить их на 10-разрядный светодиодный индикатор.

В эксперименте мы будем использовать следующие компоненты:
- Плата Arduino + WiFi – 1;
- Кабель USB;
- Плата прототипирования – 1;
- Потенциометр– 1;
- 10-разрядная линейная светодиодная шкала – 1;
- Резистор 220 Ом – 10;
- Провода ММ – 20.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Будем использовать потенциометр в качестве регулятора показаний 10-разрядной линейной светодиодной шкалы. Схема соединений показана на рис. 14.1.

Приступим к написанию скетча. Наша 10-разрядная шкала регулируется потенциометром. Показание на аналоговом входе соответствует нулевому показанию шкалы, 1023 – полному показанию шкалы. Промежуточные значения можно вычислять, как в Эксперименте 13, используя математическую пропорцию. А можно, используя встроенную функцию map():

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

Функция пропорционально переносит значение value из текущего диапазона значений fromLow .. fromHigh в новый диапазон toLow .. toHigh.

Обратите внимание, что “нижняя граница” может быть как меньше, так и больше “верхней границы”. Это может быть использовано для того чтобы “перевернуть” диапазон:

```
y = map(x, 1, 50, 50, 1);
```

Возможно использование отрицательных значений:

```
y = map(x, 1, 50, 50, -100);
```

Функция map() оперирует целыми числами. При пропорциональном переносе дробная часть не округляется по правилами, а просто отбрасывается.

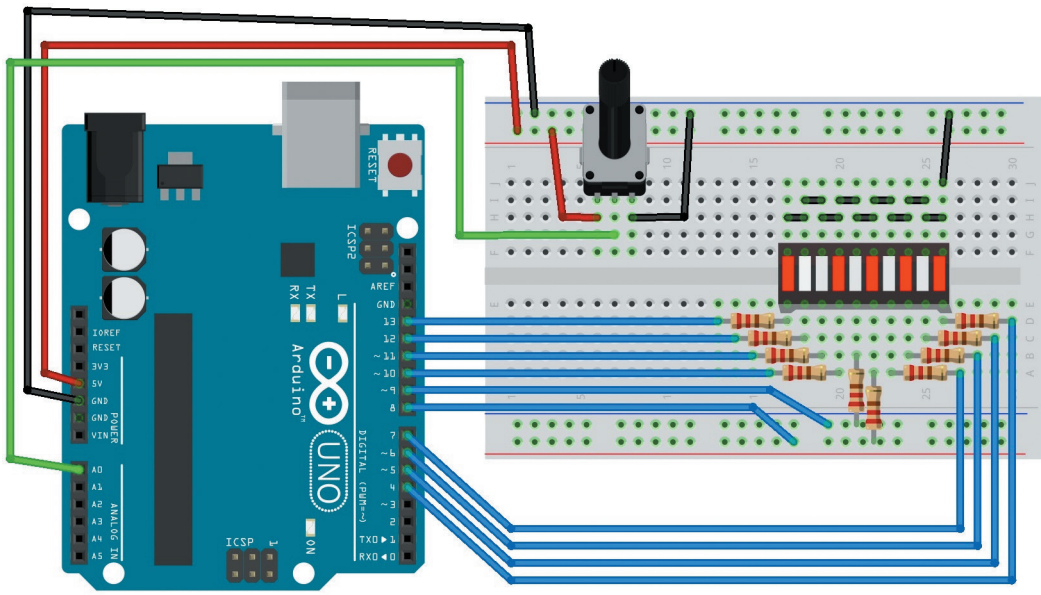


Рис. 14.1. Схема соединений для регулятора светодиодной шкалы

Приступаем к написанию скетча. Аналоговые данные потенциометра (0–1023) масштабируем в данные шкалы (0–10) с помощью функции map() и зажигаем соответствующее количество светодиодов. Содержимое скетч показано в листинге 14.1.

Листинг 14.1.

```
// Аналоговый вход A0 для подключения потенциометра
const int POT=0;
// переменная для хранения значения потенциометра
int valpot = 0;
// список контактов подключения светодиодной шкалы
const int pinsled[10]={4,5,6,7,8,9,10,11,12,13};
// переменная для хранения значения шкалы
int countleds = 0;

void setup() {
   // запуск последовательного порта
   Serial.begin(9600);
   for(int i=0;i<10;i++) {
      // Сконфигурировать контакты подсоединения шкалы как выходы
      pinMode(pinsled[i],OUTPUT);
      digitalWrite(pinsled[i],LOW);
   }
```

```
}

void loop() {
   // чтение данных потенциометра
   valpot = analogRead(POT);
   // масштабируем значение к интервалу 0-10
   countleds=map(valpot,0,1023,0,10);
   Serial.print("countleds =");
   Serial.println(countleds);
   // зажигаем количество полосок на шкале, равное countled
   for(int i=0;i<10;i++) {
      if(i<countleds) // зажигаем светодиод шкалы
         {digitalWrite(pinsled[i],HIGH);}
      else // гасим светодиод шкалы
         {digitalWrite(pinsled[i],LOW);}
   }
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_14_01**. Загружаем скетч на плату Arduino и, поворачивая ручку потенциометра, следим за показаниями светодиодной шкалы. Данные для проверки выводим в монитор последовательного порта.

Эксперимент 15.
Клавиатура по однопроводной аналоговой линии

В этом эксперименте научимся использовать аналоговый вход Arduino для подключения нескольких кнопок клавиатуры по одной линии

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino + WiFi – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Arduino LCD Keypad shield – 1;
- ❑ Светодиод – 5;
- ❑ Резистор 2,2 кОм – 1;
- ❑ Резистор 330 Ом – 1;
- ❑ Резистор 680 Ом – 1;
- ❑ Резистор 1 кОм – 1;
- ❑ Резистор 3,3 кОм – 1;
- ❑ Провода MM – 3.

Используем контроллер в режиме Arduino UNO, поэтому переключатели на плате Arduino+WiFi устанавливаем следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Рассмотрим, как можно на одном аналоговом входе Arduino создать клавиатуру из нескольких кнопок. Смотрим схему соединений на рис. 15.1.

Данная схема реализована в присутствующей в наборе плате расширения LCD Keypad Shield. Линия подключения кнопок – вход A0.

При нажатии каждая кнопка формирует определенное значение напряжения, которое после АЦП преобразуется в соответствующее значение от 0 до 1023. Таким образом, мы можем передавать информацию о нажатии разных кнопок через один пин, считывая его при помощи функции analogRead().

Значения уровня сигнала на пине A0 в зависимости от выбранной кнопки показано в таблице:

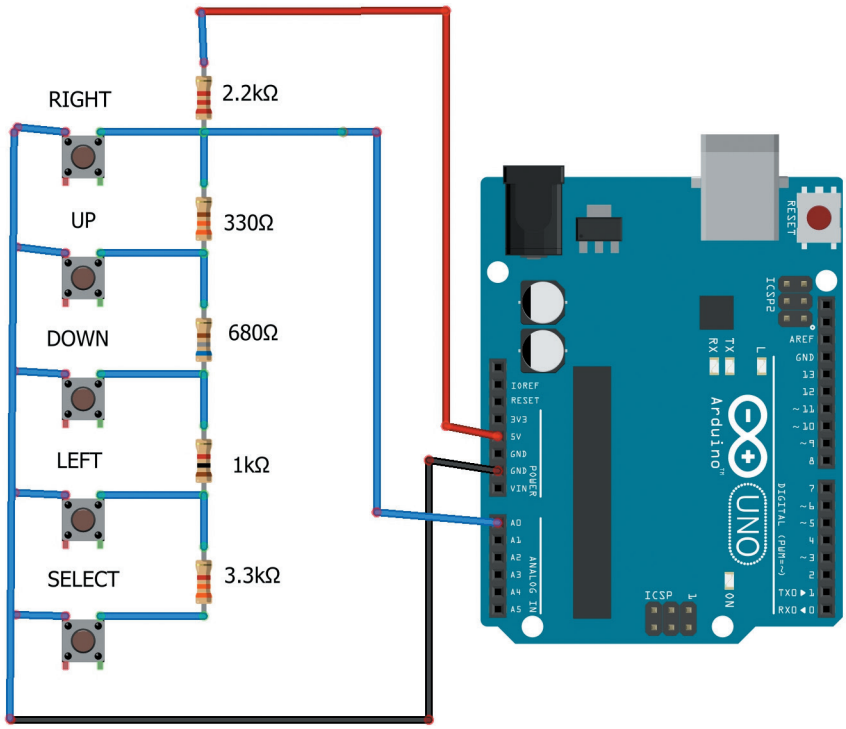


Рис. 15.1. Схема соединений для клавиатуры по однопроводной аналоговой линии

Кнопка	Значение analogRead()
RIGHT	*0-100*
UP	*100-200*
DOWN	*200-400*
LEFT	*400-600*
SELECT	*600-800*
Нет нажатия	*800-1023*

Загрузим на плату Arduino простейший скетч из листинга 15.1.

Листинг 15.1.

```
void setup() {
  Serial.begin(9600);
}
```

```
void loop() {
   Serial.println(analogRead(A0));
   delay(1000);
}
```

Откроем монитор последовательного порта и проверим соответствие значений на аналоговом порту конкретно нажатым клавишам (см. рис. 15.2).

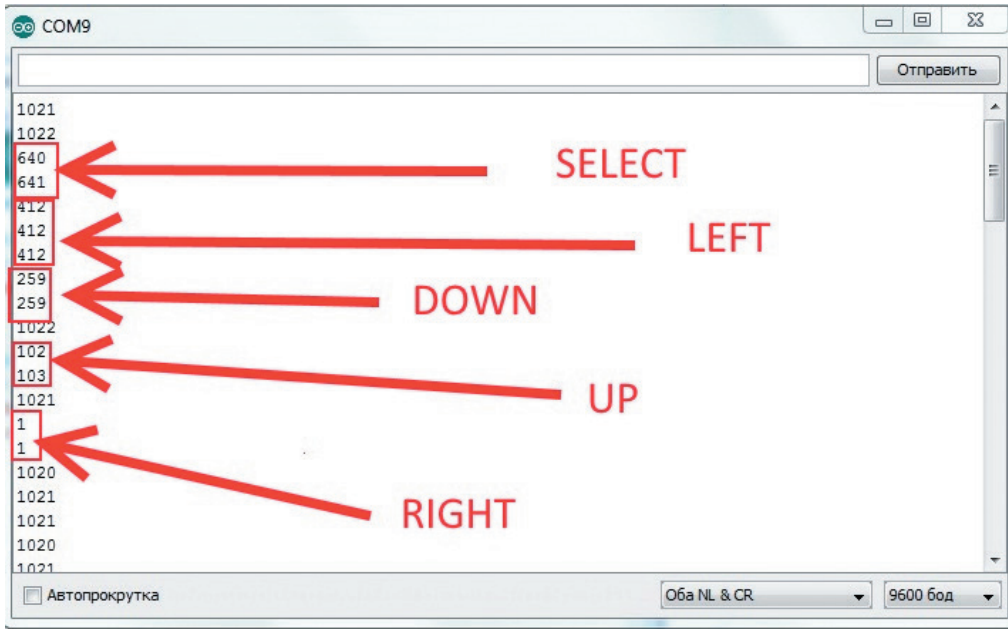


Рис. 15.2. Получаемые значения на A0 при нажатии кнопок LCD Keypad Shield

В данном методе есть два недостатка:

- нельзя отслеживать одновременное нажатие нескольких кнопок;
- возможные искажения сигнала могут привести к ложным срабатываниям.

Создадим проект управления светодиодами при помощи кнопок LCD Keypad Shield. Схема соединений показана на рис. 15.3.

Приступим к написанию скетча. При нажатии клавиши "зажигаем" соответствующий светодиод, остальные "гасим".

Содержимое скетча показано в листинге 15.2.

Листинг 15.2.

```
// переменная для хранения значения A0
int valA0 = 0;
// контактов подключения светодиодов
```

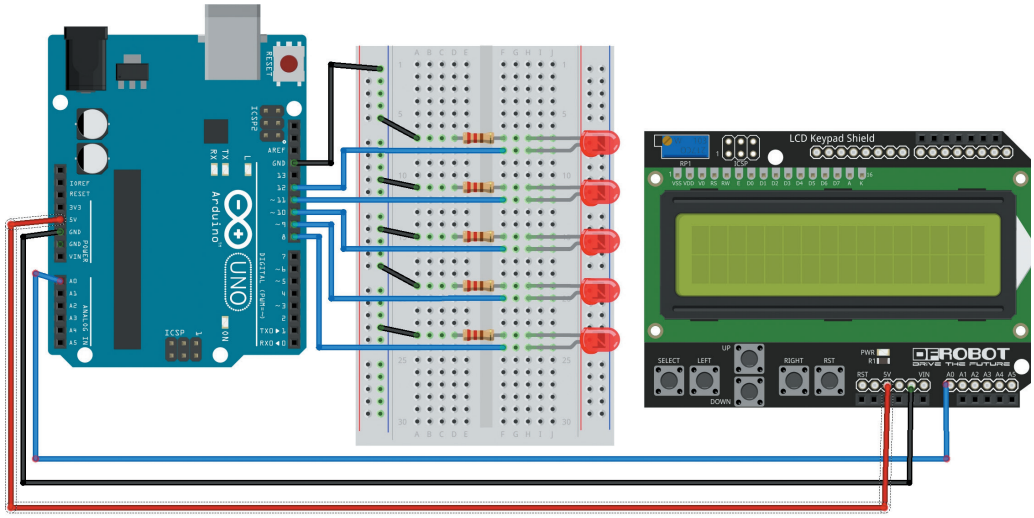


Рис. 15.3. Схема соединений для управления светодиодами по однопроводной клавиатуре

```
// RIGHT,UP,DOWN,LEFT,SELECT
const int pinsled[]={8,9,10,11,12,};


void setup() {
   // запуск последовательного порта
   Serial.begin(9600);
   for(int i=0;i<10;i++) {
      // Сконфигурировать контакты светодиодов как выходы
      // и выключить
      pinMode(pinsled[i],OUTPUT);
      digitalWrite(pinsled[i],LOW);
   }
}

void loop() {
   // чтение данных A0
   valA0 = analogRead(A0);
   // определение нажатия кнопки
   if(valA0<100) { // RIGHT
      setLed(0);
   }
   else if(valA0<200) { // UP
      setLed(0);
   }
   else if(valA0<400) { // DOWN
      setLed(0);
   }
```

```
    else if(valA0<600) { // LEFT
       setLed(0);
    }
    else if(valA0<800) { // SELECT
       setLed(0);
    }
}

// установка горящего светодиода
void setLed(int n) {
    for(int i=0;i<5;i++) {
      if(i==n)
         {digitalWrite(pinsled[i],HIGH);}
      else
         {digitalWrite(pinsled[i],LOW);}
    }
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **http://arduino-kit.ru/scetches/exp_15_02.**

Загружаем скетч на плату Arduino и нажатием кнопок устанавливаем “горящий” светодиод.

Эксперимент 16.
Широтно-импульсная модуляция. Балансир яркости двух светодиодов

В этом эксперименте мы рассмотрим широтно-импульсную модуляциию (ШИМ), которая позволяет Arduino выводить аналоговые данные на цифровые выводы.

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino + WiFi – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Светодиод – 2;
- ❑ Потенциометр – 1;
- ❑ Резистор 220 Ом – 2;
- ❑ Провода MM – 10.

Используем контроллер в режиме Arduino UNO, поэтому переключатели на плате Arduino+WiFi устанавливаем следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Плата Arduino не может на цифровой вывод выдавать произвольное напряжение. Выдается либо +5 В (HIGH), либо 0 В (LOW). Но уровнем напряжения управляется многое: например, яркость светодиода, скорость вращения мотора или звук пьезоизлучателя. Среди цифровых выводов Ардуино есть “особые” выводы, которые называются PWM. Их обозначают волнистой чертой “~”. На платах Arduino NANO и Arduino UNO ШИМ поддерживают выводы 3, 5, 6, 9, 10 и 11 (см. рис. 1.42), на плате Mega – выводы 2–13.

PWM расшифровывается, как Pulse-width modulation или широтно-импульсная модуляция (ШИМ). ШИМ – это операция получения изменяющегося аналогового значения посредством цифровых сигналов. Цифровой сигнал на выходе постоянно переключается между максимальным и минимальным значениями. Широтно-импульсно модулированный сигнал – это импульсный сигнал постоянной частоты, но переменной скважности (соотношение длительности импульса и периода его следования). Из-за того, что большинство физических процессов в природе имеют инерцию, то резкие перепады напряжения от 1 к 0 будут сглаживаться, принимая

некоторое среднее значение. С помощью задания скважности можно менять среднее напряжение на выходе ШИМ.

Переключение имеет частоту в тысячи герц. Глаз не замечает мерцания более 50 Гц, поэтому нам кажется, что светодиод не мерцает, а горит в неполную силу. Длительность включения максимального значения называется шириной импульса. Для получения различных аналоговых величин изменяется ширина импульса (см. рис. 16.1). Если скважность равняется 100%, то все время на цифровом выходе Arduino будет напряжение логическая “1» или 5 вольт. Если задать скважность 50%, то половину времени на выходе будет логическая “1», а половину – логический “0», и среднее напряжение будет равняться 2,5 вольтам. И так далее.

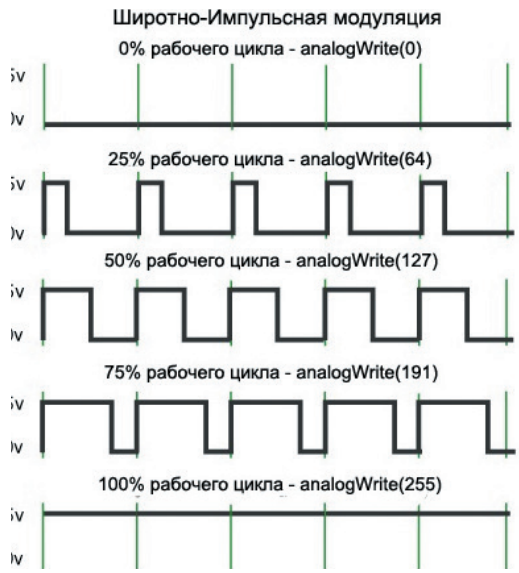


Рис. 16.1. Принцип работы широтно-импульсной модуляции (ШИМ)

В программе скважность задается не в процентах, а числом от 0 до 255. Например, команда analogWrite(10, 191) подает на цифровой выход 10 сигнал со скважностью 75% (см. рис. 16.1).

Выводы Arduino с функцией широтно-импульсной модуляции работают на частоте около 500 Гц. Функция analogWrite() никак не связана с аналоговыми входами и с функцией analogRead().

Посмотрим, как с помощью ШИМ управлять яркостью светодиодов. Подключим к ШИМ-выводам Arduino 2 светодиода, и потенциометром будем управлять яркостью двух светодиодов. Среднее положение – одинаковая (средняя) яркость двух светодиодов. Смещение ручки потенциометра уменьшает яркость левого светодиода, увеличивая яркость второго, и наоборот. Схема соединений показана на рис. 16.2.

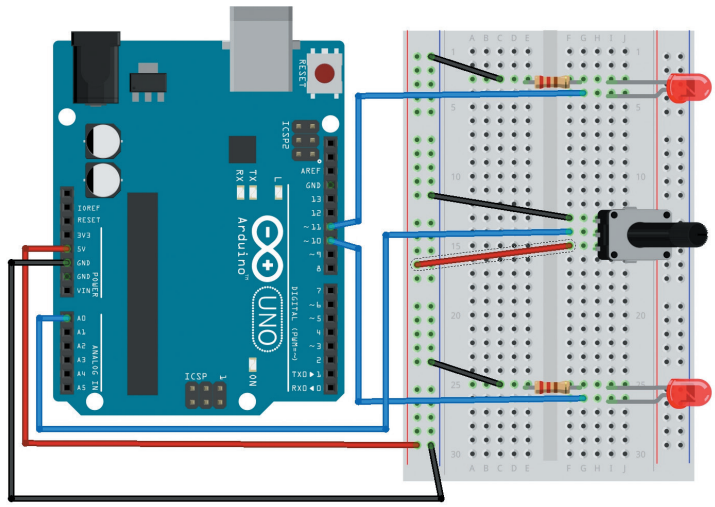


Рис. 16.2. Балансир яркости двух светодиодов

Содержимое скетча показано в листинге 16.1.

Листинг 16.1.

```
// контакты подключения потенциометра
const int pinPot=A0;
// контакты подключения светодиодов
const int pinLeftLed=11;
const int pinRightLed=11;
// переменные для хранения значений
int valPot = 0;
int valLeft = 0;
int valRight = 0;

void setup() {
   pinMode(pinLeftLed,OUTPUT);
   pinMode(pinRightLed,OUTPUT);
}

void loop() {
   // чтение данных A0
   valPot = analogRead(pinPot);
   // масштабирование
   valLeft=map(valPot,0,1023,255,0);
   valRight=map(valPot,0,1023,0,255);
   // вывод PWM
   analogWrite(pinLeftLed,valLeft);
   analogWrite(pinRightLed,valRight);
 }
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_16_01.**

Загружаем скетч на плату Arduino и потенциометром балансируем яркость двух светодиодов.

Эксперимент 17.
Радуга на RGB-светодиоде

В этом эксперименте мы узнаем
что такое RGB-светодиод,
и научимся генерировать на RGB-светодиоде
практически любой цвет

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino + WiFi – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ RGB-светодиод – 1;
- ❑ Потенциометр – 1;
- ❑ Резистор 220 Ом – 3;
- ❑ Провода MM – 10

Используем контроллер в режиме Arduino UNO, поэтому переключатели на плате Arduino+WiFi устанавливаем следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Светодиод RGB отличается от обычного тем, что содержит 3 небольших кристалла R, G, B, которые смогут синтезировать любой цвет или оттенок. RGB-светодиод имеет 4 вывода (см. рис. 17.1).

RGB расшифровывается как аббревиатура Red, Green, Blue. При помощи этих цветов можно получить любой цвет путем смешения. Подключим RGB-светодиод к плате Arduino и заставим переливаться его цветами радуги. На рис. 17.2 показана схема подключения RGB-светодиода к плате Arduino. На схеме присутствует и потенциометр, с помощью которого будем регулировать скорость изменения цветов радуги.

Теперь перейдем к написанию скетча. На самом деле радуга имеет множество цветов, а 7 цветов были придуманы только потому, что эти цвета наиболее устойчиво воспринимаются и определяются глазом.

Список этих 7 основных цветов радуги с разложением по компонентам R, G и B представлен в таблице.

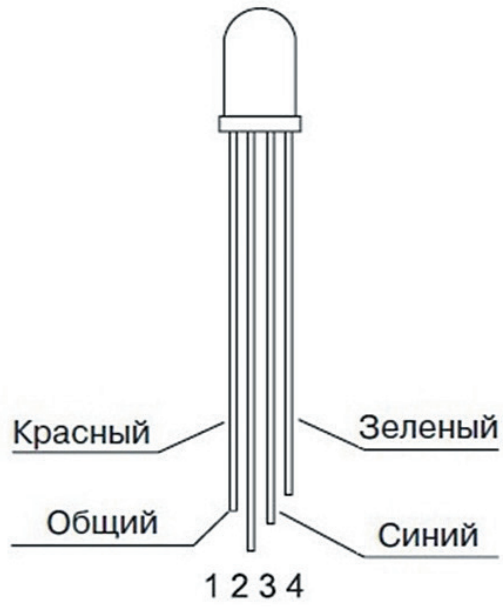


Рис. 17.1. Контакты RGB-светодиода

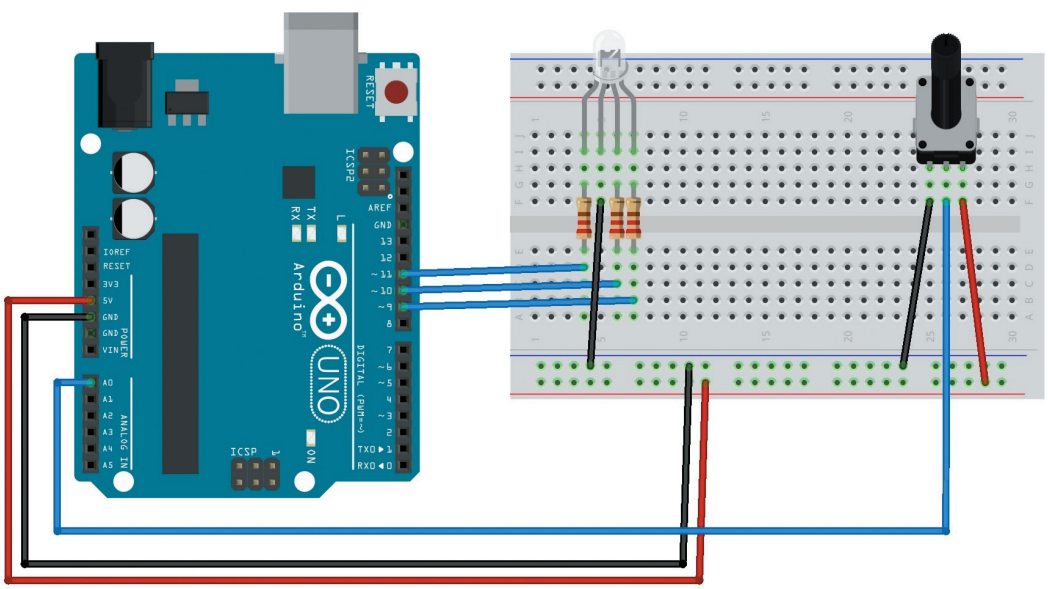


Рис. 17.2. Схема соединений RGB-светодиода для свечения цветами радуги

Цвет	R	G	B
Красный	*255*	*0*	*0*
Оранжевый	*255*	*125*	*0*
Желтый	*255*	*255*	*0*
Зеленый	*0*	*255*	*0*
Голубой	*0*	*255*	*255*
Синий	*0*	*0*	*255*
Фиолетовый	*255*	*0*	*255*

Наш светодиод должен переливаться от красного до фиолетового, проходя через все 7 основных цветов. Алгоритм вычисления любого промежуточного цвета радуги следующий:

1. Примем за начальную точку отсчета красный цвет (255, 0, 0).
2. Будем постепенно увеличивать значение зеленой составляющей G, пока не достигнем значения оранжевого (255, 125, 0), а затем и желтого цвета (255, 255, 0).
3. Постепенно уменьшим значение красной составляющей R до значения зеленого цвета (0, 255, 0).
4. Постепенно увеличим значение синей составляющей B до значения голубого цвета (0, 255, 255).
5. Постепенно уменьшим количество зеленой составляющей G до значения синего цвета (0, 0, 255).
6. Постепенно увеличим количество красной составляющей R до значения фиолетового цвета (255, 0, 255).
7. Выдерживаем небольшую паузу и переходим к шагу 1.

Содержимое скетча показано в листинге 17.1.

Листинг 17.1.

```
// пауза перед каждым изменением цвета радуги
#define MAX_PAUSE 30
#define MIN_PAUSE 1
// пин подключения среднего вывода потенциометра
const int PIN_POT=A0;
// вывод красной ноги RGB-светодиода
const int RED=11;
// вывод зеленой ноги RGB-светодиода
const int GREEN=10;
// вывод синей ноги RGB-светодиода
const int BLUE=9;
```

```
// переменная для хранения значения потенциометра
int pot;
// переменная для хранения R-составляющей цвета
int red;
// переменная для хранения G-составляющей цвета
int green;
// переменная для хранения B-составляющей цвета
int blue;

void setup()
   {;}

void loop() {
   // от красного к желтому
   red=255;green=0;blue=0;
   for(green=0;green<=255;green++)
      setRGB(red,green,blue);
   // от желтому к зеленому
   for(red=255;red>=0;red--)
      setRGB(red,green,blue);
   // от зеленого к голубому
   for(blue=0;blue<=255;blue++)
      setRGB(red,green,blue);
   // от голубого к синему
   for(green=255;green>=0;green--)
      setRGB(red,green,blue);
   // от синего к фиолетовому
   for(red=0;red<=255;red++)
      setRGB(red,green,blue);
   delay(2000);
}
// функция установки цвета RGB-светодиода
void setRGB(int r,int g,int b) {
   analogWrite(RED,r);
   analogWrite(GREEN,g);
   analogWrite(BLUE,b);
   pot=analogRead(PIN_POT);
   delay(map(pot,0,1023, MIN_PAUSE, MAX_PAUSE));
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_17_01.**

Загрузим скетч на плату, и наблюдаем на RGB-светодиоде свечение цветами радуги. С помощью потенциометра изменяем скорость изменения цветов.

Эксперимент 18.
До-ре-ми-фа-соль-ля-си. Воспроизводим звуки на Arduino

В этом эксперименте мы научимся воспроизводить на плате Arduino простейшие звуки – ноты.

В эксперименте мы будем использовать следующие компоненты:

- Плата Arduino + WiFi – 1;
- Кабель USB;
- Плата прототипирования – 1;
- Динамик – 1;
- Потенциометр – 1;
- Провода MF – 2.

Используем контроллер в режиме Arduino UNO, поэтому переключатели на плате Arduino+WiFi устанавливаем следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

У нас в наборе есть два динамика! Неплохо было бы озвучить наши проекты. И так, подключаем один из динамиков к цифровому выходу платы Arduino.

Для генерации звуков определенной частоты и длительности будем использовать Arduino-функцию tone():

tone(pin,frequency,duration);

Функция tone() генерирует на выводе прямоугольный сигнал заданной частоты (с коэффициентом заполнения 50%). Функция также позволяет задавать длительность сигнала. Если длительность сигнала не указана, он будет генерироваться до тех пор, пока не будет вызвана функция noTone() или другая функция tone().

Попробуем воспроизвести последовательность нот до, ре, ми, фа, соль, ля, си.

Схема соединений показана на рис. 18.1.

С помощью потенциометра будем регулировать скорость воспроизведения последовательности нот.

Значения частот для нот первой октавы следующее:

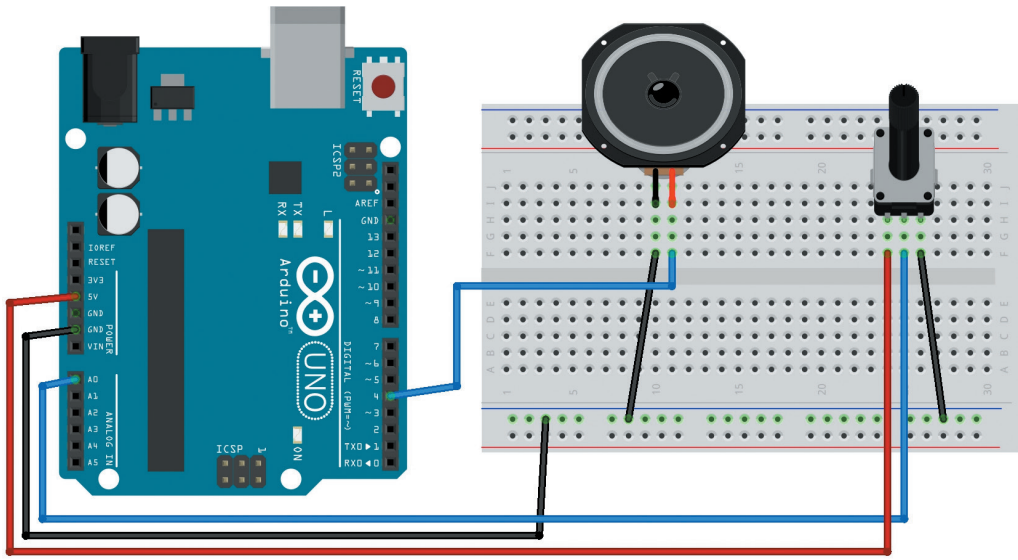


Рис. 18.1. Схема соединений для подключения динамика

до	ре	ми	фа	соль	ля	си
261	*293*	*329*	*349*	*392*	*440*	*494*

Частоты нот занесем в массив:

```
int octave1[]={261,293,329,349,392,440,494};
```

Воспроизводим ноты последовательно. Команда tone() запускает на контакте прямоугольную волну и сразу же переходит к выполнению следующего оператора, поэтому будем ставить перед воспроизведением следующей ноты оператор delay() с длительностью воспроизведения ноты и паузы:

```
tone(4,392, duration);
delay(duration);      // ждем окончания воспроизведения ноты
delay(duration/2);    // пауза между воспроизведением нот
```

Содержимое скетча показано в листинге 18.1.

Листинг 18.1.

```
// длительность воспроизведения ноты
unsigned long duration;
// max и min значение длительности
#define MAX_DURATION 3000
#define MIN_DURATION 300
```

```
// пин подключения динамика
const int pinSpeaker=4;
// массив частот для нот первой октавы
// {до, ре, ми, фа, соль, ля, си}
int octave1[]={261,293,329,349,392,440,494};
// пин подключения потенциометра
const int pinPot=A0;

void setup() {
   // сконфигурировать контакт как выход
   pinMode(pinSpeaker,OUTPUT);
}
void loop() {
   // последовательное воспроизведение звуков
   for(int i=0;i<7;i++) {
      // вычисляем скорость воспроизведения
      // (длительность ноты)
      int val=analogRead(pinPot);
      duration=map(val,0,1023,MIN_DURATION,MAX_DURATION);
      // воспроизведение ноты
      tone(pinSpeaker,octave1[i],duration);
      delay(duration);
      delay(duration/2);
   }
   // пауза перед следующим воспроизведением звукоряда
   delay(duration*5);
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_18_01.**

Загрузим скетч на плату, и слушаем звукоряд до, ре, ми, фа, соль, ля, си, с помощью потенциометра регулируем скорость воспроизведения звукоряда.

Эксперимент 19.
Воспроизводим звуки разных октав. Двумерные массивы

В этом эксперименте мы продолжим воспроизводить ноты, научимся создавать двумерные массивы и работать с ними

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino + WiFi – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Динамик – 2;
- ❑ Потенциометр – 2;
- ❑ Провода MM – 6;
- ❑ Провода MF – 2.

Используем контроллер в режиме Arduino UNO, поэтому переключатели на плате Arduino+WiFi устанавливаем следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

В эксперименте 18 мы последовательно воспроизводили ноты в пределах одной октавы. Октава — музыкальный интервал, в котором соотношение частот между звуками составляет один к двум (то есть частота высокого звука в два раза больше низкого). Субъективно на слух октава воспринимается как устойчивый, базисный музыкальный интервал. Два звука, отстоящие на октаву, воспринимаются очень похожими друг на друга, хотя явно различаются по высоте. После ноты си идет нота до, и далее ре, ми …., но уже следующей октавы.

В этом эксперименте мы добавляем еще один потенциометр, вращением которого будем выбирать октавы. Кроме того, задействуем второй динамик. В один момент времени может выполняться только одна функция tone(), поэтому одновременно можно выводить звуки на один динамик. Воспроизведение одних октав будет идти на первый динамик, других – на второй.

Схема соединений показана на рис. 19.1.

Значения частот для нот нескольких октав показано в таблице:

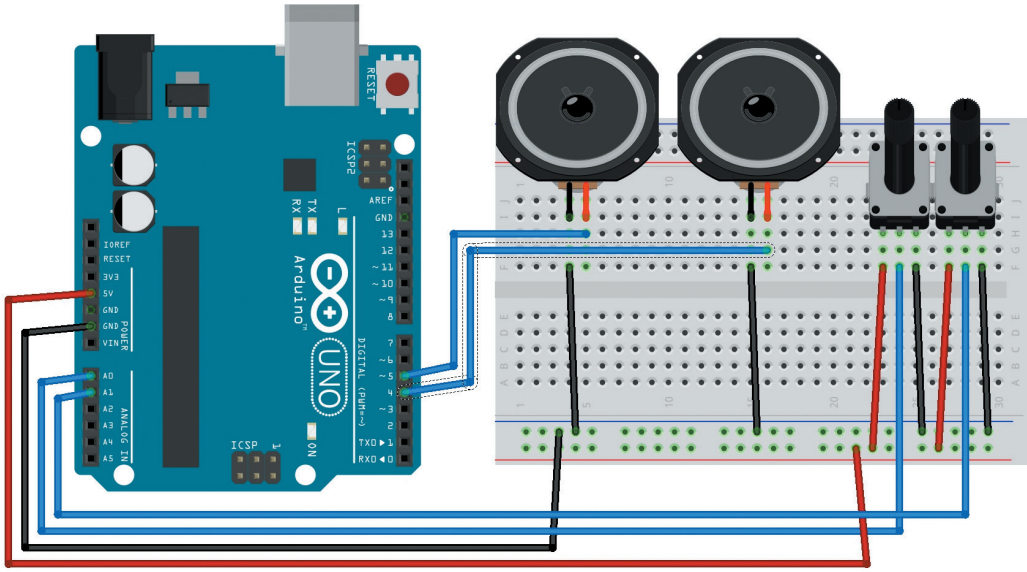


Рис. 19.1. Схема соединений для воспроизведения нот нескольких октав

	До 0	Ре 1	Ми 2	Фа 3	Соль 4	Ля 5	Си 6
Малая октава	*130*	*146*	*165*	*175*	*196*	*220*	*247*
1 октава	*261*	*293*	*329*	*349*	*392*	*440*	*494*
2 октава	*523*	*587*	*659*	*698*	*784*	*880*	*988*
3 октава	*1047*	*1175*	*1319*	*1397*	*1568*	*1760*	*1975*

Мы можем сформировать ноты по октавам в виде массивов, как в эксперименте 18. Получится 4 массива:

```
int octavem[]={130,146,165,175,196,220,247};
int octave1[]={261,293,329,349,392,440,494};
int octave2[]={523,587,659,698,784,880,988};
int octave3[]={1047,1175,1319,1397,1568,1760,1975};
```

Но можно сделать и по-другому. Посмотрите на таблицу. Массив частот звуков для каждой октавы – это одна строка в таблице, где нумерация элементов массива – это номер столбца от 0 до 6. Это одномерные массивы. Место каждого элемента в этих массивах однозначно определяется с помощью одной позиции (столбца в строке).

В двумерном массиве место каждого элемента однозначно определяется с помощью двух позиций (строки и столбца в строке). Создадим двумерный массив для нот четырех октав. Записывается так:

```
int octaves [4][7]={
                  {130,146,165,175,196,220,247},
                  {261,293,329,349,392,440,494},
                  {523,587,659,698,784,880,988},
                  {1047,1175,1319,1397,1568,1760,1975}
};
```

Как получить элемент двумерного массива? Например:

octave [0][5] – ля малой октавы,

octave [2][2] – ми второй октавы

Номер октавы мы будем получать масштабированием 0-3 значений второго потенциометра. Четные октавы выводим на левый динамик, нечетные – на правый.

И содержимое скетча показано в листинге 19.1.

Листинг 19.1.

```
// длительность воспроизведения ноты
unsigned long duration;
// номер воспроизводимой октавы
int octave;
// номер динамика
int speaker;
// max и min значение длительности
#define MAX_DURATION 3000
#define MIN_DURATION 300
// пины подключения динамиков
const int pinSpeakers[]={4,5};
// массив частот для нот
// {до, ре, ми, фа, соль, ля, си}
int octaves [4][7]={
   {130,146,165,175,196,220,247}, // малая октава
   {261,293,329,349,392,440,494}, // первая октава
   {523,587,659,698,784,880,988}, // вторая октава
   {1047,1175,1319,1397,1568,1760,1975} // третья октава
};

// пин подключения потенциометра длительности
const int pinPot=A0;
// пин подключения потенциометра выбора октав
const int pinChoiceOctaves=A1;
```

```
void setup() {
   // сконфигурировать контакты динамиков как выход
   pinMode(pinSpeakers[0],OUTPUT);
   pinMode(pinSpeakers[1],OUTPUT);
}
void loop() {
   // последовательное воспроизведение звуков
   for(int i=0;i<7;i++) {
      // вычисляем скорость воспроизведения
      int val=analogRead(pinPot);
      duration=map(val,0,1023,MIN_DURATION,MAX_DURATION);
      // вычисляем номер октавы
      val=analogRead(pinChoiceOctaves);
      octave=map(val,0,1023,0,3);
      // номер динамика
      speaker=pinSpeakers[octave%2];
      // воспроизведение ноты
      tone(speaker,octaves[octave][i],duration);
      delay(duration);
      delay(duration/2);
   }
   // пауза перед следующим воспроизведением звукоряда
   delay(duration*2);
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_19_01.**

Загрузим скетч на плату, и слушаем звукоряд до, ре, ми, фа, соль, ля, си, С помощью первого потенциометра регулируем скорость воспроизведения звукоряда, а вращением второго потенциометра выбираем октаву, при этом соседние октавы воспроизводятся на разных динамиках.

Эксперимент 20.
Музыкальный звонок

В этом эксперименте мы создадим музыкальный звонок, который будет воспроизводить мелодию

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino + WiFi – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Динамик – 1;
- ❑ Потенциометр – 1;
- ❑ Блок клавиатуры 4x4 – 1;
- ❑ Резистор 10 кОм – 1;
- ❑ Провода MM – 3;
- ❑ Провода MF – 2.

Используем контроллер в режиме Arduino UNO, поэтому переключатели на плате Arduino+WiFi устанавливаем следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Создадим музыкальный звонок, который воспроизводит мелодию при нажатии кнопки.

Схема соединений показана на рис. 20.1.

Подберем мелодию, которую будет воспроизводить музыкальная наш музыкальный звонок. Находим мелодию, записанную на нотном стане. Например, очень простую – “В траве сидел кузнечик” (ноты на рисунке 20.2).

Нам понадобятся ноты 1 и 2 октавы. Каждая нота имеет общепринятое обозначение (см. таблицу). Нот в октаве 12 (кроме до, ре, ми, фа, соль, ля, си есть еще промежуточные до-диез, ре-диез, фа—диез, соль-диез, си-бимоль). Обозначения для каждой ноты (см. таблицу 1) общеспринятые, но мы для уменьшения сложности программы промежуточным нотам назначим свои обозначения.

Занесем в массив melody[] последовательность воспроизводимых нот, список длительностей нот занесем в массив duration[], при этом длительность целой ноты

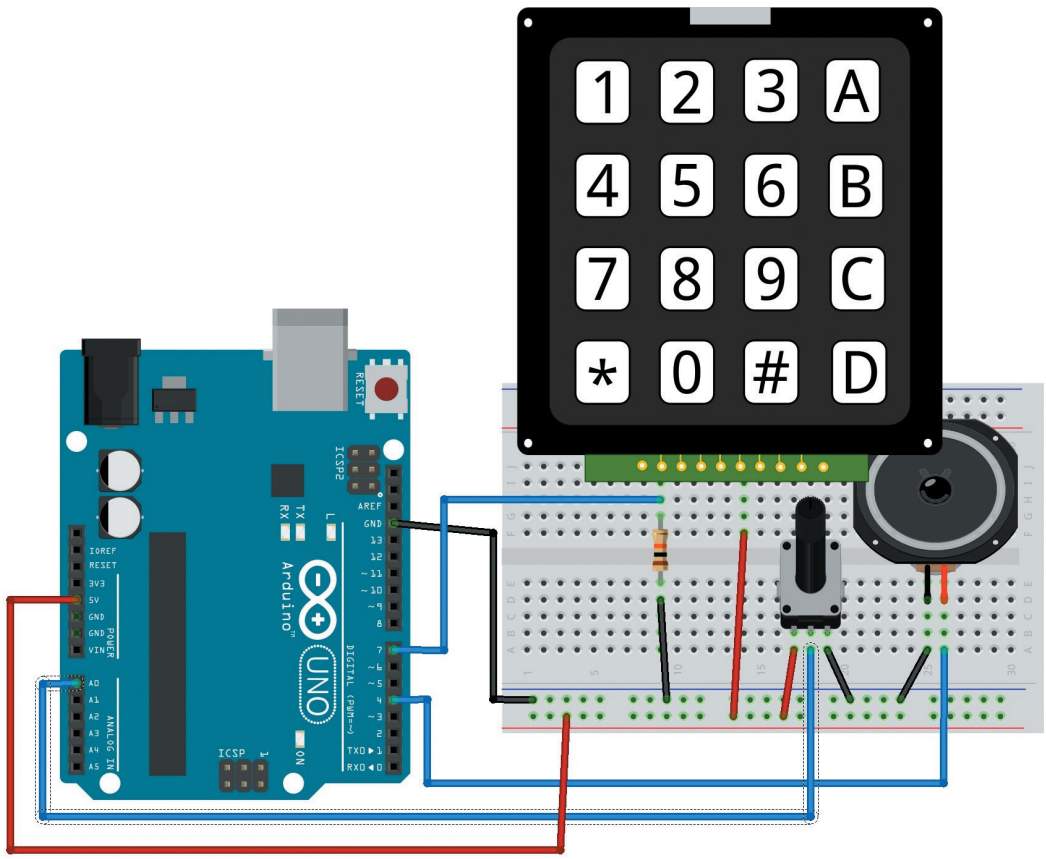


Рис. 20.1. Схема соединений для музыкального звонка

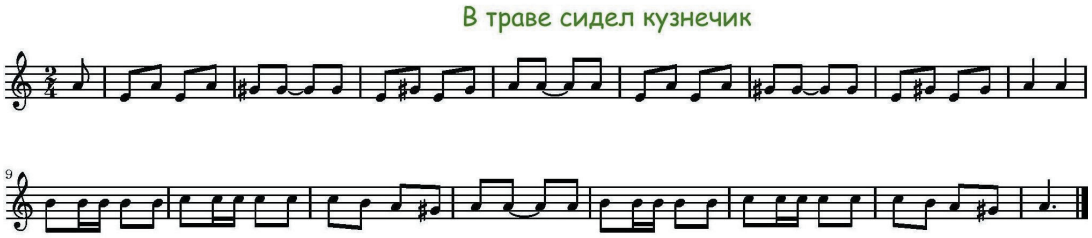


Рис. 20.2. Мелодия «В траве сидел кузнечик» на нотном стане

равна 32, половинной 16, и т.д. до 1/32 – длительность1. Данные с обозначением нот занесем в массив notes[], а данные с частотами для соответствующих нот занесем в массив frequency[]. Паузу обозначим символом '*'.

1 октава	Обозначение	Частота, Гц	2 октава	Обозначение	Частота, Гц
до	*C*	*261*	*до*	*c*	*523*
до-диез	*C#(R)*	*277*	*до-диез*	*c#(r)*	*554*
ре	*D*	*293*	*ре*	*d*	*587*
ре-диез	*D#(S)*	*311*	*ре-диез*	*d#(s)*	*622*
ми	*E*	*329*	*ми*	*e*	*659*
фа	*F*	*349*	*фа*	*f*	*698*
фа-диез	*F#(T)*	*370*	*фа-диез*	*f#(t)*	*740*
соль	*G*	*392*	*соль*	*g*	*784*
соль-диез	*G#(U)*	*415*	*соль-диез*	*g#(u)*	*830*
ля	*A*	*440*	*ля*	*a*	*880*
си-бимоль	*B*	*466*	*си-бимоль*	*b*	*932*
си	*H*	*494*	*си*	*h*	*988*

```
// МЕЛОДИЯ – массив нот и массив длительностей
char melody[]={'A', 'E','A','E','A', 'U','G','G','G',
                'E','U','E','G', 'A','A','A','A',
                'E','A','E','A', 'U','G','G','G',
                'E','U','E','G', 'A','A',

                'H','H','H','H','H', 'c','c','c','c','c',
                'c','H','A','U', 'A','A','A','A',
                'H','H','H','H','H', 'c','c','c','c','c',
                'c','H','A','U', 'A', '*'
};
int bb[]={4, 4,4,4,4, 4,4,4,4,
        4,4,4,4, 4,4,4,4,
        4,4,4,4, 4,4,4,4,
        4,4,4,4, 8,8,

        4,2,2,4,4, 4,2,2,4,4,
        4,4,4,4, 4,4,4,4,
```

```
        4,2,2,4,4, 4,2,2,4,4,
        4,4,4,4, 4,64
};
// массив для наименований нот в пределах двух октав
char names[]={'c','r','d','s','e','f','t','g','u','a','b',
'h','C','R','D','S','E','F','T','G','U','A','B', 'H','F'};
// массив частот нот
int tones[]={261,277,293,311,329,349,370,392,415,440,466,
494, 523,554,587,622,659,698,740,784,830,880,932,988};
```

В цикле loop() мы проверяем нажатие кнопки, по которому начинается проигрывание мелодии понотно, вызывая процедуру playNote(), в которую передает обозначение текущей ноты и продолжительность. Процедура playNote() находит по обозначению ноты значение соответствующей частоты и вызывает для проигрывания ноты функцию tone(). Продолжительность звучания ноты – это базовая нотная длительность (32 – для целой, 16 – для полуноты, 8 – для четвертной и т.д.) умноженная на значение темпа произведения – temp. Значение temp устанавливается по значению потенциометра, и подбирается при отладке, в работе не меняется. Для лучшего звучания музыкального отрезка после воспроизведения каждой ноты делаем небольшую паузу

```
delay(beats*tempo+tempo);
```

Мелодия воспроизводится до отжатия кнопки, при этом воспроизведение прекращается до следующего нажатия.

Содержимое скетча показано в листинге 20.1.

Листинг 20.1.

```
// МЕЛОДИЯ - массив нот и массив длительностей
char melody[]={'A', 'E','A','E','A', 'U','G','G','G',
                    'E','U','E','G', 'A','A','A','A',
                    'E','A','E','A', 'U','G','G','G',
                    'E','U','E','G', 'A','A',

                    'H','H','H','H','H', 'c','c','c','c','c',
                    'c','H','A','U', 'A','A','A','A',
                    'H','H','H','H','H', 'c','c','c','c','c',
                    'c','H','A','U', 'A', '*'
};
int bb[]={4, 4,4,4,4, 4,4,4,4,
        4,4,4,4, 4,4,4,4,
        4,4,4,4, 4,4,4,4,
        4,4,4,4, 8,8,

        4,2,2,4,4, 4,2,2,4,4,
```

```
        4,4,4,4, 4,4,4,4,
        4,2,2,4,4, 4,2,2,4,4,
        4,4,4,4, 4,64
};
// пин подключения динамика
#define PIN_SPEAKER 4
// переменные - темп воспроизведения, ноты, длительности
int tempo,notes,beats;
#define MIN_TEMPO 20
#define MAX_TEMPO 100
// пин подключения потенциометра
#define PIN_POT A0
// пин подключения кнопки
#define PIN_BUTTON 7
// Переменная для сохранения текущего состояния кнопки
int tekButton = LOW;
// Переменная для сохранения предыдущего состояния
int prevButton = LOW;

void setup()  {
   // Сконфигурировать контакт динамика как выход
   pinMode(PIN_SPEAKER, OUTPUT);
   // Сконфигурировать контакт кнопки как вход
   pinMode (PIN_BUTTON, INPUT);
}

void loop()  {
   tekButton = debounce(prevButton);
   if (prevButton == LOW && tekButton == HIGH) // если нажатие...
      {
      // получить темп воспроизведения
      tempo=map(analogRead(PIN_POT),0,1024,MIN_TEMPO,MAX_TEMPO);
      for(int i=0;i<sizeof(melody);i++)  {
         notes=melody[i];
         beats=bb[i];
         if (notes == '*')
            tone(PIN_SPEAKER,0, beats*tempo); // пауза
         else
            playNote(notes, beats*tempo); // воспроизвести ноту
         // пауза между нотами
         delay(beats*tempo+tempo);
         // проверить на отжатие
         tekButton = debounce(prevButton);
         if (prevButton == HIGH  && tekButton == LOW) }
```

```
            i=sizeof(melody); // прекратить
            tekButton == LOW;
         }
         prevButton = tekButton;
         // получить темп воспроизведения
         tempo=map(analogRead(PIN_POT),0,1024,MIN_TEMPO,MAX_TEMPO);
      }
   }
   prevButton = tekButton;
}
// процедура проигрыша ноты
void playNote(char note, int duration) {
   // массив для наименований нот в пределах двух октав
   char names[]={'c','r','d','s','e','f','t','g','u','a','b',
   'h','C','R','D','S','E','F','T','G','U','A','B', 'H','F'};
   // массив частот нот
   int tones[]={261,277,293,311,329,349,370,392,415,440,466,
   494, 523,554,587,622,659,698,740,784,830,880,932,988};
   for (int i = 0; i < sizeof(tones); i++) {
      if (names[i] == note)  {
         tone(PIN_SPEAKER,tones[i],duration);
      }
   }
}
// Функция сглаживания дребезга.
boolean debounce(boolean last) {
   // Считать состояние кнопки,
   boolean current = digitalRead(PIN_BUTTON);
   if (last != current) // если изменилось...
      {
      delay(5); // ждем 5 мс
      current = digitalRead(PIN_BUTTON);
      return current;
   }
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_20_01.zip.**

Загрузим скетч на плату, и проверяем работу звонка.

Эксперимент 21.
Библиотеки Arduino. Создание собственной библиотеки

В этом эксперименте мы познакомимся с библиотеками Arduino, и создадим собственную библиотеку для работы с семисегментным светодиодным индикатором.

В эксперименте 10 мы работали с семисегментным светодиодным индикатором, который позволяет Arduino визуализировать цифры. Сегодня соберем ту же схему. Список деталей:

- Плата Arduino UNO – 1;
- Кабель USB;
- Плата прототипирования – 1;
- Индикатор 7-сегментный одноразрядный – 1;
- Резистор 220 Ом – 8;
- Провода MM – 15.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

И схема подключения индикатора к плате Arduino на рис. 21.1.

Возможности среды программирования Arduino могут быть существенно расширены за счет использования библиотек. Библиотеки расширяют функциональность программ и несут в себе дополнительные функции, например, для работы с аппаратными средствами, функции по обработке данных и т.д. Ряд библиотек устанавливается автоматически вместе со средой разработки, однако вы также можете скачивать или создавать собственные библиотеки. Использование библиотек существенно упрощает работу над проектами, потому что можно сосредоточиться на основной логике программы, не тратя время на множество мелочей. Сегодня огромное количество библиотек выложено в интернете, где их можно легко скачать, причем совершенно бесплатно.

Для того чтобы подключить библиотеку, нужно написать всего одну строку в начале скетча:

```
#include <name_library.h>
```

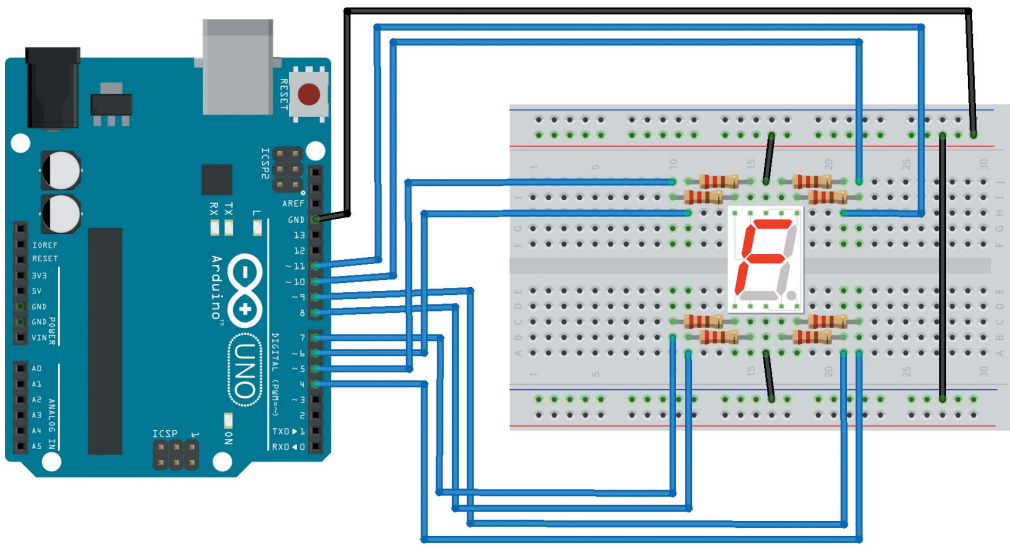


Рис. 21.1. Схема соединений для подключения индикатора к плате Arduino

например:

```
#include < LiquidCrystal.h>
```

Некоторые библиотеки работают, используя методы и функции других библиотек, тогда нужно подключать две библиотеки, сначала подключается та, методы и функции которой использует вторая, например:

```
// Подключение библиотеки Wire для работы с шиной I2C
 #include <Wire.h>
// Подключение библиотеки LiquidCrystal_I2C
#include <LiquidCrystal_I2C.h>
```

Для работы с большинством библиотек, нужно создать объект (экземпляр класса библиотеки), через который будут доступны их функции и методы, например:

```
LiquidCrystal_I2C lcd(0x27,20,4);
```

lcd это объект библиотеки LiquidCrystal_I2C, через объект обращаются к функциям и методам библиотеки.

Научимся создавать собственную библиотеку. В главе 6.5. мы рассматривали работу со светодиодным индикатором D5651. Создадим библиотеку, которая будет упрощать вывод цифр на данный индикатор. Назовем ее D5651.

Библиотека должна иметь как минимум два файла:

заголовочный файл (расширение .h);

файл с исходным кодом (расширение .cpp).

В первом файле содержится описание самого класса, переменные, константы. А второй файл содержит программный код методов.

Создадим заголовочный файл D5651.h.

Все остальное содержимое h-файла необходимо заключить в конструкцию, исключающую повторное подключение библиотеки:

```
#ifndef Button_h  // если библиотека Button не подключена
#define Button_h  // тогда подключаем ее
// ............
#endif
```

Внутри конструкции необходимо включить файл Arduino.h, содержащий стандартные константы и переменные языка Ардуино. В обычных программах он добавляется автоматически, а для библиотеки должен быть указан явно следует написать:

```
#include "Arduino.h"
```

Необходимо добавить описание класса. Класс – это набор функций и переменных, объеденных в одном месте. Функции и переменные могут быть публичными (public), что означает общий доступ к ним всех, кто использует библиотеку, или частными (private), что означает доступ к ним только внутри класса. Конструктор имеет тоже имя, что и класс, но не имеет типа возвращаемого значения.

```
class D5651
{
  public:
    D5651(byte *pins);
    void setNumber(int num);
  private:
    byte *pinsS;
};
Содержимое файла приведено в листинге 21.1.
Листинг 21.1.
#ifndef D5651_h
#define D5651_h

#include "Arduino.h"

class D5651
{
  public:
    D5651(byte *pins);
    void setNumber(int num);
  private:
    byte *pinsS;
};

#endif
```

Рассмотрим файл реализации D5651.cpp.

В начале кода находится директива #include. Разрешается доступ к характеристикам в головном файле библиотеки:

```
#include <D5651.h>
```

Далее по коду находится конструктор. Он используется для создания экземпляра создаваемого класса.

```
D5651::D5651(byte *pins) {
   for(int i=0;i<8;i++) {
     pinsS= pins;
     for(int i=0;i<8;i++)
      pinMode(pinsS[i], OUTPUT);  // определяем вывод как вход
   }
}
```

Далее реализация метода. Код D5651:: означает, что функция принадлежит классу D5651.

Содержимое файла приведено в листинге 21.2.

Листинг 21.2.

```
#include <D5651.h>

// описание конструктора класса D5651
D5651::D5651(byte *pins) {
   for(int i=0;i<8;i++) {
     pinsS= pins;
     for(int i=0;i<8;i++)
      pinMode(pinsS[i], OUTPUT);  // определяем вывод как вход
   }
}

// вывод цифры
void D5651::setNumber(int num) {
byte numbers[10] = { B11111100,   // 0
                     B01100000,   // 1
                     B11011010,   // 2
                     B11110010,   // 3
                     B01100110,   // 4
                     B10110110,   // 5
                     B10111110,   // 6
                     B11100000,   // 7
                     B11111110,   // 8
                     B11100110    // 9
};
   for(int i=0;i<7;i++) {
      if(bitRead(numbers[num],7-i)==HIGH) // зажечь сегмент
         digitalWrite(pinsS[i],HIGH);
      else // потушить сегмент
         digitalWrite(pinsS[i],LOW);
   }
}
```

В папке examples можно создавать примеры использования библиотеки. Пример в листинге 21.3.

Листинг 21.3.

```
#include <D5651.h>

byte p[]={2,3,4,5,6,7,8,9};
D5651 s(p);

void setup() {;}

void loop() {
  for(int i=0;i<10;i++) {
    p.setNumber(i);
    delay(1000);
  }
}
```

Для того чтобы Arduino IDE выделяла цветом новые типы и методы из нашей библиотеки необходимо создать файл keywords.txt.

```
D5651      KEYWORD1
setNumber  KEYWORD2
```

Каждая строка содержит ключевое слово, табуляцию (не пробелы) и тип ключевого слова. KEYWORD1 определяет классы, KEYWORD2 – методы.

Скачать данную библиотеку можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/ D5651.zip.**

Эксперимент 22.
Матричная клавиатура 4x4

В этом эксперименте мы подключим матричную клавиатуру 4x4, которая позволяет экономить входы Arduino.

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Клавиатура матричная 4x4 – 1;
- ❑ Провода MM – 8

Переключатели на плате Arduino+WiFi установите следующим образом

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Иногда мы сталкиваемся с проблемой нехватки портов на Arduino. Для этого была придумана матричная клавиатура. Такая клавиатура используется, когда требуется большое количество кнопок.

Самыми распространенными являются 16 кнопочные клавиатуры 4x4. Принцип их работы достаточно прост, Arduino поочередно подает логическую единицу на каждый 4 столбцов, в этот момент 4 входа Arduino считывают значения.

Принципиальная схема соединений клавиатуры 4x4 показана на рис.22.1.

Заметьтим, чтобы входы 9, 8, 7, 6 не подтянуты ни к "земле", ни к питанию, будем использовать внутренний подтягивающий резистор 20кОм:

```
pinMode(9, INPUT_PULLUP);
```

И фактическая схема подключения на рис. 22.2

Для работы с матричными клавиатурами удобно использовать библиотеку Keypad. Скачать данную библиотеку можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/Keypad.zip.** Затем в Arduino IDE выбираем пункт меню Эскиз → Include Library → Add /ZIP library... и в появившемся окне выбираем путь к скачанному zip-архиву и нажимаем кнопку Open.

Выбранная библиотека должна появиться в списке установленных библиотек, zip-файл будет распакован в папке libraries, находящейся в папке со скетчами.

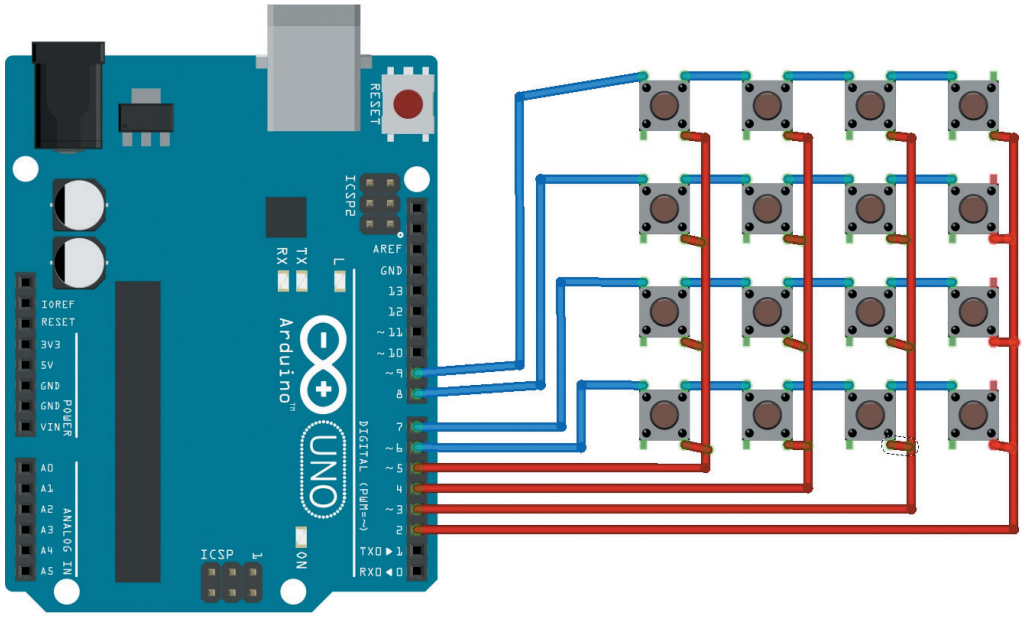


Рис. 22.1. Схема принципиальная для подключения клавиатуры к плате Arduino

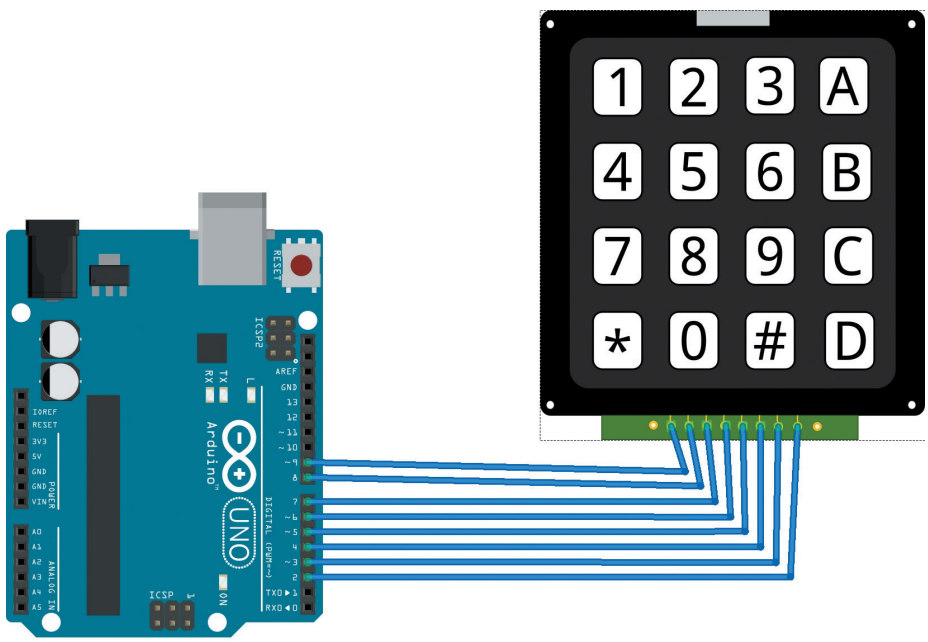


Рис. 22.2. Схема подключения клавиатуры к плате Arduino

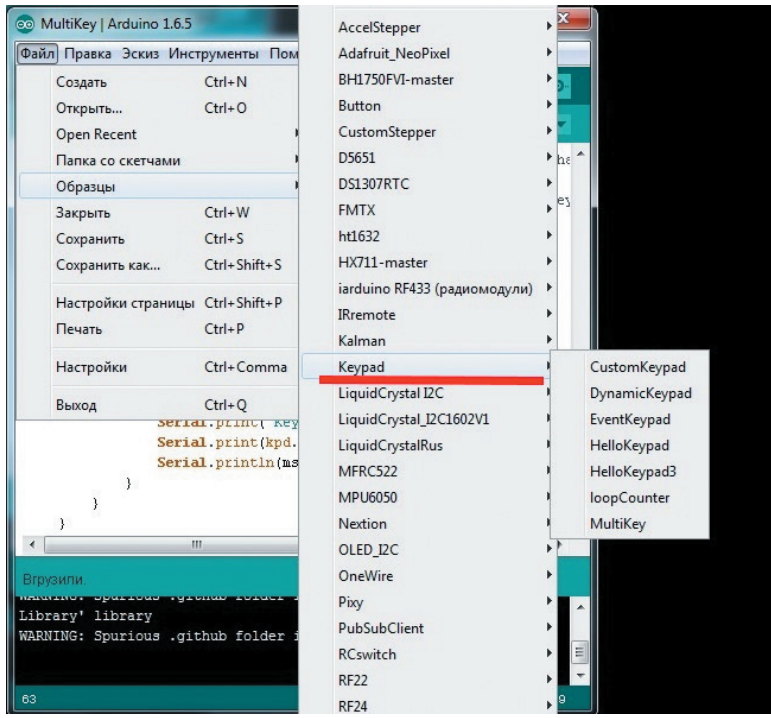


Рис. 22.3. Библиотека установлена

Добавленную библиотеку можно будет использовать в скетчах, примеры появятся в меню Файл→ Образцы только после перезагрузки Arduino IDE (рис. 22.3).

Библиотека поддерживает множественные нажатия, определяет нажатие, отжатие и долгое нажатие каждой клавиши. В листинге 22.1 приведен скетч слежения за всеми клавишами и выводом изменения их состояний в монитор последовательного порта. Необходимо назначить каждой кнопке определенный символ.

кнопка	символ	кнопка	символ
K1	*1*	*K9*	*7*
K2	*2*	*K10*	*8*
K3	*3*	*K11*	*9*
K4	*A*	*K12*	*C*
K5	*4*	*K13*	*
K6	*5*	*K14*	*0*
K7	*6*	*K15*	#
K8	*B*	*K16*	*D*

И содержимое скетча в листинге 22.1.

Листинг 22.1.

```
// подключение библиотеки
#include <Keypad.h>
// размеры клавиатуры 4x4
const byte ROWS = 4;
const byte COLS = 4;
// символы для клавиш
char keys[ROWS][COLS] = {
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};
// контакты считывания
byte rowPins[ROWS] = {9, 8, 7, 6};
// контакты подачи 1
byte colPins[COLS] = {5, 4, 3, 2};
// создание объекта
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins,
                        ROWS, COLS );
// для формирования сообщений
String msg;


void setup() {
    // запуск последовательного порта
    Serial.begin(9600);
    msg = "";
    // длинное нажатие - 5 сек
    kpd.setHoldTime(5000);
}

void loop() {
    if (kpd.getKeys())
    {
        // сканирование массива состояний кнопок
        for (int i=0; i<LIST_MAX; i++)
        {
            // только с измененным статусом
            if ( kpd.key[i].stateChanged )
            {
              // получение состояния клавиш
              // IDLE, PRESSED, HOLD, or RELEASED
                switch (kpd.key[i].kstate) {
                    case PRESSED:
```

```
                    msg = " PRESSED.";
                break;
                    case HOLD:
                    msg = " HOLD.";
                break;
                    case RELEASED:
                    msg = " RELEASED.";
                break;
                    case IDLE:
                    msg = " IDLE.";
                }
                // вывод состояния кнопок
                Serial.print("Key ");
                Serial.print(kpd.key[i].kchar);
                Serial.println(msg);
            }
        }
    }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_22_01.zip.**

Загружаем скетч на плату Arduino и открываем монитор последовательного порта (см. рис. 22.4).

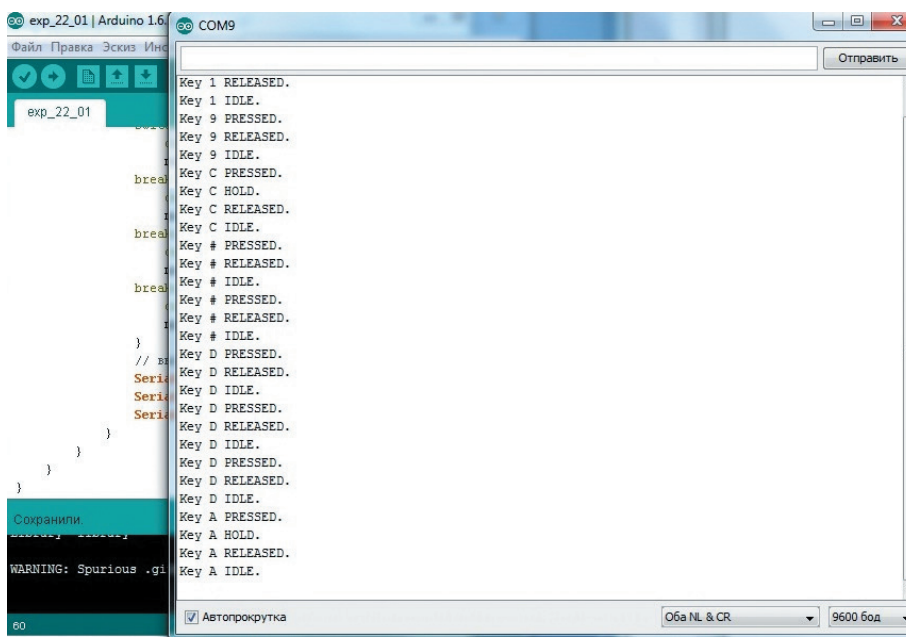


Рис. 22.4. Вывод данных состояний кнопок

Эксперимент 23.
Пианино на матричной клавиатуре

В этом эксперименте мы создадим маленькое пианино на матричной клавиатуре

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Клавиатура матричная 4x4 – 1;
- ❑ Динамик – 1;
- ❑ Провода MM – 8.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Создадим мини-пианино на матричкой клавиатуре. Будем воспроизводить только основные ноты (до, ре, ми, фа, соль, ля, си) 1 и 2 октавы.

Схема соединений показана на рис. 23.1.

Т.к. в один момент времени возможно выполнение только одной функции tone(), определим следующие правила воспроизведения:

- ❑ нажатие кнопки приводит к воспроизведению соответствующей ноты, отпускание к выключению воспроизведения;
- ❑ нажатые кнопки заносятся в массив воспроизведения, последняя нажатая – в начало массива, массив сдвигаетя;
- ❑ отжатие кнопки приводит к удалению ее из массива воспроизведения;
- ❑ если отжата первая кнопка воспроизведения, происходит сдвиг массива и начинает воспроизводиться нота предудущей нажатой кнопки до ее отжатия.

Соответствие кнопок воспроизводимым нотам приведено в таблице:

К	К	К	К	К	К	К	К	К	К	К	К	К	К	К	К
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
1 октава								**2 октава**							
C	*D*	*E*	*F*	*G*	*A*	*B*	*H*	*c*	*d*	*e*	*f*	*g*	*a*	*b*	*h*
до	*ре*	*ми*	*фа*	*соль*	*ля*	*си b*	*си*	*до*	*ре*	*ми*	*фа*	*соль*	*ля*	*си b*	*си*

Создадим массив воспроизведения:

```
int play[16]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

и количество нажатых кнопок:

```
int counts=0;
```

При событии RRESSED (нажатие) заносим символ ноты в нулевую позицию массива play[] (массив непустой сначала делаем сдвиг элеметов массива) и запускаем tone() для воспроизведения соответствующей ноты.

При событии RELEASED (отжатие) удаляем символ ноты из массива play[] (и делаем сдвиг элеметов массива влево до удаленной ноты) и запускаем tone() для нулевого элемента play[] (или noTone(), если массив стал пустым).

```
void shiftAdd(int cnt) {
     for(int i=cnt;i>0;i--) {
        play[i]= play[i-1];
       }
}
void shiftDelete(int codeNote, int cnt) {
     for(int i=0;i<cnt;i++) {
        if(play[i]== codeNote)  {
           for(int j=i;j<cnt;j++) {
              play[j]= play[j+1];
           }
       }
}
```

И все содержимое скетча в листинге 23.1.

Листинг 23.1.

```
// подключение библиотеки
#include <Keypad.h>
// размеры клавиатуры 4x4
const byte ROWS = 4;
const byte COLS = 4;
// символы для клавиш
char keys[ROWS][COLS] = {
{'C','D','E','F'},
{'G','A','B','H'},
{'c','d','e','f'},
{'g','a','b','h'}
```

```
};
// контакты считывания
byte rowPins[ROWS] = {9, 8, 7, 6};
// контакты подачи 1
byte colPins[COLS] = {5, 4, 3, 2};
// создание объекта
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins,
                        ROWS, COLS );
// для формирования сообщений
String msg;
// массив воспроизведения:
int play[16]={16,16,16,16,16,16,16,16,16,16,16,16,16,16,16,16};
// количество одновременно нажатых кнопок:
int counts=0;
// список частот для нот:
int freq[]={261,293,329,349,392,440,466,494,
            523,587,659,698,784,880,932,988,0};
// пин подключения динамика
int pinSpeaker=10;

void setup() {
    // запуск последовательного порта
    Serial.begin(9600);
    msg = "";
    // длинное нажатие - 5 сек
    kpd.setHoldTime(5000);
}

void loop() {
    if (kpd.getKeys())
    {
        // сканирование массива состояний кнопок
        for (int i=0; i<LIST_MAX; i++)
        {
            // только с измененным статусом
            if ( kpd.key[i].stateChanged )
            {
              // получение состояния клавиш
              // IDLE, PRESSED, HOLD, or RELEASED
                switch (kpd.key[i].kstate) {
                    case PRESSED:
                    msg = " PRESSED.";
                    shiftAdd(counts);
                    play[0]= kpd.key[i].kcode;
                    counts++;
                    tone(pinSpeaker, freq[play[0]]);
                break;
                    case HOLD:
```

```
                    msg = " HOLD.";
                break;
                    case RELEASED:
                    msg = " RELEASED.";
                    shiftDelete(kpd.key[i].kcode,counts);
                    counts--;
                    if(counts>0)
                       {tone(pinSpeaker, freq[play[0]]);}
                    else
                       {noTone(pinSpeaker);}
                break;
                    case IDLE:
                    msg = " IDLE.";
                }
                // вывод состояния кнопок
                Serial.print("Key ");
                Serial.print(kpd.key[i].kchar);
                Serial.print(msg);
                Serial.print("   =");
                Serial.println(counts);
            }
        }
    }
}
void shiftAdd(int cnt) {
     for(int i=cnt;i>0;i--) {
        play[i]= play[i-1];
       }
}
void shiftDelete(int codeNote, int cnt) {
     for(int i=0;i<cnt;i++) {
        if(play[i]== codeNote)  {
           for(int j=i;j<cnt;j++) {
              play[j]= play[j+1];
           }
        }
     }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке
https://arduino-kit.ru/scetches/exp_23_01.zip.
Загружаем скетч на плату Arduino и выступаем в роли начинающего Arduino-пианиста!

Эксперимент 24.
ЖК дисплей на контроллере HD44780

В этом эксперименте мы познакомимся с ЖК индикаторами фирмы Winstar на контроллере HD44780

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ LCD Keypad shield – 1;
- ❑ Провода MM – 8.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Часто в проектах требуется получать некоторую визуальную информацию с электронного устройства. Если информация может быть представлена в символьном виде, то одним из вариантов ее отображения является использование символьных жидкокристаллических индикаторов (ЖКИ, или LCD). Наиболее популярны и доступны по цене символьные индикаторы, реализованные на базе контроллера Hitachi HD44780. У нас он расположен на LCD Keypad shield. Символьные ЖК-дисплеи Winstar имеют 16 выводов (рис. 24.1).

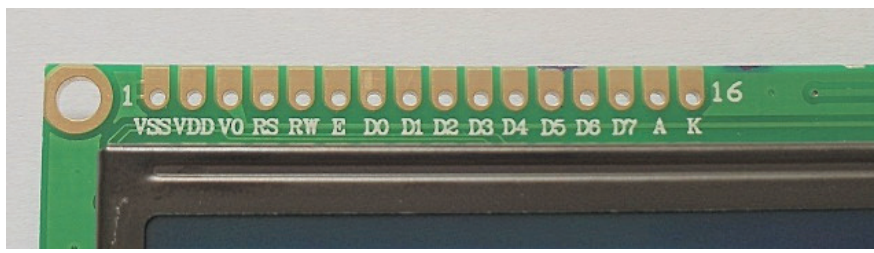


Рис. 24.1. Выводы ЖК-дисплея Winstar1602

Назначение контактов ЖК-дисплея и соответствие контактам на LCD Keypad shield представлено в таблице:

№ вывода	Название	Функция	Контакт на LCD keypad shield
1	*Vss*	*Общий (GND)*	*GND*
2	*Vdd*	*Напряжение питания (3 или 5 В)*	*+5 В*
3	*Vo*	*Контрастность*	
4	*RS*	*Команды/Данные*	*8*
5	*R/W*	*Чтение/Запись*	
6	*E*	*Разрешение чтения/записи*	*9*
7	*DB0*	*Линия данных 0*	
8	*DB1*	*Линия данных 1*	
9	*DB2*	*Линия данных 2*	
10	*DB3*	*Линия данных 3*	
11	*DB4*	*Линия данных 4*	*4*
12	*DB5*	*Линия данных 5*	*5*
13	*DB6*	*Линия данных 6*	*6*
14	*DB7*	*Линия данных 7*	*7*
15	*A*	*Напряжение подсветки (+)*	
16	*K*	*Напряжение подсветки (–)*	

Подсоединяется LCD Keypad shield к плате Arduino, как обычный шилд (вставкой штыревых контактов шилда в соответствующие контакты платы Arduino), либо проводами, как показано на рис. 24.2.

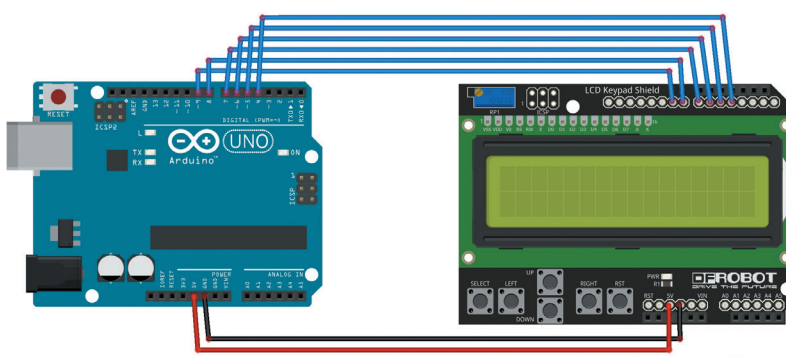


Рис. 24.2. Соединение платы Arduino и LCD Keypad shield

После подачи питания, настроим контрастность. Вращением потенциометра на в зависимости от того, выбран 4 или 8 битный режим обмена данными. Для сокращения требуемого числа выводов микроконтроллера можно работать в 4-битном режиме. В этом случае, на выводах DB4-DB7 сначала будет передаваться старшие четыре бита данных/команды, затем — младшие четыре бита. Выводы DB0-DB3 останутся незадействованными.

Теперь перейдем к программированию ЖКИ. Для программирования дисплея будем использовать стандартную библиотеку LiquidCrystal. Существует два основных вида библиотек Arduino: стандартные и дополнительные. Стандартные библиотеки уже присутствуют в Arduino IDE. Список установленных библиотек можно в пункте Эскиз → Include Library. В библиотеке LiquidCrystal есть несколько примеров Эскиз → Include Library → LiquidCrystal (рис. 24.3), которые помогут освоиться с программированием дисплея.

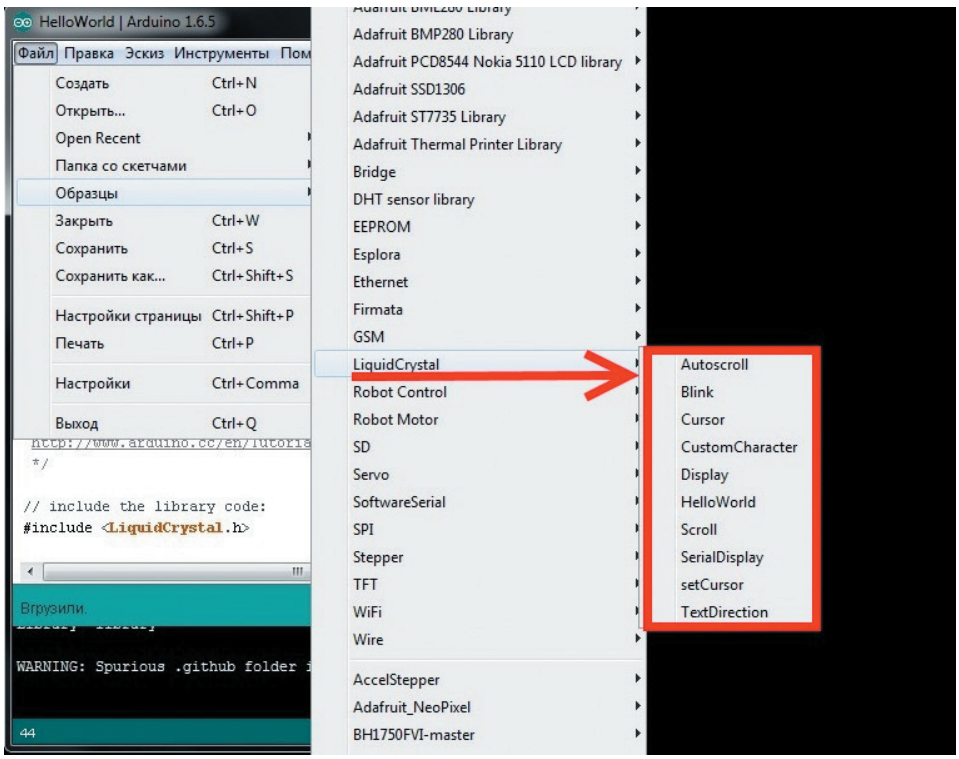


Рис. 24.3. Примеры библиотеки LiquidCrystal

Откроем пример Hello World и разберем его.

Первая строка – это подключение библиотеки LiquidCrystal.

```
#include <LiquidCrystal.h>
```

Как правило, код библиотеки оформляется в виде класса, поэтому для работы с библиотекой создаем экземпляр объекта LiquidCrystal:

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

Объект типа LiquidCrystal при создании имеет параметры:

- Пин подключения контакта RS (12);
- Пин подключения контакта E (11);
- Пин подключения контакта D4 (5);
- Пин подключения контакта D5 (4);
- Пин подключения контакта D6 (3);
- Пин подключения контакта D7 (2).

Для дисплея, подключенного по схеме на рис. 24.2 создание экземпляра объекта LiquidCrystal будет выглядеть следующим образом:

```
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
```

Разбираем код далее. В процедуре setup()

```
lcd.begin(16, 2);
```

Функция begin() определяет размерность (количество символов в ширину и высоту) дисплея, в нашем случае 16x2.

```
lcd.print("hello, world!");
```

Функция print() печатает текст на дисплее из текущей позиции курсора, изначально курсор находится в левом верхнем углу (0,0), поэтому текст печатается с начала первой строки, курсор перемещается при этом в позицию (13,0).

Далее в цикле loop():

```
lcd.setCursor(0, 1);
```

Функция print(col,str) устанавливает курсор в позицию col,str.

И далее функция print() печатает каждую секунду значение функции millis() – количество миллисекунд, прошедшее с начала выполнения программы.

Загрузим скетч на плату Arduino и убедимся в его работоспособности.

Далее в экспериментах мы часто будем использовать данный дисплей.

Эксперимент 25.
Создаем калькулятор на матричной клавиатуре

В этом эксперименте мы создадим простой калькулятор, используя матричную клавиатуру и дисплей на LCD Keypad shield в качестве экрана.

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ LCD Keypad shield – 1;
- ❑ Клавиатура матричная 4x4– 1;
- ❑ Провода MM – 16.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Наш калькулятор будет производить сложение, вычитание, умножение и деление двух чисел. Схема соединений показана на рис. 25.1.

Соответствие кнопок клавиатуры цифрам и действиям представлено в таблице:

K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
1	*2*	*3*	*+*	*4*	*5*	*6*	*-*	*7*	*8*	*9*	***	*=*		*D*	*/*

(K14 – пробел, соответствует клавище забоя (удаления предыдущего символа)).

Переходим к написанию скетча. Нам желательно видеть изображение курсора, указывающего место занесения следующего символа:

```
lcd.cursor();
```

Все получаемые символы будем заносить в буфер:

```
char buf[32];
```

При определении нажатия кнопки клавиатуры, отправляем код нажатой кнопки в процедуру addkey(). Процедура выполняет следующее:

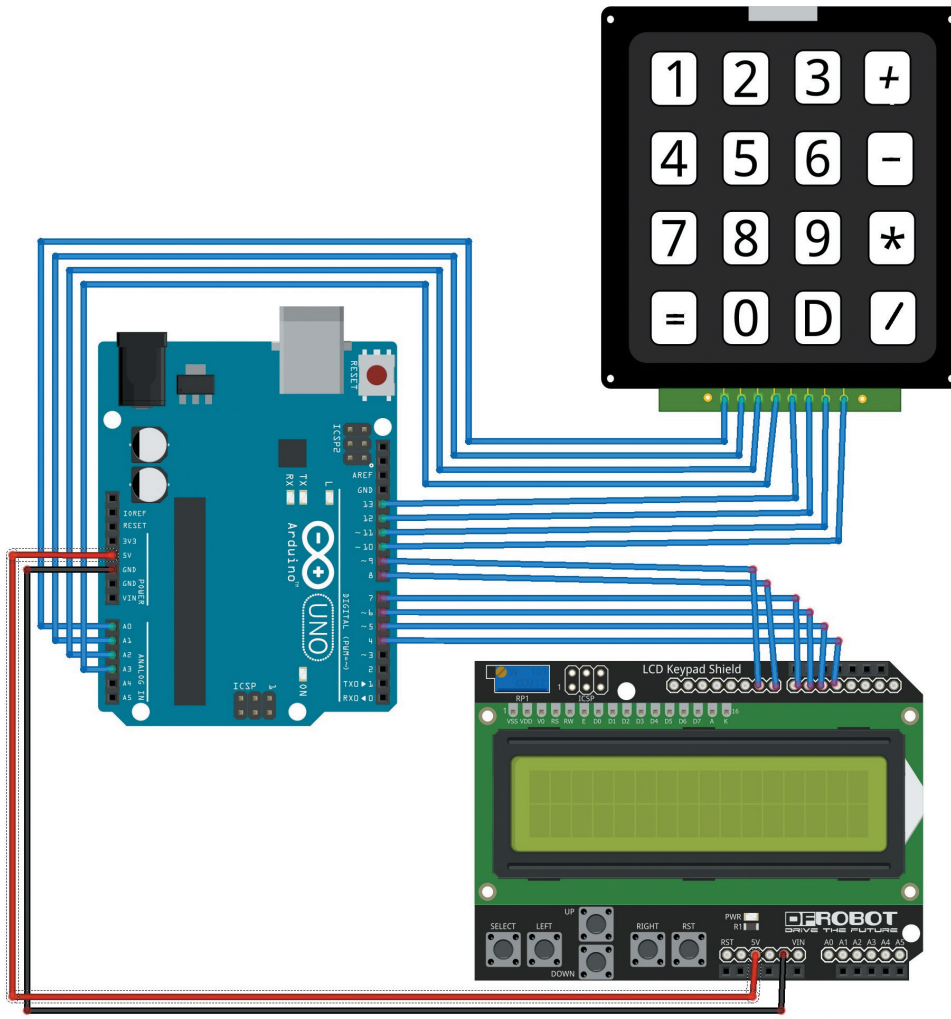


Рис. 25.1. Схема соединений для калькулятора

- при получении цифр добавляет код в буфер и устанавливает endkey=1;
- при получении клавиш ‘+’, ‘-’, ‘*’, ‘/’ добавляет код в буфер, если endkey=1, и устанавливает endkey=2;
- при получении ‘ ‘ удаляет предыдущий символ и устанавливает endkey=1, если последний элемент буфера цифра, и endkey=2, если последний элемент буфера действие;
- при получении ‘=’ вызывается getsumma() для вычисления суммы и устанавливает endkey=3 (любое следующее нажатии кнопки приводит к очищению буфера и экрана).

Функция getsumma() парсит буфер, получает первое число, второе, действие и производит математическую операцию с выводом результата на экран.

И все содержимое скетча в листинге 25.1.

Листинг 25.1.

```
// подключение библиотек
#include <LiquidCrystal.h>
#include <Keypad.h>

LiquidCrystal lcd(8,9,4,5,6,7);
const char keys[4][4]={{'1','2','3','+'},
                       {'4','5','6','-'},
                       {'7','8','9','*'},
                       {'=','0',' ','/'}
};
byte rows[] = {A0, A1, A2, A3 };
byte cols[] = {13, 12, 11, 10};
Keypad keypad1 = Keypad( makeKeymap(keys), rows, cols, 4, 4);

// позиция курсора
int pos=0;
int endkey=0;  // 1 - цифра, 2 - действиеб 3 - результат
// буфер для ввода
char buf[32];

void setup()
{
  lcd.begin(16,2);
  // вывод заставки
  lcd.cursor();
  lcd.setCursor(3,0);
  lcd.print("Calculator");
  lcd.setCursor(2,1);
  lcd.print("Arduino-kit.ru");
  delay(5000);
  lcd.clear();
  for(int i=0;i<32;i++)
     buf[i]=0;
}


void loop() {
  char key = keypad1.getKey();
  if (key){
    // добавить в буфер
    addkey(key);
  }
```

```
}
// добавление цифр и действий в буфер
void addkey(char k) {
  switch(k) {
    case '+': if(endkey==1) {
                buf[pos]=k;tolcd(k);endkey=2;
              }
        break;
    case '-': if(endkey==1) {
                buf[pos]=k;tolcd(k);endkey=2;
              }
        break;
    case '*': if(endkey==1) {
                buf[pos]=k;tolcd(k);endkey=2;
              }
        break;
    case '/': if(endkey==1) {
                buf[pos]=k;tolcd(k);endkey=2;
              }
        break;
    case ' ': pos=max(0,pos-1);tolcd(k);
              pos=pos-1;lcd.setCursor(pos%15,pos/15);
              if(pos==0)
                 endkey=0;
              else if(buf[pos-1]>=0x30 && buf[pos-1]<=0x39)
                  endkey=1;
              else
                  endkey=2;
        break;
    case '=': if(endkey==1) {
                buf[pos]=k;tolcd(k);
                getsumma();
                endkey=3;
              }
        break;
    // 0-9
    default : if(endkey==3) {
                startover();pos=0;}
              buf[pos]=k;tolcd(k);endkey=1;
        break;
  }
}
// вывод на экран
void tolcd(char k) {
  lcd.setCursor(pos%15,pos/15);
  lcd.print(k);
  pos=pos+1;
}
// подсчет суммы
```

```
void getsumma() {
   String number1="";
   String number2="";
   char d;
   int i;
   int summa;
   // получить первое число
   for(i=0;i<pos;i++) {
     if(buf[i]>=0x30 && buf[i]<=0x39)
        number1+=buf[i];
     else
        break;
   }
   // действие
   d=buf[i];
   // получить второе число
   for(i=i+1;i<pos;i++) {
     if(buf[i]>=0x30 && buf[i]<=0x39)
        number2+=buf[i];
     else
        break;
   }
   switch(d) {
      case '+': summa=number1.toInt()+number2.toInt();
         break;
      case '-': summa=number1.toInt()-number2.toInt();
         break;
      case '*': summa=number1.toInt()*number2.toInt();
         break;
      case '/': summa=number1.toInt()/number2.toInt();
         break;
      default:
         break;
   }
   lcd.setCursor(pos%15,pos/15);
   lcd.print(summa);
}
// начать сначала - обнуление буфера
// и очищение экрана
void startover() {
   for(int i=0;i<=pos;i++) {
      buf[i]=0;
   }
   lcd.clear();
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_25_01.zip.**

Загружаем скетч на плату Arduino и пользуемся калькулятором!

Эксперимент 26.
Управляем движущимся символом на экране дисплея

В этом эксперименте мы нарисуем на экране дисплея свой символ и будем управлять его движением с помощью кнопок клавиатуры

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ LCD Keypad shield – 1;
- ❑ Провода MM – 16.

Переключатели на плате Arduino+WiFi установите следующим образом

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Подключим к плате Arduino и напишем крохотную игру. Будем перемещать символ на экране дисплея с помощью кнопок клавиатуры LCD Keypad shield. Подсоединяем LCD Keypad shield к плате Arduino, как обычный шилд (установкой штыревых контактов шилда в соответствующие контакты платы Arduino), либо соединяем проводами, как показано на рис. 26.1.

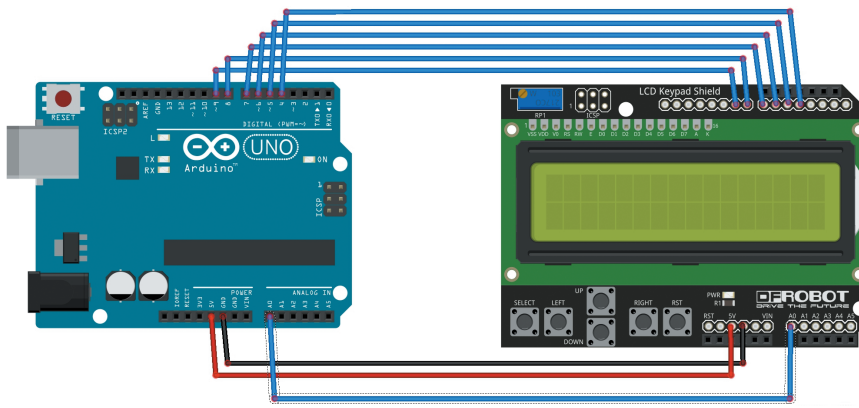


Рис. 26.1. Соединение платы Arduino и LCD Keypad shield

Приступим к написанию скетча. Функция createChar() библиотеки LiquidCrystal создает пользовательский символ для использования на жидкокристаллическом дисплее. Поддерживаются до восьми символов 5x8 пикселов (нумерация с 0 до 7). Создание каждого пользовательского символа определяется массивом из восьми байтов — один байт для каждой строки. Пять младших значащих битов каждого байта определяют пикселы в этой строке. Для вывода пользовательского символа на экран используется функция write() с номером символа в качестве параметра.

Синтаксис функции createChar():

```
lcd.createChar(num, data)
```

Параметры:

- lcd — переменная типа LiquidCrystal;
- num — номер создаваемого символа (0 to 7);
- data — данные символьных пикселов.

Для создания пользовательского символа можно использовать удобный ресурс **http://arduino.on.kg/lcdchargenerator** (рис. 26.2), где очень удобно не только рисовать символ, но и происходит генерации кода для Arduino

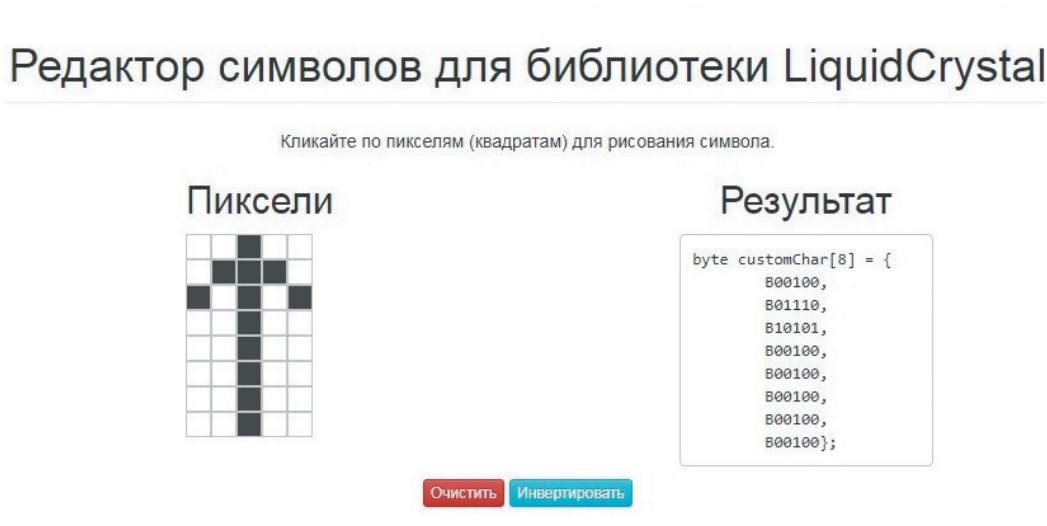


Рис. 26.2. Страницы для генерации пользовательских символов http://arduino.on.kg/lcdchargenerator

Сгенерируем 5 символов, изображающих движение влево, вправо, вверх, вниз и стояние на месте. Клавишами клавиатуры LCD Keypad shield будем запускать направление движения символа:

Кнопка	Символ на экране	Действие
SELECT	*char(0)*	*на месте*
LEFT	*char(1)*	*влево*
RIGHT	*char(2)*	*вправо*
UP	*char(3)*	*вверх*
DOWN	*char(4)*	*вниз*

Работа с аналоговыми кнопками LCD Keypad shield была рассмотрена в эксперименте 15. По нажатию кнопок изменяем значения переменных dirx, diry и номер символа, но изменение положения символа и смену символа производим только раз в секунду, используя millis() (см. эксперимент 9).

Сначала стираем симаол в предыдущей позиции (пишем в нее пробел):

```
lcd.setCursor(posy,posx);
lcd.write(' ');
```

Затем вычисляем следующую позицию для символа. Т.к. экран имеет конечные размеры, необходимо не выпускать символ за пределы экрана:

```
posy=max(min(15,posy+diry),0);
posx=max(min(1,posx+dirx),0);
А затем в новую позицию выводим новый символ:
lcd.setCursor(posy,posx);
lcd.write((uint8_t)simv);
Содержимое скетча показано в листинге 26.1.
```

Листинг 26.1.

```
// подключение библиотеки
#include <LiquidCrystal.h>
// создание экземпляра
LiquidCrystal lcd(8,9,4,5,6,7);

// массивы данных для пользовательских символов
byte customChar0[8] = {
    B00000,B00100,B01110,B11111,
    B01110,B00100,B00000,B00000};
byte customChar1[8] = {
    B00000,B00100,B01000,B11111,
    B01000,B00100,B00000,B00000};
byte customChar2[8] = {
    B00000,B00100,B00010,B11111,
    B00010,B00100,B00000,B00000};
byte customChar3[8] = {
```

```
  B00000,B00100,B01110,B10101,
    B00100,B00100,B00100,B00000};
byte customChar4[8] = {
    B00000,B00100,B00100,B00100,
    B10101,B01110,B00100,B00000};

// координаты x, y
int posx=0;
int posy=0;
// движение по x, y
int dirx=0;
int diry=0;
// текущий символ для отображнения
int simv=0;
// для задержки
unsigned long millis1=0;

void setup()
{
  // Создание новых символов новый символ - код 0 (0-7)
  lcd.createChar(0, customChar0);
  lcd.createChar(1, customChar1);
  lcd.createChar(2, customChar2);
  lcd.createChar(3, customChar3);
  lcd.createChar(4, customChar4);
  // Устанавливаем количество строк и столбцов.
  lcd.begin(16, 2);
  lcd.clear();
  // Печатаем символ с кодом 0
}

void loop() {
   // чтение данных A0
   int valA0 = analogRead(A0);
   // определение нажатия кнопки
   if(valA0<100) { // RIGHT
      setVar(1,0,2);
   }
   else if(valA0<200) { // UP
      setVar(0,-1,3);
   }
   else if(valA0<400) { // DOWN
      setVar(0,1,4);
   }
```

```
    else if(valA0<600) { // LEFT
       setVar(-1,0,1);
    }
    else if(valA0<800) { // SELECT
       setVar(0,0,0);
    }
    if(millis()-millis1>1000) {
       // удаление символа из предыдущей позиции (запись пробела)
       lcd.setCursor(posy,posx);
       lcd.write(' ');
       // вычисление следующей позиции
       posy=max(min(15,posy+diry),0);
       posx=max(min(1,posx+dirx),0);
       // вывод символа в следующую позицию
       lcd.setCursor(posy,posx);
       lcd.write((uint8_t)simv);
       // пауза (скорость перемещения)
       millis1=millis();
    }
}
// изменение переменных
// dirx, diry, simv
void setVar(int dy, int dx, int s) {
  dirx=dx;
  diry=dy;
  simv=s;
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_26_01.zip.**

Загружаем скетч на плату Arduino и управляем движением символа с помощью кнопок!

Эксперимент 27.
4-х разрядная светодиодная матрица

В этом эксперименте мы рассмотрим светодиодные матрицы и научимся выводить на них графические элементы

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ 4-х разрадная светодиодная матрица – 1;
- ❑ Провода MM – 2.
- ❑ Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Познакомимся с еще одним устройством, которое можно использовать для визуализации проектов Arduino. В экспериментах 4, 10 мы рассматривали шкалу и семисегментный индикатор, они состоят из отдельных светодиодов, соединенных вместе.

Если разместить светодиоды не в виде цифры или шкалы, а в виде сетки, то получится графический индикатор, на котором можно отобразить не только число, но и какое-то изображение. Такая сетка называется матричным индикатором, а в случае использования светодиодов — светодиодной матрицей. Разрешение матричного индикатора — это количество точек по горизонтали и вертикали. Самые распространенные индикаторы имеют разрешение 8×8 точек. Схема соединений светодиодов в матрице показана на рис. 27.1.

Для создания изображения на такой матрице с использованием динамической индикации необходимо задействовать 16 контактов Arduino! К счастью, разработаны специализированные микросхемы для управления индикаторами, например MAX721, которая позволяет управлять матрицей по 3 проводам. Если требуется светодиодная матрица с большим разрешением, то ее составляют из нескольких 8×8 индикаторов.

Схема подключения показана на рис. 27.2. Для питания 4-х разрадной светодиодной матрицы используем внешний источник питания 5В. Используем батарейный отсек на 2 батареи 18650, и понижаем его до 5В на понижающем преобразователе.

При программировании будем использовать библиотеку Max72xxPanel, которую можно скачать на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/arduino-Max72xxPanel-master.zip.** Дополнительно необходимо установить

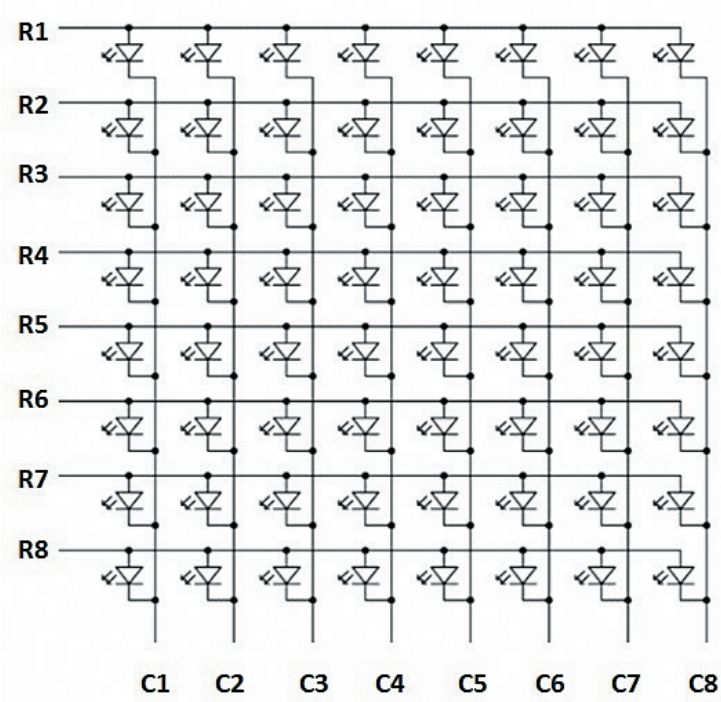


Рис. 27.1. Схема соединения светодиодов в матрице 8x8

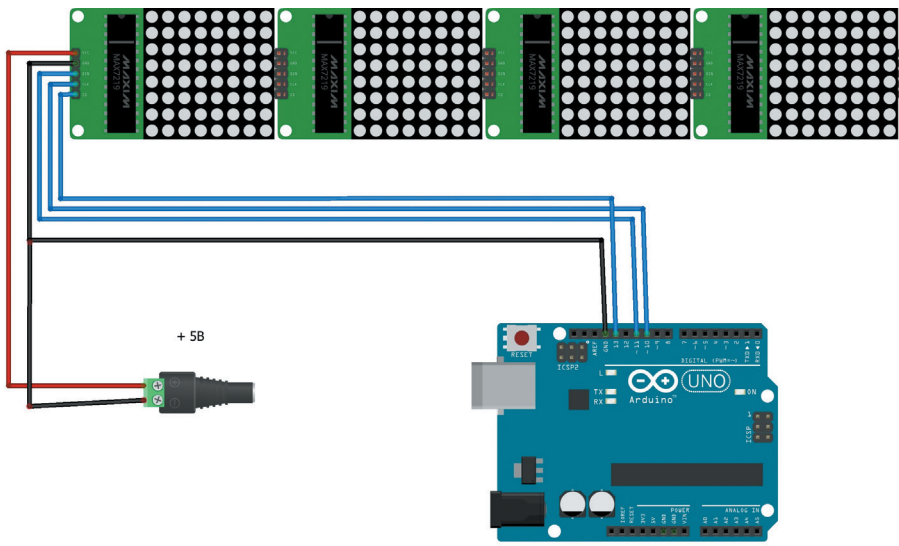


Рис. 27.2. Схема подключения матрицы 4x8x8 к плате Arduino

и библиотеку Adafruit-GFX-Library, необходимую для вывода графических примитивов на дисплеи.

В листинге 27.1. предствлен скетч вывода точки и гашения точки на матрицу в рандомные позиции.

Получаем рандомные позиции координат и зажигаем

```
matrix.drawPixel(x, y, HIGH);
Или гасим светодиод в матрице
matrix.drawPixel(x, y, LOW);
```

После включения и выключения пикселей с помощью функции drawPixel(), необходимо вызвать функцию write().

Листинг 27.1.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

// пин CS
int pinCS = 10;
// количество матриц по-горизонтали
int numberOfHorizontal = 4;
// количество матриц по-вертикали
int numberOfVertical = 1;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal,
                                   numberOfVertical);
// координаты x,y
int x, y;

void setup() {
    // яркость от 0 до 15
    matrix.setIntensity(8);
}

void loop() {
    x=random(0,32);
    y=random(0,8);
    // зажигаем пиксель
    matrix.drawPixel(x, y, HIGH);
    // вывод всех пикселей на матрицы
    matrix.write(); y
    delay(10);
    x=random(0,32);
    y=random(0,8);
    // гасим пиксель
    matrix.drawPixel(x, y, LOW);
    // вывод всех пикселей на матрицы
    matrix.write();
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_27_01.zip.**

Загружаем скетч на плату и наблюдаем за меняющимся заполнением экрана.

Эксперимент 28.
Вывод спрайтов и символов на 4-х разрядную светодиодную матрицу

В этом эксперименте мы рассмотрим вывод графических изображений (спрайтов) на 4-х разрядную светодиодную матрицу

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ 4-х разрадная светодиодная матрица – 1;
- ❑ Провода MM – 2.
- ❑ Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Схему соединений берем из эксперимента 27 (рис. 27.2). В эксперименте 27 мы зажигали и гасили пикселы на экране светодиодной матрицы. Попробуем вывести на экран маленькую картинку – спрайт. Для создания изображения спрайта будем использовать массив из восьми байт. Каждый байт массива будет отвечать за строку матрицы, а каждый бит в байте за точку в строке.

```
const byte sprite1[8] = {
    0B00111100,
    0B01000010,
    0B10100101,
    0B10000001,
    0B10100101,
    0B10011001,
    0B01000010,
    0B00111100
};
```

Загружаем на плату Arduino скетч из листинга 28.1 (попиксельный вывод картинки из массива).

Листинг 28.1.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>
// пин CS
int pinCS = 10;
// количество матриц по-горизонтали
int numberOfHorizontal = 4;
// количество матриц по-вертикали
int numberOfVertical = 1;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal, number-
OfVertical);
// изображение
const byte sprite1[8] = {
    0b00111100,
    0b01000010,
    0b10100101,
    0b10000001,
    0b10100101,
    0b10011001,
    0b01000010,
    0b00111100
};

void setup() {
    // яркость от 0 до 15
    matrix.setIntensity(6);
    // очистка экрана
    matrix.fillScreen(LOW);
    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, sprite1[y] & (1<<x));
        }
    }
    // вывод всех пикселей на матрицу
    matrix.write();
}

void loop() {;
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_28_01.zip.

После загрузки скетча на экране видим изображение, но оно повернуто на 90 градусов! А причина в том, что одноразрядные матрицы в данной сборке стоят неправильно. Смотрим рис. 28.1.

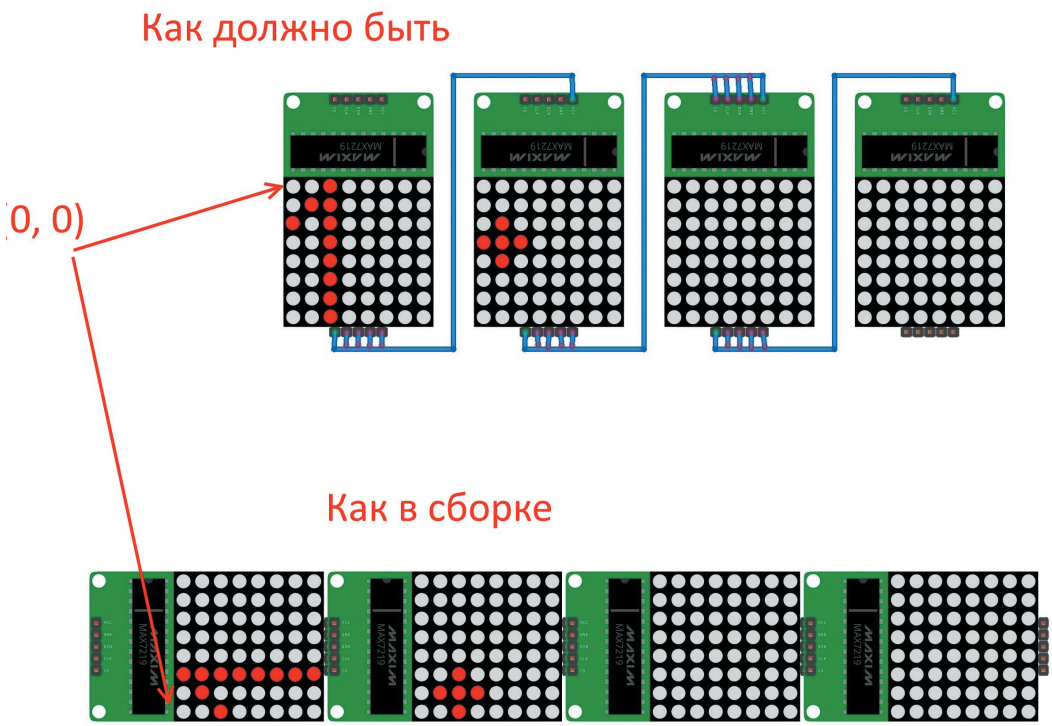


Рис. 28.1. Схема соединения матриц

В библиотеке Max72xxPanel есть функция setRotation(), которая задает ориентацию изображения на матрице.

```
matrix.setRotation(0,1);
```

первый параметр – это индекс матрицы, второй – количество поворотов на 90 градусов.

Так же можно выводить на матрицу и любой другой символ, например, букву или цифру. Для этого необходимо создать изображения для необхпдимых символов. К счастью, в библиотеке Adafruit-GFX-Library помимо функций для работы с графикой и текстом, имеется и база латинских букв в верхнем и нижнем регистрах, а также все знаки препинания и прочие служебные символы. Символы имеют размер 5x8. Отобразить символ на матрице можно с помощью функции drawChar():

drawChar(x, y, символ, цвет, фон, размер);

В листинге 28.2 приведен скетч вывода текста на нашу 4-разрядную светодиодную матрицу.

Листинг 28.2.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>
// пин CS
int pinCS = 10;
// количество матриц по-горизонтали
int numberOfHorizontal = 4;
// количество матриц по-вертикали
int numberOfVertical = 1;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal,
                                   numberOfVertical);
// текст для вывода
String text = "Ok95!";

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

void loop() {
    // очистить экран
    matrix.fillScreen(LOW);
    for ( int i = 0 ; i < text.length(); i++ ) {
        // поворот на 90 градусов
        matrix.setRotation( i, 1 );
        // вывод символов
        matrix.drawChar(i*6, 0, text[i], HIGH, LOW, 1);
        // вывод на матрицу
        matrix.write();
        delay(1000);
    }
    delay(5000);
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_28_02.zip.**
После загрузки скетча на экране видим текст!

Эксперимент 29.
«Бегущая строка» на 4-х разрядной светодиодной матрице

В этом эксперименте мы создадим "бегущую строку" на 4-х разрядной светодиодной матрице

В эксперименте мы будем использовать следующие компоненты:

- Плата Arduino UNO – 1;
- Кабель USB – 1;
- Плата прототипирования – 1;
- 4-х разрадная светодиодная матрица – 1;
- Потенциометр – 1;
- Провода MM – 2;
- Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

В этом эксперименте создадим "бегущую строку" на 4-х разрядной светодиодной матрице. Схема соединений показана на рис. 29.1. Потенциометр будем использовать для регулирования скорости.

Приступаем к написанию скетча. Строку для вывода будем хранить в переменной text:

```
String text = "Arduino-KIT";
```

Размер матрицы по горизонтали 32 позиции. Длина "бегущей строки" равна длине строки умноженной на 6 (5 ширина символа + 1 интервал между символами).

Переменная offset – координата начала строки. Начиная с этой координаты, выводим символы на матрицу.

Крайнее левое положение:

```
offset=0-text.length()*6;
```

По достижении крайнего положения устанавливаем offset=32.

Количество матриц по-горизонтали берем с запасом, чтобы не происходило сбоев с выводом.

```
int numberOfHorizontal = 15;
```

Скорость движения регулируем потенциометром.

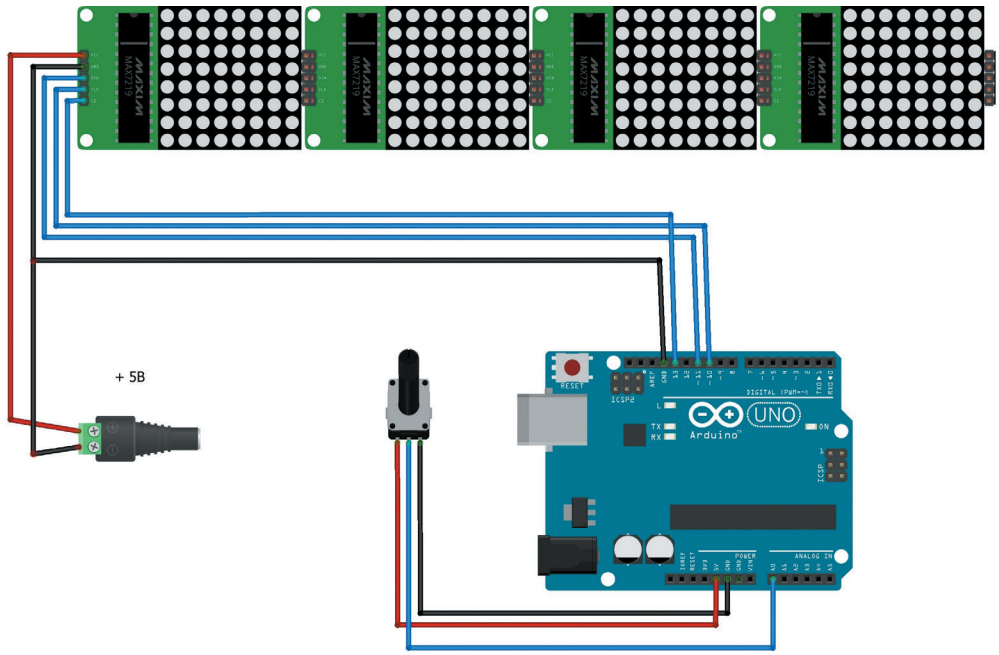


Рис. 29.1. Схема соединения для «бегущей строки» на 4-х разрядной светодиодной матрице

Содержимое скетча представлено в листинге 29.1.

Листинг 29.1.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>
// пин CS
int pinCS = 10;
// количество матриц по-горизонтали (берем с запасом!!!)
int numberOfHorizontal = 15;
// количество матриц по-вертикали
int numberOfVertical = 1;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal,
                      numberOfVertical);
// строка для вывода
String text = "Arduino-KIT 2019";
// текущее смещение от 0
int offset=32;
// максимальное значение скорости
```

```
int maxspeed1=100;
// минимальное значение скорости
int minspeed1=1000;

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

void loop() {
    // очистка экрана
    matrix.fillScreen(LOW);
    // вывод строки с позиции offset
    for ( int i = 0 ; i < text.length(); i++ ) {
        matrix.setRotation( i, 1 );
        if(i*6+offset>(-6) && i*6+offset<32) {
            matrix.drawChar(i*6+offset, 0, text[i], HIGH, LOW, 1);
        }
    }
    matrix.write();
    // задержка (скорость)
    int speed1=(analogRead(A0),0,1023,minspeed1,maxspeed1);
    delay(speed1);
    // изменение смещения
    offset=offset-1;
    // в начало - позиция 32
    if(offset+text.length()*6==0) {
       offset=32;
    }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_29_01.zip.**

Загружаем скетч на плату Arduino и наблюдаем “бегущую строку”, потенциометром регулируем скорость.

Эксперимент 30.
Русификация «бегущей строки» на 4-х разрядной светодиодной матрице

В этом эксперименте мы добавим русские буквы в "бегущую строку" на 4-х разрядной светодиодной матрице

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ 4-х разрадная светодиодная матрица – 1;
- ❑ Потенциометр – 1;
- ❑ Провода MM – 2;
- ❑ Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Схему соединений берем из эксперимента 29 (рис. 29.1).

В эксперименте 29 мы создали "бегущую строку" на 4-х разрядной светодиодной матрице. Однако, если в строку вставить русские буквы, в бегущей строке мы получим кракозябры. Это происходит по причине того, что в графической библиотеке Adafruit_GFX нет русского шрифта. Следовательно, его необходимо добавить. Нужно заменить в файле glcdfont.c из библиотеки Adafruit GFX определенные символы на русские в нужной кодировке.

Заменим файл glcdfont.c в библиотеке Adafruit-GFX файлом, скачанным с сайта Arduino-kit **(https://arduino-kit.ru/scetches/glcdfont.c).**

Однако и здесь есть проблема. Шрифт glcdfont.c рассчитан на однобайтную кодировку букв, а Arduino IDE использует для русских букв двухбайтовую UTF-8.

В русской кодировке UTF-8 прослеживается определенная закономерность, которая позволяет сделать преобразование из UTF-8 в однобайтовую русскую кодировку Windows-1251.

В листинге 30.1 показана функция utf8rus(), которая получает исходную строку, символы с кодами 0x00-0xBF пропускает без изменения в выходную строку, а в оставшихся кодах отбирает русские буквы и перекодирует их.

Листинг 30.1.

```
String utf8rus(String source)
{
  int i,k;
  String target;
  unsigned char n;
  char m[2] = { '0', '\0' };

  k = source.length(); i = 0;

  while (i < k) {
    n = source[i]; i++;
    if (n >= 0xC0) {
      switch (n) {
        case 0xD0: {
          n = source[i]; i++;
          if (n == 0x81) { n = 0xA8; break; }
          if (n >= 0x90 && n <= 0xBF) n = n + 0x2F;
          break;
        }
        case 0xD1: {
          n = source[i]; i++;
          if (n == 0x91) { n = 0xB8; break; }
          if (n >= 0x80 && n <= 0x8F) n = n + 0x6F;
          break;
        }
      }
    }
    m[0] = n; target = target + String(m);
  }
return target;
}
```

Теперь скетч для “бегущей строки” на 4-х разрядной светодиодной матрице примет вид, приведенный в листинге 30.2.

Листинг 30.2.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>
// пин CS
int pinCS = 10;
```

```
// количество матриц по-горизонтали
int numberOfHorizontal = 15;
// количество матриц по-вертикали
int numberOfVertical = 1;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal,
                          numberOfVertical);
// строка для вывода
String text = "Ардуино-КИТ 2019";
// текущее смещение от 0
int offset=32;
// максимальное значение скорости
int maxspeed1=100;
// минимальное значение скорости
int minspeed1=1000;

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

void loop() {
    String textnew=utf8rus(text);
    // очистка экрана
    matrix.fillScreen(LOW);
    // вывод строки с позиции offset
    for ( int i = 0 ; i < textnew.length(); i++ ) {
        matrix.setRotation( i, 1 );
        if(i*6+offset>(-6) && i*6+offset<32) {
            matrix.drawChar(i*6+offset, 0, textnew[i], HIGH, LOW, 1);
        }
    }
    matrix.write();
    // задержка (скорость)
    int speed1=(analogRead(A0),0,1023,minspeed1,maxspeed1);
    delay(speed1);
    // изменение смещения
    offset=offset-1;
    // в начало - позиция 32
    if(offset+text.length()*6==0)
       offset=32;
}

String utf8rus(String source)
{
  int i,k;
  String target;
  unsigned char n;
```

```
    char m[2] = { '0', '\0' };

    k = source.length(); i = 0;

    while (i < k) {
      n = source[i]; i++;

      if (n >= 0xC0) {
        switch (n) {
          case 0xD0: {
            n = source[i]; i++;
            if (n == 0x81) { n = 0xA8; break; }
            if (n >= 0x90 && n <= 0xBF) n = n + 0x2F;
            break;
          }
          case 0xD1: {
            n = source[i]; i++;
            if (n == 0x91) { n = 0xB8; break; }
            if (n >= 0x80 && n <= 0x8F) n = n + 0x6F;
            break;
          }
        }
      }
      m[0] = n; target = target + String(m);
    }
  return target;
  }
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_30_02.zip.**

Загружаем скетч на плату Arduino и наблюдаем русифицированную "бегущую строку", потенциометром регулируем скорость.

Эксперимент 31.
Загрузка по последовательному порту текста для «бегущей строки» на 4-х разрядной светодиодной матрице

В этом эксперименте мы будем изменять содержимое "бегущей строки", отправляя на плату Arduino данные по последовательному порту с компьютера

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ 4-х разрадная светодиодная матрица – 1;
- ❑ Потенциометр – 1;
- ❑ Провода MM – 2;
- ❑ Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

В этом эксперименте мы будем изменять содержимое "бегущей строки", отправляя данные на Arduino по последовательному порту. на 4-х разрядной светодиодной матрице. Схему соединений берем из эксперимента 29 (рис. 29.1).

Любая плата Arduino имеет, как минимум, один аппаратный последовательный интерфейс UART. Платы Arduino MEGA и Arduino DUE имеют по три порта. Плата Arduino UNO имеет один порт UART, сигналы которого подключены к выводам 0 (сигнал RX) и 1 (сигнал TX). Сигналы имеют логические уровни TTL (0…5 В). Через эти выводы (0 и 1) можно подключить к плате другое устройство имеющее интерфейс UART. Кроме функции связи с другими контроллерами порт UART платы Arduino UNO используется для загрузки в контроллер программы из компьютера. Для этого к этим же сигналам (RX и TX) подключены соответствующие

выводы микросхемы ATmega16U2 – преобразователя интерфейса USB/UART. Преобразователь интерфейса ATmega16U2 позволяет подключать плату Ардуино к компьютеру через USB порт. На компьютер устанавливается драйвер. Он создает на компьютере виртуальный COM порт. Через него и происходит обмен.

Для работы с аппаратными UART контроллерами в Ардуино существует встроенный класс Serial. Он предназначен для управления обменом данными через UART.

Приступим к написанию скетча.

Разрешаем работу порта UART на скорости обмена 9600 бод (бит в сек).

Serial.begin(9600);

В цикле loop() проверяем наличие данных в буфере последовательного порта и заносим их в накопительную строку. Приходящая строка должна заканчиваться символом '#'. При получении данного символа, полученную строку записываем в переменную для бегущей строки и устанавливаем смещение 32.

```
if (stringComplete) {
   textrus=inputString;
   Serial.println(textrus);
   offset=32;
   // очистить строку
   inputString = "";
   stringComplete = false;
}
```

Содержимое скетча представлено в листинге 31.1.

Листинг 31.1.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>
// пин CS
int pinCS = 10;
// количество матриц по-горизонтали
int numberOfHorizontal = 15;
// количество матриц по-вертикали
int numberOfVertical = 1;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal, number-
OfVertical);
// строка для вывода
String textrus = "Ардуино-КИТ 2019";
// текущее смещение от 0
int offset=32;
// максимальное значение скорости
int maxspeed1=100;
// минимальное значение скорости
```

```
int minspeed1=1000;
 // данные, пришедшие из последовательного порта
String inputString = "";
// строка пришла
boolean stringComplete = false;

void setup() {
    // яркость от 0 до 15
    matrix.setIntensity(7);
    // запуск последовательного порта
    Serial.begin(9600);
   // резервирование 30 bytes для inputString:
   inputString.reserve(30);
   Serial.println(utf8rus(textrus));
}

void loop() {
    // проверка прихода строки из последовательного порта
    if (stringComplete) {
       textrus=inputString;
       Serial.println(textrus);
       offset=32;
       // очистить строку
       inputString = "";
       stringComplete = false;
    }

    String text=utf8rus(textrus);
    // очистка экрана
    matrix.fillScreen(LOW);
    // вывод строки с позиции offset
    for ( int i = 0 ; i < text.length(); i++ ) {
        matrix.setRotation( i, 1 );
        if(i*6+offset>(-6) && i*6+offset<32) {
            matrix.drawChar(i*6+offset, 0, text[i], HIGH, LOW, 1);
        }
    }
    matrix.write();
    // задержка (скорость)
    int speed1=(analogRead(A0),0,1023,minspeed1,maxspeed1);
    delay(speed1);
    // изменение смещения
    offset=offset-1;
    // в начало - позиция 32
    if(offset+text.length()*6==0)
       offset=32;
}

String utf8rus(String source)
{
  int i,k;
```

```
  String target;
  unsigned char n;
  char m[2] = { '0', '\0' };

  k = source.length(); i = 0;

  while (i < k) {
    n = source[i]; i++;

    if (n >= 0xC0) {
      switch (n) {
        case 0xD0: {
          n = source[i]; i++;
          if (n == 0x81) { n = 0xA8; break; }
          if (n >= 0x90 && n <= 0xBF) n = n + 0x2F;
          break;
        }
        case 0xD1: {
          n = source[i]; i++;
          if (n == 0x91) { n = 0xB8; break; }
          if (n >= 0x80 && n <= 0x8F) n = n + 0x6F;
          break;
        }
      }
    }
    m[0] = n; target = target + String(m);
  }
return target;
}
//
void serialEvent() {
  boolean flag1=false;

  while (Serial.available() && flag1==false) {
    // получить байт:
    char inChar = (char)Serial.read();
    if (inChar == '#') {
      stringComplete = true;
      flag1=true;
    }
    else   // добавление в строку
      inputString += inChar;
 }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_31_01.zip.

Загружаем скетч на плату Arduino, открываем монитор последовательного порта и отправляем строку (в конце символ '#') (см. рис. 31.1) и наблюдаем данный текст на "бегущей строке".

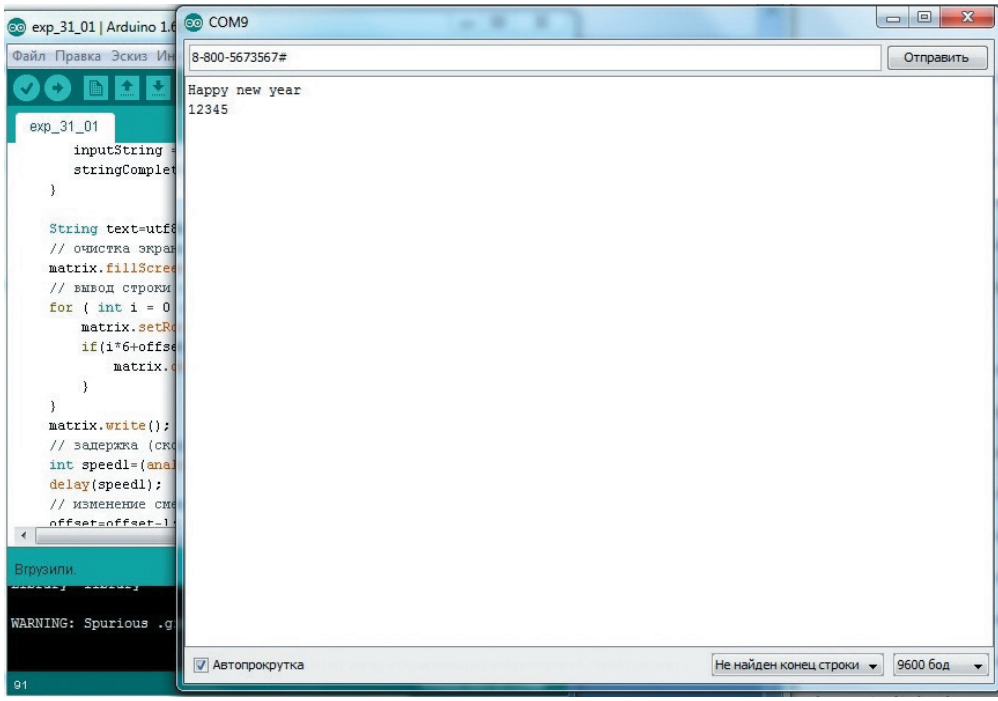


Рис. 31.1. Отправка данных «бегущей строки» из монитора последовательного порта Arduino IDE.

Эксперимент 32.
Подключаем двухкоординатный джойстик

В этом эксперименте мы подключим к плате Arduino аналоговый джойстик и визуализируем его показания на светодиодной матрице

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ 4-х разрадная светодиодная матрица – 1;
- ❑ Аналоговый джойстик – 1;
- ❑ Резистор 10 кОм – 1;
- ❑ Провода MM – 4;
- ❑ Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Джойстик – это удобное устройство для передачи информации от человека к микроконтроллеру или компьютеру. Видов джойстиков по количеству степеней свободы, принципу считывания показаний и используемым технологиям существует большое количество.

Аналоговый двухкоординатный джойстик, который присутствует в наборе, представляет собой ручку, закрепленную на шаровом шарнире с двумя взаимно перпендикулярными осями. При наклоне ручки, ось вращает подвижный контакт потенциометра, благодаря чему изменяется напряжение на его выходе. Также аналоговый джойстик имеет тактовую кнопку, которая срабатывает при вертикальном надавливании на ручку.

Схема подключения двухкоординатного джойстика к плате Arduino показана на рис. 32.2.

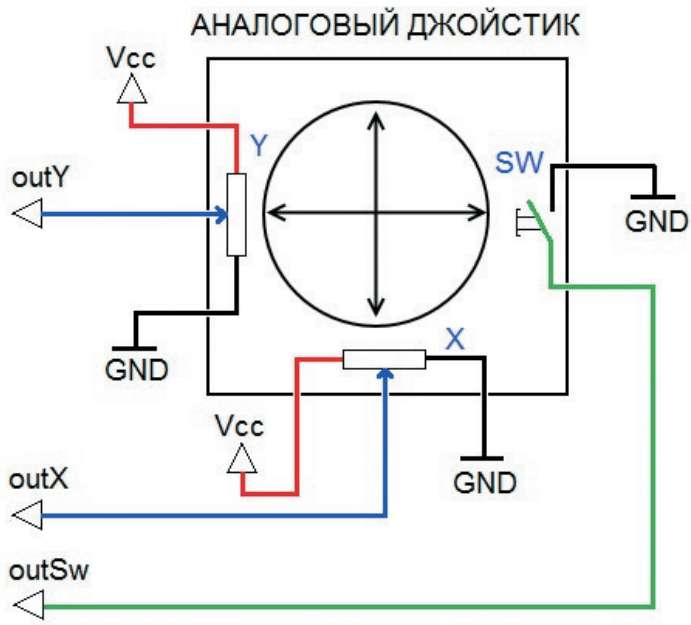


Рис. 32.1. Принципиальная схема аналогового джойстика

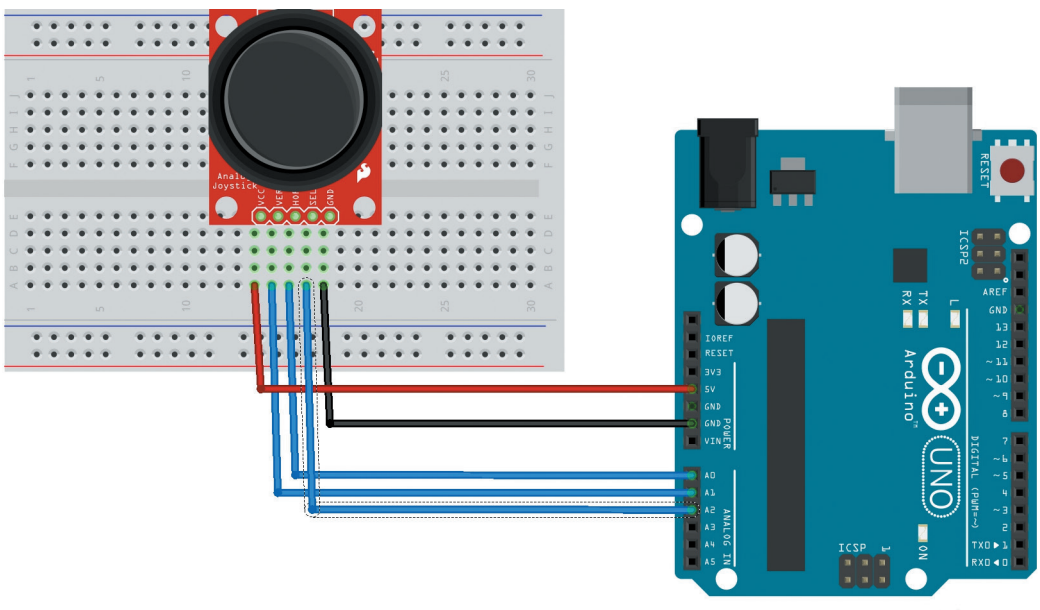


Рис. 32.2. Схема подключения двухкоординатного джойстика к плате Arduino

Содержимое скетча вывода показаний джойстика по двум осям x и y, и состояния кнопки sel () в монитор последовательного порта показано в листинге 32.1.

Листинг 32.1.

```
// пины подключения
int pinX = A0;
int pinY = A1;
int pinSel = A2;

void setup() {
    // запуск последовательного порта
    Serial.begin(9600);
    // пин подключения кнопки INPUT_PULLUP
    pinMode(pinSel, INPUT_PULLUP);
}

void loop() {
    // вывод данных
    Serial.print(«X=»);Serial.print(analogRead(pinX));
    Serial.print(«  Y=»);Serial.print(analogRead(pinY));
    Serial.print(«  Sel=»);Serial.println(digitalRead(pinSel));
    delay(200);
}
```

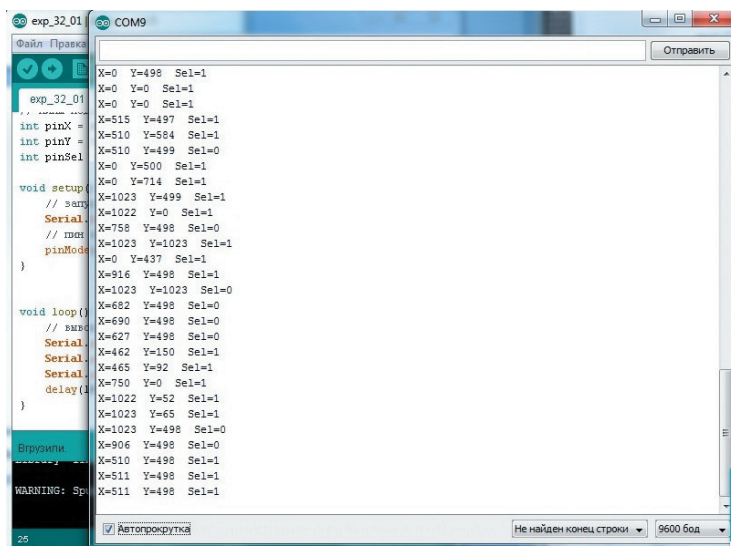


Рис. 32.3. Вывод данных с джойстика в монитор последовательного порта

Загружаем данный скетч на плату Arduino, открываем монитор последовательного порта и смотрим вывод данных при вращении ручки и нажатии кнопки джойстика (рис. 32.3). Теперь визуализируем данные по осям x и y джойстика на одной светодиодной матрице. Используем одну светодиодную матрицу. Данные по каждой из осей x и y (0-1023) масштабируем на матрицу 7x7 и строим прямоугольник согласно рис. 32.4.**Листинг 32.2.**

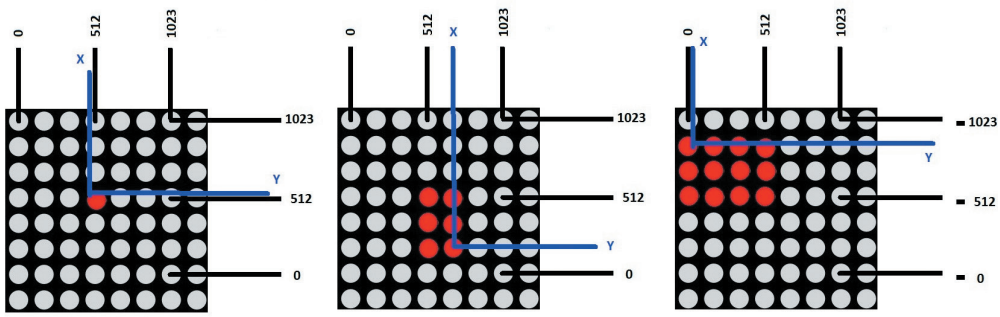


Рис. 32.4. Содержимое скетча показано в листинге 32.2

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 10;
// количество матриц по-горизонтали
int numberOfHorizontal = 1;
// количество матриц по-вертикали
int numberOfVertical = 1;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal,
                      numberOfVertical);
// пины подключения джойстика
int pinX = A0;
int pinY = A1;
// переменные
int x, y, x1, x2, y1, y2;

void setup() {
    // яркость от 0 до 15
    matrix.setIntensity(8);
}

void loop() {
```

```
    // получение данных с джойстика и масштабирование
    x=analogRead(pinX);
    y=analogRead(pinY);
    x=map(x,0,1023,0,7);
    y=map(y,0,1023,0,7);
    // очищение матрицы
    matrix.fillScreen(LOW);
    // вычисление от и до для for
    x1=min(3,x);
    x2=max(3,x);
    y1=min(3,y);
    y2=max(3,y);
    // зажигание пикселей
    for(int i1=x1;i1<=x2;i1++) {
      for(int i2=y1;i2<=y2;i2++) {
         matrix.drawPixel(i1, i2, HIGH);
      }
    }
    // вывод всех пикселей на матрицы
    matrix.write();
    delay(100);
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/ exp_32_02.zip.**

Загружаем скетч на плату Arduino, и наблюдаем на светодиодной матрице визуализацию положения осей x и y джойстика.

Эксперимент 33.
Игра «Змейка”. Управляем перемещением «змейки» на светодиодной матрице с помощью джойстика

В этом эксперименте мы приступим к написанию игры “змейка” и начнем с написания программного кода для управления движением “змейки” с помощью джойстика.

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ 4-х разрадная светодиодная матрица – 1;
- ❑ Аналоговый джойстик – 1;
- ❑ Динамик – 1;
- ❑ Резистор 10 кОм – 1;
- ❑ Провода MM – 4;
- ❑ Провода MF – 4.

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Переключатели на плате Arduino+WiFi установите следующим образом:

Приступим к написанию игры “Змейка” (известна еще как “Питон”).

Схема соединений представлена на рис. 33.1.

Используем 41 матрицу по горизонтали и 4 по вертикали.

Каждый элемент “змейки” – один светодиод матрицы. Данные о звеньях будем сохранять в массиве, каждый из элементов массива – координаты X и Y левого верхнего угла каждого звена:

```
// структура для описания координаты одного звена 8x8
struct Pos {
  int x;
  int y;
};
// для звеньев змейки, если не хватает max 40 – можно сделать
// больше!!!
Pos snake[40]={{4,4},{4,3},{4,2},{4,1},{4,0},
```

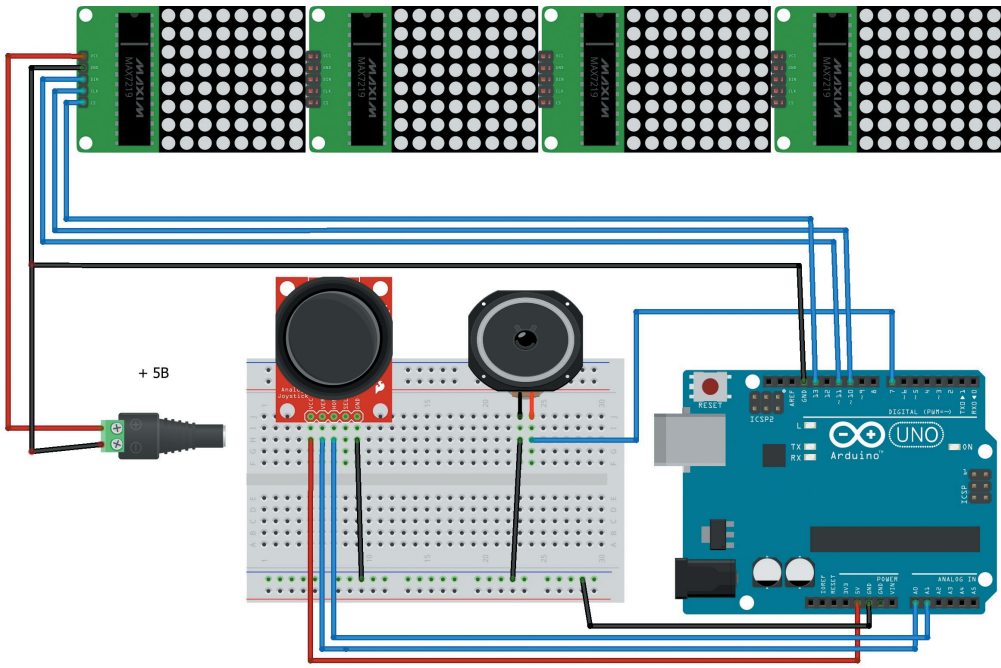


Рис. 33.1. Схема соединений для игры «Змейка».

```
{0,0},{0,0},{0,0},{0,0},{0,0},
{0,0},{0,0},{0,0},{0,0},{0,0},
{0,0},{0,0},{0,0},{0,0},{0,0},
{0,0},{0,0},{0,0},{0,0},{0,0},
{0,0},{0,0},{0,0},{0,0},{0,0},
{0,0},{0,0},{0,0},{0,0},{0,0},
{0,0},{0,0},{0,0},{0,0},{0,0}
};
```

Переменная – указатель на хвост змейки (размер змейки):

```
int offsetsnake=6;
```

Переменные для направления движения змейки

```
// перемещение по осям x и y
int dX=1; // -1, 0, 1
int dY=0; // -1, 0, 1
```

И переменные для скорости – времени, после которого происходит очередное изменение положения змейки:

```
// скорость
int speedsnake=500;
unsigned long millissnake=0;
```

Для изменения направления змейки будем использовать джойстик, но только четкие установки направления:

```
    if (analogRead(pinY) >950) // вправо
       {setdir(0,1);}
```

```
    else if (analogRead(pinY)<70) // влево
       {setdir(0,-1);}
    else if (analogRead(pinX)<70) // вверх
       {setdir(1,0);}
    else if (analogRead(pinX)>950) // вниз
       {setdir(-1,0);}
    else
       ;
….
// установить новое направление движения
void setdir(int x,int y) {
    int x1,y1;
    x1=x;y1=y;
    if((dX+x)==0 && abs(x)>0)
      {y1=y;x1=dX;}
    if((dY+y)==0 && abs(y)>0)
      {x1=x;y1=dY;}
    // установить
    dX=x1;dY=y1;
}
```

При переходе головы змейки за пределы поля, выводим ее с другой стороны.

```
if(snake[0].x==8)
  snake[0].x=0;
if(snake[0].x==-1)
  snake[0].x=7;
if(snake[0].y==32)
  snake[0].y=0;
if(snake[0].y==-1)
  snake[0].y=31;
```

Содержимое скетча показано в листинге 33.1.

Листинг 33.1.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 10;
// количество матриц по-горизонтали
int numberOfHorizontal = 1;
// количество матриц по-вертикали
int numberOfVertical = 4;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal, number-
OfVertical);
// пины подключения джойстика
int pinX = A0;
int pinY = A1;
// структура для описания координаты одного звена 8x8
struct Pos {
  int x;
  int y;
};
```

```
// массив всех звеньев змейки
Pos snake[40]={{4,4},{4,3},{4,2},{4,1},{4,0},
               {0,16},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0}
               };
int offsetsnake=5;

// перемещение по осям x и y и коэффициент (шаг)
int dX=0; // -1, 0, 1
int dY=1; // -1, 0, 1
// скорость
int speedsnake=500;
unsigned long millissnake=0;

// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 10;
// количество матриц по-горизонтали
int numberOfHorizontal = 1;
// количество матриц по-вертикали
int numberOfVertical = 4;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal, number-
OfVertical);
// пины подключения джойстика
int pinX = A0;
int pinY = A1;
// структура для описания координаты одного звена 8x8
struct Pos {
  int x;
  int y;
};
// массив всех звеньев змейки
Pos snake[40]={{4,4},{4,3},{4,2},{4,1},{4,0},
               {0,16},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0}
               };
int offsetsnake=5;
// массив для корма
Pos food[10]={{9,9},{9,9},{9,9},{9,9},{9,9},
```

```
                {9,9},{9,9},{9,9},{9,9},{9,9}
              };
int counterfood=0;


// перемещение по осям x и y и коэффициент (шаг)
int dX=0; // -1, 0, 1
int dY=1; // -1, 0, 1
// скорость
int speedsnake=300;
unsigned long millissnake=0;
// скорость
int speedfood=5000;
unsigned long millisfood=0;

boolean blink1=true;

void setup() {
    // яркость от 0 до 15
    matrix.setIntensity(8);
    // очищение матрицы
    matrix.fillScreen(LOW);
    // пин динамика
    pinMode(7,OUTPUT);
    Serial.begin(9600);

}

void loop(){
   // получение смещения по осям x и y
   if (analogRead(pinY) >950) // вправо
      {setdir(0,1);}
   else if (analogRead(pinY)<70) // влево
      {setdir(0,-1);}
   else if (analogRead(pinX)<70) // вверх
      {setdir(1,0);}
   else if (analogRead(pinX)>950) // вниз
      {setdir(-1,0);}
   else
      ;
   //// изменение положения змейки
   if(millis()-millissnake>=speedsnake) {
      // стереть предыдущее
      for(int i=0;i<offsetsnake;i++) {
         matrix.drawPixel(snake[i].x, snake[i].y, LOW);
      }
      //// новая позиция для змейки
      // остальные
      for(int i=offsetsnake-1;i>0;i--) {
         snake[i].x=snake[i-1].x;
         snake[i].y=snake[i-1].y;
      }
      // первая
```

```
    snake[0].x=snake[0].x+dX;
    snake[0].y=snake[0].y+dY;
    // выход за границу - появление с другой стороны
    if(snake[0].x==8)
      snake[0].x=0;
    if(snake[0].x==-1)
      snake[0].x=7;
    if(snake[0].y==32)
      snake[0].y=0;
    if(snake[0].y==-1)
      snake[0].y=31;
    // нарисовать новое
    for(int i=0;i<offsetsnake;i++) {
       matrix.drawPixel(snake[i].x, snake[i].y, HIGH);
    }
    // проверка съеден ли корм
    if(atefood(snake[0].x,snake[0].y)) {
       tone(7,494,200);
       delay(200);
       tone(7,349,200);
       offsetsnake++;
    }
    // вывести весь корм
    for(int i=0;i<10;i++) {
       if(food[i].x<8) {
          if(blink1)
             matrix.drawPixel(food[i].x, food[i].y, HIGH);
          else
             matrix.drawPixel(food[i].x, food[i].y, LOW);
 }
    }
    blink1=!blink1;
    // вывод всех пикселей на матрицы
    matrix.write();
    millissnake=millis();
}
//// появление корма
if(millis()-millisfood>=speedfood && counterfood<10) {
    // генерация случайных x и y
    int xf=random(0,8);
    int yf=random(0,32);
    // проверка отсутствия в списке корма
    for(int i=0;i<10;i++) {
       if(food[i].x==xf && food[i].y==yf)
          ;
       else if(food[i].x==9 && food[i].y==9){
          // не попадает на змейку
          if(nosnake(xf,yf)) {
             // добавить в массив
             addfood(xf,yf,i);
             // и звук
             tone(7,262,200);
             counterfood++;
```

```
                i=10;
              }
            }
          }
          millisfood=millis();
      }
  }
  // установить новое направление движения
  void setdir(int x,int y) {
      int x1,y1;
      x1=x;y1=y;
      if((dX+x)==0 && abs(x)>0)
        {y1=y;x1=dX;}
      if((dY+y)==0 && abs(y)>0)
        {x1=x;y1=dY;}
      // установить
      dX=x1;dY=y1;
  }
  // проверка непопадания корма на змейку
  boolean nosnake(int x,int y) {
      boolean ok=true;
      for(int i=0;i<40;i++) {
          if(snake[i].x==x && snake[i].y==y)
              ok=false;
      }
      return ok;
  }
  // добавить корм в массив
  void addfood(int x,int y, int pos) {
      food[pos].x=x;
      food[pos].y=y;
  }
  // проверка съела змейка корм?
  boolean atefood(int x,int y) {
      boolean ok=false;
      for(int i=0;i<10;i++) {
          if(food[i].x==x && food[i].y==y) {
              food[i].x=9;food[i].y=9;
              counterfood--;
              ok=true;
              Serial.println(i);
          }
      }
      return ok;
  }
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_33_01.zip.**

Загружаем скетч на плату Arduino, и с помощью джойстика управляем движением змейки на светодиодной матрице.

Эксперимент 34.
Игра «Змейка”. Добавляем корм для «змейки»

В этом эксперименте мы продолжим написание игры “змейка”, добавим генерацию корма и рост “змейки”

В эксперименте мы будем использовать компоненты и схему соединений из предыдущего эксперимента 33.

Продолжим написание игры “Змейка”.

Добавим генерирование корма для “змейки”. Нам необходим массив для хранения координат корма для максимального количества 10 (инициазизируем значениями за пределами экрана):

```
// массив для корма
Pos food[10]={{9,9},{9,9},{9,9},{9,9},{9,9},
              {9,9},{9,9},{9,9},{9,9},{9,9}
             };
int counterfood=0;
```

Через время speedfood генерируем корм и отображаем его на экране:

```
//// появление корма
if(millis()-millisfood>=speedfood && counterfood<10) {
   // генерация случайных x и y
   int xf=random(0,8);
   int yf=random(0,32);
   // проверка отсутствия в списке корма
   for(int i=0;i<10;i++) {
      if(food[i].x==xf && food[i].y==yf)
         ;
      else if(food[i].x==0 && food[i].y==0){
         // не попадает на змейку
         if(nosnake(xf,yf)) {
            // добавить в массив
            addfood(xf,yf,i);
            // и звук
            tone(7,262,200);
            counterfood++;
            i=10;
         }
      }
   }
   millisfood=millis();
}
```

Проверяем отсутствие сгенерированного значения координат в базе, в процедуре nosnake() не попадает ли корм на "змейку" и заносим данные в массив food:

```
// проверка непопадания корма на змейку
boolean nosnake(int x,int y) {
   boolean ok=true;
   for(int i=0;i<40;i++) {
      if(snake[i].x==x && snake[i].y==y)
         ok=false;
   }
   return ok;
}
// добавить корм в массив
void addfood(int x,int y, int pos) {
   food[pos].x=x;
   food[pos].y=y;
}
```

Вывод изображения "корма" производим в момент вывода изображения самой "змейки" и для "корма" делаем мигание:

```
if(millis()-millissnake>=speedsnake) {
   …….
   // вывести весь корм
   for(int i=0;i<10;i++) {
      if(food[i].x<8) {
         if(blink1)
            matrix.drawPixel(food[i].x, food[i].y, HIGH);
         else
            matrix.drawPixel(food[i].x, food[i].y, LOW);
      }
   }
   blink1=!blink1;
   // вывод всех пикселей на матрицы
   matrix.write();
   millissnake=millis();
}
```

В цикле после изменения положения "змейки" проверяем, не съела ли "змейка" корм – совпадение попадание головы на координаты всех позиций корма.

```
// проверка - съела змейка корм?
if(atefood(snake[0].x,snake[0].y)) {
   tone(7,494,200);
   delay(200);
   tone(7,349,200);
   offsetsnake++;
}
И сама функция atefood():
// проверка съела змейка корм?
boolean atefood(int x,int y) {
   boolean ok=false;
   for(int i=0;i<10;i++) {
      if(food[i].x==x && food[i].y==y) {
         food[i].x=9;food[i].y=9;
```

```
            counterfood--;
            ok=true;
            Serial.println(i);
          }
      }
      return ok;
   }
```

При генерации “корма” и съедании “корма” организуем звуковое оформление через подключенный к контакту 7 динамик.

Содержимое скетча показано в листинге 34.1.

Листинг 34.1.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 10;
// количество матриц по-горизонтали
int numberOfHorizontal = 1;
// количество матриц по-вертикали
int numberOfVertical = 4;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal, number-
OfVertical);
// пины подключения джойстика
int pinX = A0;
int pinY = A1;
// структура для описания координаты одного звена 8x8
struct Pos {
  int x;
  int y;
};
// массив всех звеньев змейки
Pos snake[40]={{4,4},{4,3},{4,2},{4,1},{4,0},
               {0,16},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0},
               {0,0},{0,0},{0,0},{0,0},{0,0}
               };
int offsetsnake=5;
// массив для корма
Pos food[10]={{9,9},{9,9},{9,9},{9,9},{9,9},
                {9,9},{9,9},{9,9},{9,9},{9,9}
               };
int counterfood=0;
```

```
// перемещение по осям x и y и коэффициент (шаг)
int dX=0; // -1, 0, 1
int dY=1; // -1, 0, 1
// скорость
int speedsnake=300;
unsigned long millissnake=0;
// скорость
int speedfood=5000;
unsigned long millisfood=0;

boolean blink1=true;

void setup() {
    // яркость от 0 до 15
    matrix.setIntensity(8);
    // очищение матрицы
    matrix.fillScreen(LOW);
    // пин динамика
    pinMode(7,OUTPUT);
    Serial.begin(9600);

}

void loop(){
   // получение смещения по осям x и y
   if (analogRead(pinY) >950) // вправо
      {setdir(0,1);}
   else if (analogRead(pinY)<70) // влево
      {setdir(0,-1);}
   else if (analogRead(pinX)<70) // вверх
      {setdir(1,0);}
   else if (analogRead(pinX)>950) // вниз
      {setdir(-1,0);}
   else
      ;
   //// изменение положения змейки
   if(millis()-millissnake>=speedsnake) {
      // стереть предыдущее
      for(int i=0;i<offsetsnake;i++) {
         matrix.drawPixel(snake[i].x, snake[i].y, LOW);
      }
      //// новая позиция для змейки
      // остальные
      for(int i=offsetsnake-1;i>0;i--) {
         snake[i].x=snake[i-1].x;
         snake[i].y=snake[i-1].y;
      }
      // первая
      snake[0].x=snake[0].x+dX;
      snake[0].y=snake[0].y+dY;
```

```
    // выход за границу - появление с другой стороны
    if(snake[0].x==8)
      snake[0].x=0;
    if(snake[0].x==-1)
      snake[0].x=7;
    if(snake[0].y==32)
      snake[0].y=0;
    if(snake[0].y==-1)
      snake[0].y=31;
    // нарисовать новое
    for(int i=0;i<offsetsnake;i++) {
       matrix.drawPixel(snake[i].x, snake[i].y, HIGH);
    }
    // проверка съеден ли корм
    if(atefood(snake[0].x,snake[0].y)) {
       tone(7,494,200);
       delay(200);
       tone(7,349,200);
       offsetsnake++;
    }
    // вывести весь корм
    for(int i=0;i<10;i++) {
       if(food[i].x<8) {
          if(blink1)
             matrix.drawPixel(food[i].x, food[i].y, HIGH);
          else
             matrix.drawPixel(food[i].x, food[i].y, LOW);
       }
    }
    blink1=!blink1;
    // вывод всех пикселей на матрицы
    matrix.write();
    millissnake=millis();
}
//// появление корма
if(millis()-millisfood>=speedfood && counterfood<10) {
    // генерация случайных x и y
    int xf=random(0,8);
    int yf=random(0,32);
    // проверка отсутствия в списке корма
    for(int i=0;i<10;i++) {
       if(food[i].x==xf && food[i].y==yf)
          ;
       else if(food[i].x==9 && food[i].y==9){
          // не попадает на змейку
          if(nosnake(xf,yf)) {
             // добавить в массив
             addfood(xf,yf,i);
             // и звук
             tone(7,262,200);
             counterfood++;
```

```
                i=10;
              }
            }
          }
          millisfood=millis();
      }
  }
  // установить новое направление движения
  void setdir(int x,int y) {
     int x1,y1;
     x1=x;y1=y;
     if((dX+x)==0 && abs(x)>0)
       {y1=y;x1=dX;}
     if((dY+y)==0 && abs(y)>0)
       {x1=x;y1=dY;}
     // установить
     dX=x1;dY=y1;
  }
  // проверка непопадания корма на змейку
  boolean nosnake(int x,int y) {
     boolean ok=true;
     for(int i=0;i<40;i++) {
        if(snake[i].x==x && snake[i].y==y)
           ok=false;
     }
     return ok;
  }
  // добавить корм в массив
  void addfood(int x,int y, int pos) {
     food[pos].x=x;
     food[pos].y=y;
  }
  // проверка съела змейка корм?
  boolean atefood(int x,int y) {
     boolean ok=false;
     for(int i=0;i<10;i++) {
        if(food[i].x==x && food[i].y==y) {
           food[i].x=9;food[i].y=9;
           counterfood--;
           ok=true;
           Serial.println(i);
        }
     }
     return ok;
  }
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_34_01.zip.**

Загружаем скетч на плату Arduino, и проверяем функционал. “Змейка” растет, съедая “корм”.

Остается доделать совсем чуть-чуть до полной готовности игры.

Эксперимент 35.
Игра «Змейка». Последние штрихи.

В этом эксперименте мы завершаем написание игры "змейка", добавим окончание игры и переход на следующий уровень

В эксперименте мы будем использовать компоненты и схему соединений из предыдущего эксперимента 35.

Продолжим и закончим написание игры "Змейка". Реализуем переход на следующий уровень и окончание игры.

Игра завершается при наезде "змейки" на себя:

```
// проверка не съела ли себя
boolean outsnake(int x,int y) {
   boolean ok=false;
   for(int i=1;i<40;i++) {
      if(x==snake[i].x && y==snake[i].y) {
         ok=true;
      }
   }
   return ok;
}
```

В этом случае необходимо закончить игру (надпись END) и установить начальные данные для игры сначала:

```
// надпись на экране
void header(String h,int t) {
   // очистить экран
   matrix.fillScreen(LOW);
   // надпись
   for ( int i = 0 ; i < h.length(); i++ ) {
       matrix.setRotation( i, 1 );
       matrix.drawChar(0, i*8, h[i], HIGH, LOW, 1);
   }
   matrix.write();
   for ( int i = 0 ; i < h.length(); i++ ) {
       matrix.setRotation( i, 0 );
       matrix.drawChar(0, i*8, ' ', HIGH, LOW, 1);
   }
   delay(t);
}
```

```
// очистить настройки
void clear() {
   // очистить экран
   matrix.fillScreen(LOW);
   // массив snake
   for(int i=0;i<10;i++) {
      food[i].x=9;
      food[i].y=9;
   }
   counterfood=0;
   // массив food
   for(int i=0;i<40;i++) {
      snake[i].x=0;
      snake[i].y=0;
   }
   snake[0].x=4;snake[0].y=4;
   snake[1].x=4;snake[1].y=3;
   snake[2].x=4;snake[2].y=2;
   snake[3].x=4;snake[3].y=1;
   snake[4].x=4;snake[4].y=0;
   offsetsnake=5;
   // направление
   dX=1;
   dY=0;
}
Переход на новый уровень:
// новый уровень?
if(offsetsnake>39) {
   level++;
   header("lev"+String(level),5000);
   clear();
   speedsnake=max(50,300-level*30);
   speedfood=max(1000,5000-level*1000);
}
Содержимое скетча показано в листинге 35.1.
```

Листинг 35.1.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 10;
// количество матриц по-горизонтали
int numberOfHorizontal = 1;
// количество матриц по-вертикали
int numberOfVertical = 4;
// создание объекта
```

```
  Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal, number-
OfVertical);
  // пины подключения джойстика
  int pinX = A0;
  int pinY = A1;
  // структура для описания координаты одного звена 8x8
  struct Pos {
    int x;
    int y;
  };
  // массив всех звеньев змейки
  Pos snake[40]={{4,4},{4,3},{4,2},{4,1},{4,0},
              {0,16},{0,0},{0,0},{0,0},{0,0},
              {0,0},{0,0},{0,0},{0,0},{0,0},
              {0,0},{0,0},{0,0},{0,0},{0,0},
              {0,0},{0,0},{0,0},{0,0},{0,0},
              {0,0},{0,0},{0,0},{0,0},{0,0},
              {0,0},{0,0},{0,0},{0,0},{0,0},
              {0,0},{0,0},{0,0},{0,0},{0,0}
              };
  int offsetsnake=5;
  // массив для корма
  Pos food[10]={{9,9},{9,9},{9,9},{9,9},{9,9},
                {9,9},{9,9},{9,9},{9,9},{9,9}
              };
  int counterfood=0;


  // перемещение по осям x и y и коэффициент (шаг)
  int dX=0; // -1, 0, 1
  int dY=1; // -1, 0, 1
  // скорость
  int speedsnake=300;
  unsigned long millissnake=0;
  // скорость
  int speedfood=5000;
  unsigned long millisfood=0;
  // уровень
  int level=0;

  boolean blink1=true;

  void setup() {
      // яркость от 0 до 15
      matrix.setIntensity(8);
      // очищение матрицы
      matrix.fillScreen(LOW);
      // пин динамика
```

```
    pinMode(7,OUTPUT);
    Serial.begin(9600);

}

void loop(){
   // получение смещения по осям x и y
   if (analogRead(pinY) >950) // вправо
      {setdir(0,1);}
   else if (analogRead(pinY)<70) // влево
      {setdir(0,-1);}
   else if (analogRead(pinX)<70) // вверх
      {setdir(1,0);}
   else if (analogRead(pinX)>950) // вниз
      {setdir(-1,0);}
   else
      ;
   //// изменение положения змейки
   if(millis()-millissnake>=speedsnake) {
      // стереть предыдущее
      for(int i=0;i<offsetsnake;i++) {
         matrix.drawPixel(snake[i].x, snake[i].y, LOW);
      }
      //// новая позиция для змейки
      // остальные
      for(int i=offsetsnake-1;i>0;i--) {
         snake[i].x=snake[i-1].x;
         snake[i].y=snake[i-1].y;
      }
      // первая
      snake[0].x=snake[0].x+dX;
      snake[0].y=snake[0].y+dY;
      // выход за границу - появление с другой стороны
      if(snake[0].x==8)
        snake[0].x=0;
      if(snake[0].x==-1)
        snake[0].x=7;
      if(snake[0].y==32)
        snake[0].y=0;
      if(snake[0].y==-1)
        snake[0].y=31;
      // нарисовать новое
      for(int i=0;i<offsetsnake;i++) {
         matrix.drawPixel(snake[i].x, snake[i].y, HIGH);
      }
      // проверка не съела ли себя
      if(outsnake(snake[0].x,snake[0].y)) {
         header("END",5000);
         clear();
         speedsnake=300;
         speedfood=5000;
         level=0;
     }
```

```
      // проверка съеден ли корм
      if(atefood(snake[0].x,snake[0].y)) {
         tone(7,494,200);
         delay(200);
         tone(7,349,200);
         offsetsnake++;
      }
      // вывести весь корм
      for(int i=0;i<10;i++) {
         if(food[i].x<8) {
            if(blink1)
               matrix.drawPixel(food[i].x, food[i].y, HIGH);
            else
               matrix.drawPixel(food[i].x, food[i].y, LOW);
         }
      }
      blink1=!blink1;
      // новый уровень?
      if(offsetsnake>39) {
         level++;
         header("lev"+String(level),5000);
         clear();
         speedsnake=max(50,300-level*30);
         speedfood=max(1000,5000-level*1000);
      }
      // вывод всех пикселей на матрицы
      matrix.write();
      millissnake=millis();
   }
   //// появление корма
   if(millis()-millisfood>=speedfood && counterfood<10) {
      // генерация случайных x и y
      int xf=random(0,8);
      int yf=random(0,32);
      // проверка отсутствия в списке корма
      for(int i=0;i<10;i++) {
         if(food[i].x==xf && food[i].y==yf)
            ;
         else if(food[i].x==9 && food[i].y==9){
            // не попадает на змейку
            if(nosnake(xf,yf)) {
               // добавить в массив
               addfood(xf,yf,i);
               // и звук
               tone(7,262,200);
               counterfood++;
               i=10;
            }
         }
      }
      millisfood=millis();
   }
}
// установить новое направление движения
```

```
void setdir(int x,int y) {
   int x1,y1;
   x1=x;y1=y;
   if((dX+x)==0 && abs(x)>0)
     {y1=y;x1=dX;}
   if((dY+y)==0 && abs(y)>0)
     {x1=x;y1=dY;}
   // установить
   dX=x1;dY=y1;
}
// проверка непопадания корма на змейку
boolean nosnake(int x,int y) {
   boolean ok=true;
   for(int i=0;i<40;i++) {
      if(snake[i].x==x && snake[i].y==y)
         ok=false;
   }
   return ok;
}
// добавить корм в массив
void addfood(int x,int y, int pos) {
   food[pos].x=x;
   food[pos].y=y;
}
// проверка съела змейка корм?
boolean atefood(int x,int y) {
   boolean ok=false;
   for(int i=0;i<10;i++) {
      if(food[i].x==x && food[i].y==y) {
         food[i].x=9;food[i].y=9;
         counterfood--;
         ok=true;
         Serial.println(i);
      }
   }
   return ok;
}
// проверка не съела ли себя
boolean outsnake(int x,int y) {
   boolean ok=false;
   for(int i=1;i<40;i++) {
      if(x==snake[i].x && y==snake[i].y) {
         ok=true;
      }
   }
   return ok;
}
// надпись на экране
void header(String h,int t) {
   // очистить экран
```

```
    matrix.fillScreen(LOW);
    // надпись
    for ( int i = 0 ; i < h.length(); i++ ) {
        matrix.setRotation( i, 1 );
        matrix.drawChar(0, i*8, h[i], HIGH, LOW, 1);
    }
    matrix.write();
    for ( int i = 0 ; i < h.length(); i++ ) {
        matrix.setRotation( i, 0 );
        matrix.drawChar(0, i*8, ' ', HIGH, LOW, 1);
    }
    delay(t);
}
// очистить настройки
void clear() {
    // очистить экран
    matrix.fillScreen(LOW);
    // массив snake
    for(int i=0;i<10;i++) {
       food[i].x=9;
       food[i].y=9;
    }
    counterfood=0;
    // массив food
    for(int i=0;i<40;i++) {
       snake[i].x=0;
       snake[i].y=0;
    }
    snake[0].x=4;snake[0].y=4;
    snake[1].x=4;snake[1].y=3;
    snake[2].x=4;snake[2].y=2;
    snake[3].x=4;snake[3].y=1;
    snake[4].x=4;snake[4].y=0;
    offsetsnake=5;
    // направление
    dX=0;
    dY=1;
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_35_01.zip.**

Загружаем скетч на плату Arduino, и проверяем функционал. Игра готова!!!

Эксперимент 36.
Индикатор влажности почвы на датчике FC-28

С этого эксперимента мы начинаем работать с датчиками, и создадим индикатор увлажненности почвы на датчике FC-28

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Датчик влажности почвы FC-28 – 1;
- ❑ 10-разрядная линейная светодиодная шкала – 1;
- ❑ Резистор 220 Ом – 10;
- ❑ Провода MM – 11;
- ❑ Провода MF – 3.
- ❑ Провода FF – 2.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Датчик или сенсор – это устройство, с помощью которого мы измеряем значение какого-либо технологического параметра. Датчики позволяют определять, что происходит во внешней среде, и действовать на основе этой информации. Датчики, наверное, можно назвать органами чувств системы. Любой датчик состоит из чувствительного элемента и преобразовательной системы, выполняющей преобразование входного воздействия любой физической величины в сигнал, удобный для дальнейшего использования. Самый простой тип датчиков – аналоговые датчики. Это первичные преобразователи. Такой тип датчиков применяется в системах непрерывного измерения и регулирования. Принцип действия таких датчиков состоит в том, что при изменении параметра происходит соответствующее изменение его выходного сигнала. Выходное напряжение может принимать значение от 0 В до напряжения питания. Хотя обычно рабочий диапазон напряжений более узкий.

Модуль влажности почвы предназначен для определения влажности земли, в которую он погружен. Он позволяет узнать о недостаточном или избыточном поливе ваших домашних или садовых растений. Модуль состоит из двух частей: контактного щупа и датчика на основе компаратора LM393, который выдает напряжение на выход D0 по принципу: влажная почва – низкий логический уровень, сухая почва – высокий логический уровень. Уровень определяется пороговым значением, которое можно регулировать с помощью потенциометра. На вывод A0 подается аналоговое значение, которое можно передавать в контроллер для дальнейшей обработки, анализа и принятия решений.

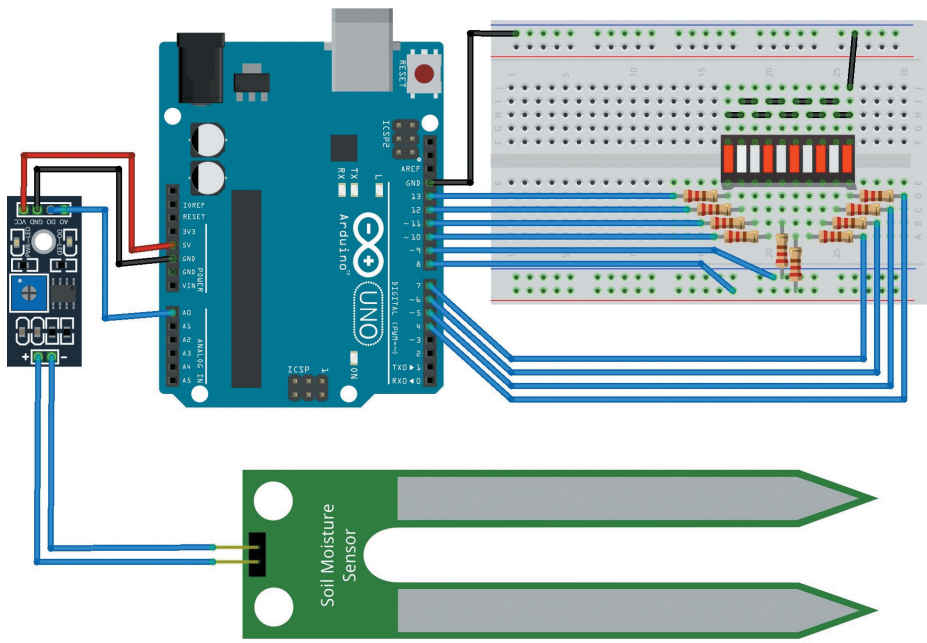


Рис. 36.1. Схема соединений для индикатора влажности почвы

Создадим индикатор влажности почвы. Схема соединений показана на рис. 36.1

Для индикации уровня влажности почвы будем использовать 10-разрядную линейную светодиодную шкалу. Значение переменных для полного полива (minvalue) и сильной сухости почвы (maxvalue) получим экспериментально. Большей сухости почвы соответствует большее значение аналогового сигнала. С помощью функции map масштабируем аналоговое значение датчика в значение нашего светодиодного индикатора. Чем больше влажность почвы, тем больше значение шкалы (количество зажженных светодиодов).

Содержимое скетча представлено в листинге 36.1.

Листинг 36.1.

```
// контакт подключения аналогового выхода датчика
int aPin=A0;
// контакты  подключения светодоодов индикации
int ledPins[]={13,12,11,10,9,8,7,6,5,4};
// переменная для сохранения значения датчика
int avalue=0;
// переменная количества светящихся светодиодов
int countled=10;
// значение полного полива
int minvalue=220;
// значение критической сухости
int maxvalue=600;

void setup()
  {
  // настройка выводов индикации светодиодов
  // в режим OUTPUT
  for(int i=0;i<10;i++)
    {
    pinMode(ledPins[i],OUTPUT);
    }
  }

void loop()
  {
  // получение значения с аналогового вывода датчика
  avalue=analogRead(aPin);
  // вывод значения в монитор последовательного порта Arduino
  Serial.print("avalue=");Serial.println(avalue);
  // масштабируем значение на 10 светодиодов
  countled=map(avalue,maxvalue,minvalue,0,10);
  // индикация уровня влажности
  for(int i=0;i<10;i++)
    {
    if(i<=countled)
       digitalWrite(ledPins[i],HIGH); //зажигаем светодиод
    else
       digitalWrite(ledPins[i],LOW);  // гасим светодиод
    }
  // пауза перед следующим получением значения 1000 мс
  delay(1000);
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_36_01.zip.**

Загружаем скетч на плату Arduino и получаем возможность издали определять влажность почвы вашего любимого цветка!

Эксперимент 37.
Звуковая сигнализация превышения уровня воды

В этом эксперименте мы создадим проект с использованием датчика уровня воды

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Датчик уровня воды – 1;
- ❑ Динамик – 1;
- ❑ Провода MM – 2;
- ❑ Провода MF – 3.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Датчики воды предназначены для определения уровня воды в различных емкостях, где недоступен визуальный контроль, с целью предупреждения перенаполнения емкости водой через критическую отметку. Конструкции датчиков уровня воды могут быть различными – поплавковые, погружные, врезные. Датчик воды, присутствующий в наборе – погружной. Чем больше погружение датчика в воду, тем меньше сопротивление между двумя соседними проводами.

В данном эксперименте создадим проект звуковой сигнализации затопления помещения. При погружении датчика в воду, сигнализация издает три вида звуковых сигналов (небольшое затопление, средний уровень, критический уровень), соответствуюший трем уровням воды. Для воспроизведения звуковых сигналов будем использовать один из динамиков.

Схема соединений показана на рис. 37.1.

Значение аналоговых сигналов на аналоговом входе Arduino для трех уровней погружения определяем экспериментальным путем:

> 400 – минимальное погружение;

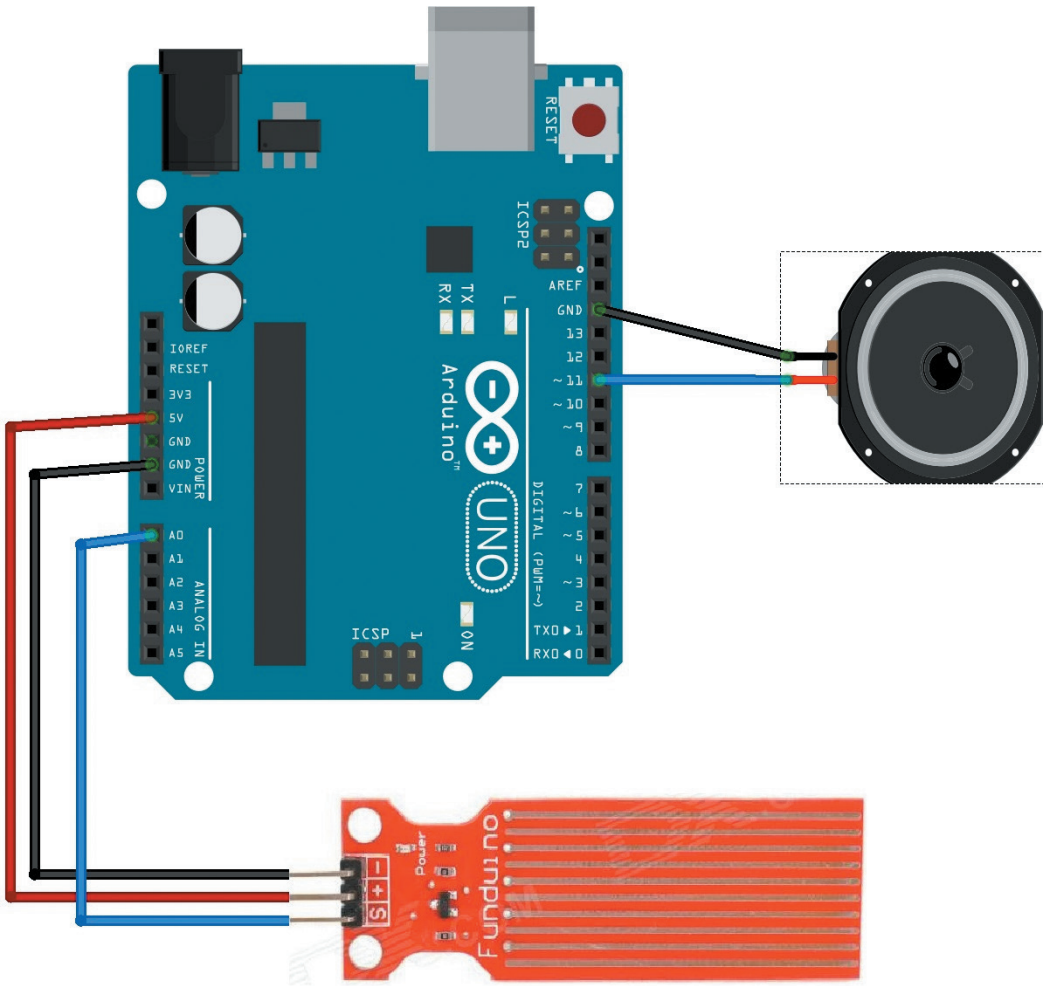


Рис. 37.1. Схема соединений звуковой сигнализации уровня воды

> 500 – средний уровень погружения;

> 600 – большое погружение.

Соответственно для каждого уровня погружения на динамике воспроизводится звуковой сигнал разной частоты:

минимальное погружение – 293 Гц (нота ре 1 октавы);

средний уровень погружения – 466 Гц (нота си-бимоль 1 октавы);

большое погружение – 587 Гц (нота ре 2 октавы).

При отсутствии погружения звуковой сигнал на динамике не воспроизводится.

Содержимое скетча пердставлено в листинге 37.1.

Листинг 37.1.

```
// контакт подключения аналогового выхода датчика
int aPin=A0;
// контакт  подключения вывода реле
int soundPin=11;
// частота звукового сигнала
int freq[3]={587,466,293};
// переменная для сохранения значения датчика
int avalue=0;
// значение уровней
int levels[3]={600,500,400};
// текущий уровень
int level=0;

void setup()
  {
  // инициализация последовательного порта
  Serial.begin(9600);
  // настройка выводов индикации светодиодов
  // в режим OUTPUT
  pinMode(soundPin,OUTPUT);
  }

void loop()
  {
  // получение значения с аналогового вывода датчика
  avalue=analogRead(aPin);
  // вывод значения в монитор последовательного порта Arduino
  Serial.print(«avalue=»);Serial.println(avalue);
  // вывод звука различной частоты для разных уровней погружения
  if(avalue>levels[0])
    tone(soundPin,freq[0],2000);
  else if(avalue>levels[1])
    tone(soundPin,freq[1],2000);
  else if(avalue>levels[2])
    tone(soundPin,freq[2],2000);
  else
    noTone(soundPin);
  // пауза перед следующим получением значения 1000 мс
  delay(1000);
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/ exp_37_01.zip.**

Загружаем скетч на плату Arduino и проверяем систему в действии!

Эксперимент 38.
Индикатор шума на датчике звука

В этом эксперименте создадим индикатор шума на датчике звука, данные будем выводить в виде графика

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Датчик звука – 1;
- ❑ Провода MM – 3.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Аналоговый датчик звука состоит из платы, на котором смонтированы выходы, усилитель звука, подстроечный резистор и электретный микрофон, чувствительный к звуку, приходящему во всех направлениях.

Схема соединений для подключения датчика звука к плате показана на рис. 38.1

Мы можем выводить аналоговые данные, получаемые с датчика звука в последовательный порт, но это не очень наглядно и совсем неудобно. Гораздо удобнее выводить данные в виде графика. И Arduino IDE предоставляет такую возможность (версии старше 1.6.6). Рассмотрим, как это делать.

Пишем скетч вывода аналоговых данных, получаемых с датчика звука в последовательный порт. Содержимое скетча показано в листинге 38.1.

Листинг 38.1.

```
// пин подключения датчика
int pinSensor=A0;

void setup()
{
  // подключение последовательного порта
  Serial.begin(9600);
}
```

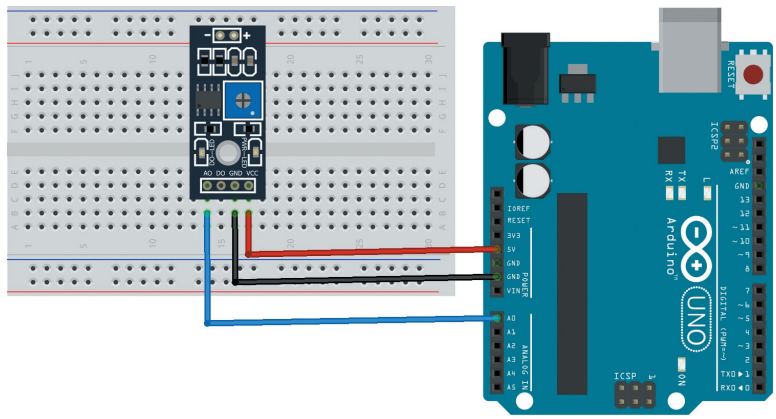


Рис. 38.1. Схема соединений для подключения датчика звука

```
void loop() {
  Serial.println(analogRead(A0));
  // небольшая пауза
  delay(5);
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/ exp_38_01.zip.**

После загрузки скетча в Arduino IDE выбираем пункт Инструменты →Плоттер по последовательному соединению (или нажимаем комбинацию клавиш Ctrl+Shift+L). На экране появляется график изменения в реальном времени данных с датчика звука (рис. 38.2).

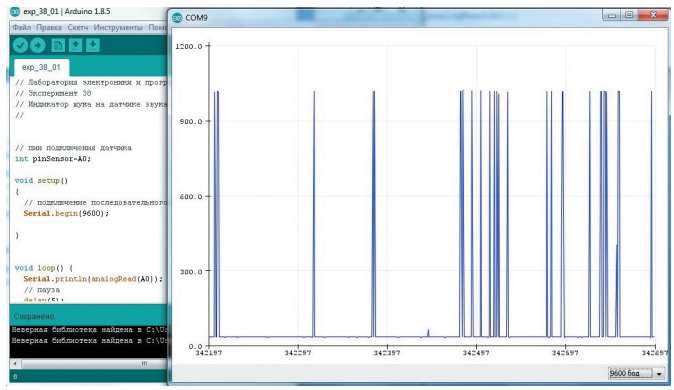


Рис. 38.2. График изменения в реальном времени данных с датчика звука

Подобный график можно строить для любых, изменяющихся во времени данных. Данные выводятся командой:
Serial.println(data);

Эксперимент 39.
Измерение влажности и температуры воздуха датчиком DHT11

В этом эксперименте будем получать с датчика DHT11 данные относительной влажности и температуры воздуха и выводить их на ЖК-дисплей

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Датчик DHT11 – 1;
- ❑ LCD Keypad shield – 1;
- ❑ Провода MM – 11.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

DHT11 — это цифровой датчик влажности и температуры, состоящий из термистора и емкостного датчика влажности. Также датчик содержит в себе АЦП для преобразования аналоговых значений влажности и температуры. Датчик DHT11 не обладают высоким быстродействием и точностью, но зато прост, недорог и отлично подходят для контроля влажности в помещении.

Данные с датчика будем выводить на ЖК-дисплей. Схема соединений показана на рис. 39.1.

Для работы с датчиком будем использовать Arduino-библиотеку DHT, которую можно скачать с сайта Arduino-kit по ссылке https://arduino-kit.ru/scetches/ DHT.zip. Получаем данные с датчика и выводим на дисплей LCD Keypad shield, а также отправляем в последовательный порт.

Содержимое скетча показано в листинге 39.1.

Листинг 39.1.

```
// подключение библиотек
#include <LiquidCrystal.h>
#include "DHT.h"
```

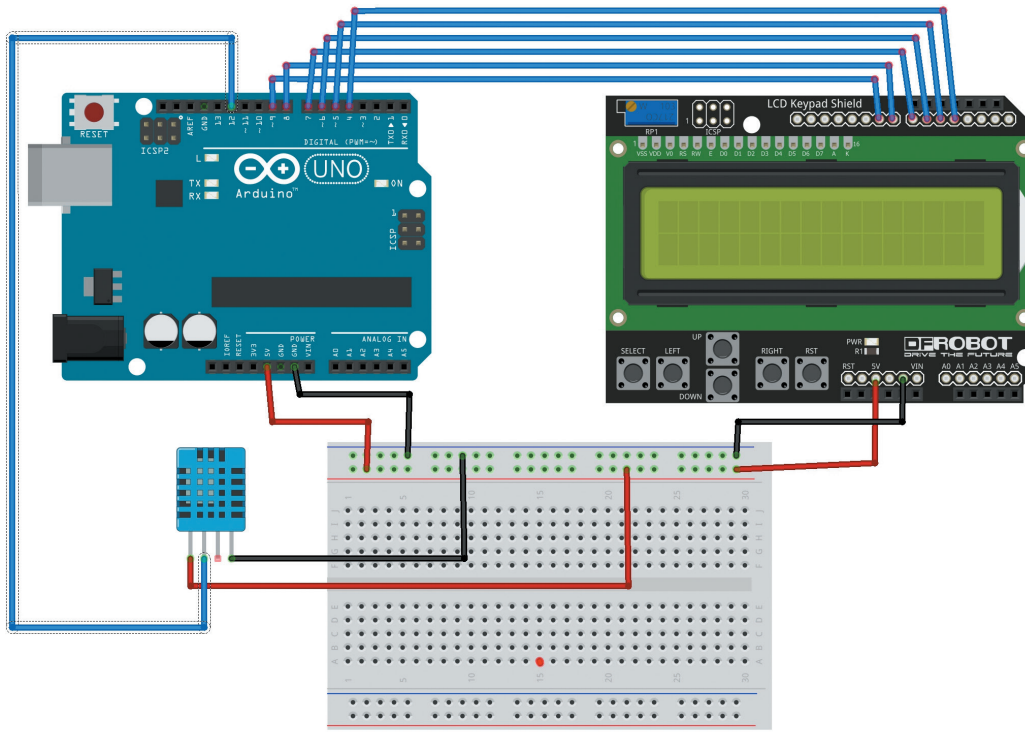


Рис. 39.1. Схема соединений для вывода данных с датчика DHT11 на дисплей

```
// пин для подключения датчика DHT
#define DHTPIN 12
// тип датчика DHT
#define DHTTYPE DHT11   // DHT 11

// создание экземпляра LiquidCrystal
LiquidCrystal lcd(8,9,4,5,6,7);
// создание экземпляра DHT
DHT dht(DHTPIN, DHTTYPE);



void setup()
{
  // подключение последовательного порта
  Serial.begin(9600);
  // инициализация дисплея
  lcd.begin(16,2);
  // очистить
  lcd.clear();
  // запуск датчика DHT
```

```
  dht.begin();
}

void loop() {
  // получение данных влажности
  float h = dht.readHumidity();
  // получение данных температуры
  float t = dht.readTemperature();
  // вывод на дисплей
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("H= ");
  lcd.print(h);lcd.print(" %");
  lcd.setCursor(0,1);
  lcd.print("T= ");
  lcd.print(t);lcd.print(" *C");
  // вывод в последовательный порт
  Serial.print("Humidity= ");
  Serial.print(h);Serial.print(" %");
  Serial.print("   Temperature= ");
  Serial.print(t);Serial.println(" *C");
  // не чаще 2 секунд
  delay(2000);
}
```

Скачать данный скетч можно на сайте на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/ exp_39_01.zip.**

Загружаем скетч на плату Arduino и на экране дисплея видим показания датчика DHT11. Данные также можно посмотреть в мониторе последовательного порта Arduino IDE (рис. 39.2).

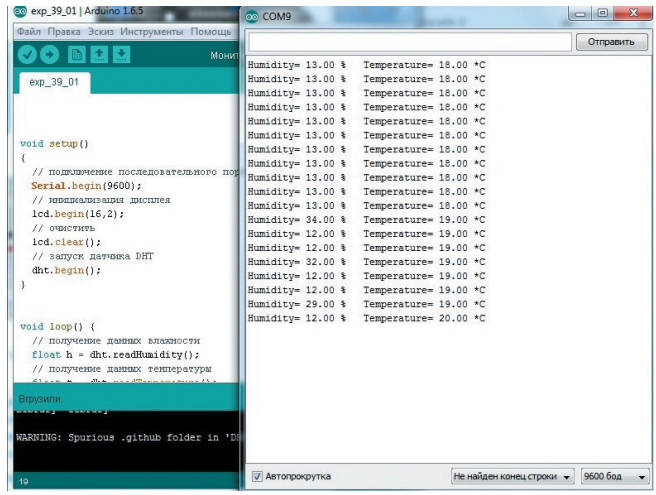


Рис. 39.2. Вывод в последовательный порт данных с датчика DHT11

Эксперимент 40.
Индикатор освещенности на датчике GY30

В этом эксперименте рассмотрим подключение к плате Arduino датчиков, работающих по протоколу I2C и создадим индикатор освещенности на датчике GY-30 и RGB-светодиоде

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Датчик GY-30 – 1;
- ❑ RGB-светодиод – 1;
- ❑ Резистор 220 Ом – 3;
- ❑ Провода MM – 11.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Последовательный протокол обмена данными I2C использует для передачи данных две двунаправленные линии связи:

- ❑ SDA (Serial Data) – шина последовательных данных;
- ❑ SCL (Serial Clock) – шина тактирования.

Также используются две линии для питания. Шины SDA и SCL подтягиваются к шине питания через резисторы.

В сети должно быть хотя бы одно ведущее устройство (master), которое инициализирует передачу данных и генерирует сигналы синхронизации и ведомые устройства (slave), которые передают данные по запросу ведущего. У каждого ведомого устройства есть уникальный адрес, по которому ведущий и обращается к нему. К одной шине I2C может быть подключено до 127 устройств, в том числе несколько ведущих.

Arduino использует для работы по интерфейсу I2C два контакта. Каждая версия Arduino имеет свои выводы I2C:

> 400 – минимальное погружение;

Плата Arduino	Вывод SDA	Вывод SCL
Arduino UNO, Nano	*4*	*5*
Arduino Mega	*20*	*21*
Arduino Leonardo	*2*	*3*
Arduino Due	*20*	*21*

Датчик GY30 на базе чипа BH1750, представляет собой высокоточный цифровой датчик интенсивности света, выдающий значение в люксах (люкс равен освещенности поверхности площадью 1 м2 при световом потоке падающего на нее излучения, равном 1 лм). Данный датчик работает по протоколу I2C. Датчик имеет два варианта подключения к шине I2C (рис. 40.1).

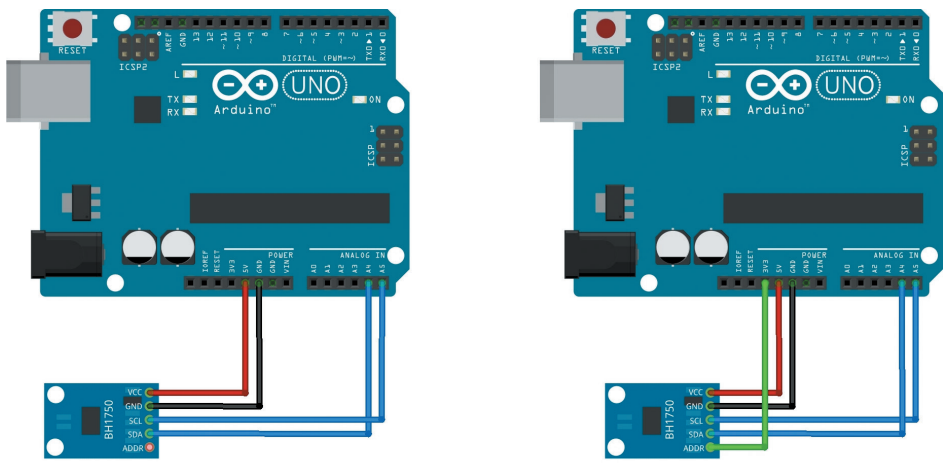


Рис. 40.1. Варианты подключение датчика GY30 к Arduino

Для обмена данными с устройствами по шине I2C в Arduino есть стандартная библиотека Wire.

Загрузим на плату Arduino скетч из листинга 40.1 для поиска устройств, подключенных к шине I2C.

Листинг 40.1

```
#include «Wire.h»
void setup() {
   Wire.begin();
   // запуск последовательного порта
   Serial.begin(9600);
}
```

```
void loop() {
int devices;
byte err, add;

   Serial.println(«Start scan I2C bus...»);
   devices = 0;
   Serial.print(«  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F»);
   for(add = 0; addr<= 127; addr++ ) {
      if((addr% 0x10) == 0) {
         Serial.println();
         if(addr< 16)
         Serial.print('0');
         Serial.print(addr, 16);
         Serial.print(« «);
      }
   Wire.beginTransmission(addr);err = Wire.endTransmission();
   if (err == 0) {
      if (addr<16)
         Serial.print(«0»);
      Serial.print(addr, HEX);
      devices++;
   }
   else {
      Serial.print(«--»);
   }
   Serial.print(« «);
   delay(1);
   }
   Serial.println();
   if (nDevices == 0)
      Serial.println(«No I2C devices found\n»);

   delay(2500);
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_40_01.zip.**

Загружаем скетч на плату Arduino и открываем монитор последовательного порта (рис. 40.2).

Датчик GY-30 может иметь, в зависимости от уровня сигнала на входе ADDR два адреса (0x23 и 0x5C). Это значит, что к одной плате Arduino можно подсоединить одновременно два датчика.

Создадим на датчике GY-30 и RGB-светодиоде индикатор освещенности. Схема соединений показана на рис. 40.3.

Индикатором недостаточной освещенности будет синее свечение RGB-светодиода, излишней – красного, комфортной – зеленого. Нормы освещенности для офисных помещений приведены на рис. 40.4.

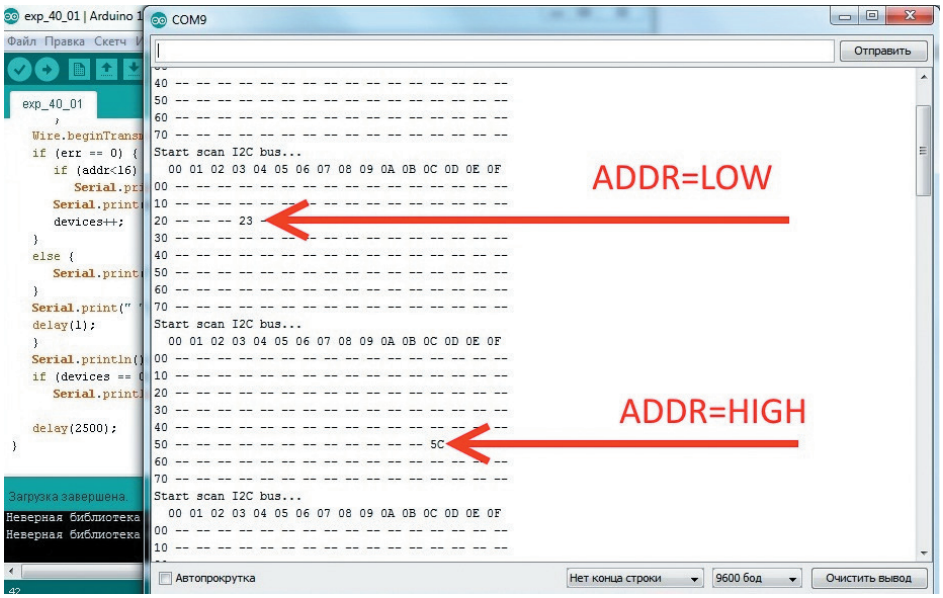


Рис. 40.2. I2C-адреса датчика GY-30 для разных вариантов подключения

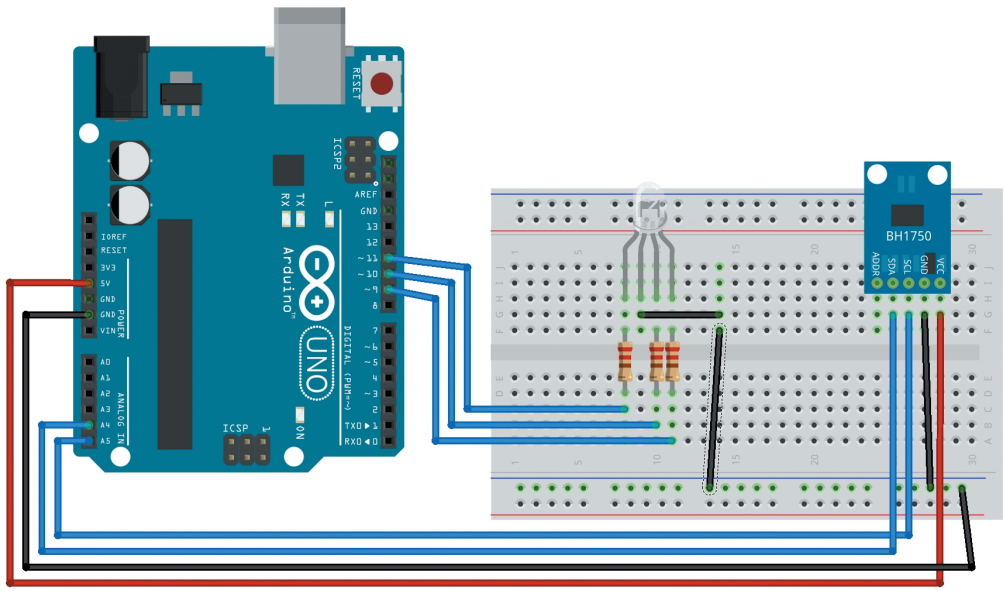


Рис. 40.3. Схема соединений для индикатора освещенности

Нормы освещенности офисных помещений

Тип офиса	Средняя освещенность (E c), люкс	Коэффициент неравномерности освещенности (E min/E ср)	Поверхность нормирования освещенности	Класс помещения*
Небольшой офис	500	0,8	Г 0,8	A-B
Большой открытый офис	750	0,8	Г 0,8	A-B
Конференц-зал	300	0,5	Г 0,8	A-B
Приемная, комната секретаря	300-500	0,5	Г 0,8	A-B
Технические помещения, коридоры	100-200	0,5	Г 0,0	C-D-E

* Примечание:классы помещений: A-B — работа, требующая высокой зрительной концентрации; C — работа, не требующая высокой зрительной концентрации, с высокой степенью мобильности сотрудников; D — работа с низким требованием к зрительной концентрации, при которой работники часто перемещаются в ограниченном пространстве; E — работа с низким требованием к зрительной концентрации (в помещениях нет определенных рабочих мест, сотрудники часто перемещаются в рабочей зоне).

Рис. 40.4. Нормы освещенности офисных помещений

Для получения показаний с датчика GY-30 будем использовать Arduino-библиотеку BH1750, которую можно скачать с сайта Arduino-kit по ссылке https://arduino-kit.ru/scetches/ BH1750FVI.zip.

Содержимое скетча показано в листинге 40.2.

Листинг 40.2.

```
// подключение библиотеки Wire (для I2C)
#include <Wire.h>
// подключение библиотеки для bh1750
#include <BH1750FVI.h>
// создание экземпляра датчика
BH1750FVI myBH1750;
// пины подключения RGB
int pinR=11;
int pinG=9;
int pinB=10;
// границы освещения в норме
const int minL=400;
const int maxL=700;

void setup()
{
   // запуск последовательного порта
   Serial.begin(9600);
   // инициализация дисплея
   // запуск bh1750
   myBH1750.begin();
```

```
    // конфигурация пинов RGB
    pinMode(pinR,OUTPUT);
    pinMode(pinG,OUTPUT);
    pinMode(pinB,OUTPUT);
}

void loop() {
    // получение данных с датчика
    myBH1750.setSensitivity(1);
    Serial.print(«Light level= «);
    int light=myBH1750.readLightLevel();
    Serial.print(light);
    Serial.println(«lx»);
    if(light<minL) {
        // от синего к зеленому
        int offset=map(light,0,minL,0,255);
        setRGB(0,255-offset,offset);
    }
    else if(light>minL) {
        // от зеленого к красному
        int offset=map(light,maxL,5000,0,255);
        setRGB(offset,255-offset,0);
    }
    else {
        // зеленый
        setRGB(0,255,0);
    }
    delay(1000);
}
// установка цвета RGB-светодиода
void setRGB(int r, int g, int b) {
    analogWrite(pinR, r);
    analogWrite(pinG, g);
    analogWrite(pinB, b);
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_40_02.zip.**

Загружаем скетч на плату Arduino. Проверяем работу индикатора освещения.

Эксперимент 41.
Домашняя метеостанция на датчике BMP280 и DHT11

В этом эксперименте подключим к плате Arduino датчик атмосферного давления BMP280 и создадим домашнюю метеостанцию с выводом данных на ЖК-дисплей

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Датчик BMP280 – 1;
- ❑ Датчик DHT11 – 1;
- ❑ LCD Keypad shield – 1;
- ❑ Провода MM – 15.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Барометр – устройство, измеряющее атмосферное давление. Электронные барометры используются в робототехнике и различных электронных устройствах. Наиболее распространенными и доступными являются датчики давления от фирмы BOSH: BMP085, BMP180, BMP280 и другие. Датчик BMP280 создан специально для приложений, где требуются малые размеры и пониженное потребление энергии. Он измеряет атмосферное давление и температуру. Используется обычно для определения высоты и в метеостанциях. Состоит из пьезо-резистивного датчика, термодатчика, АЦП, энергонезависимой памяти, ОЗУ и микроконтроллера. Датчик работает по протоколам I2C и SPI. Схема соединений для подключения датчика BMP280 по протоколу I2C показана на рис. 41.1.

Для работы с датчиком будем использовать Arduino-библиотеку Adafruit_BMP280, которую можно скачать с сайта Arduino-kit по ссылке **https://arduino-kit.ru/scetches/Adafruit_BMP280.zip.** Установим библиотеку и загрузим пример из библиотеки bmp280test. Откроем монитор последовательного порта и что мы видим – Arduino не находит датчик (рис. 41.2)!

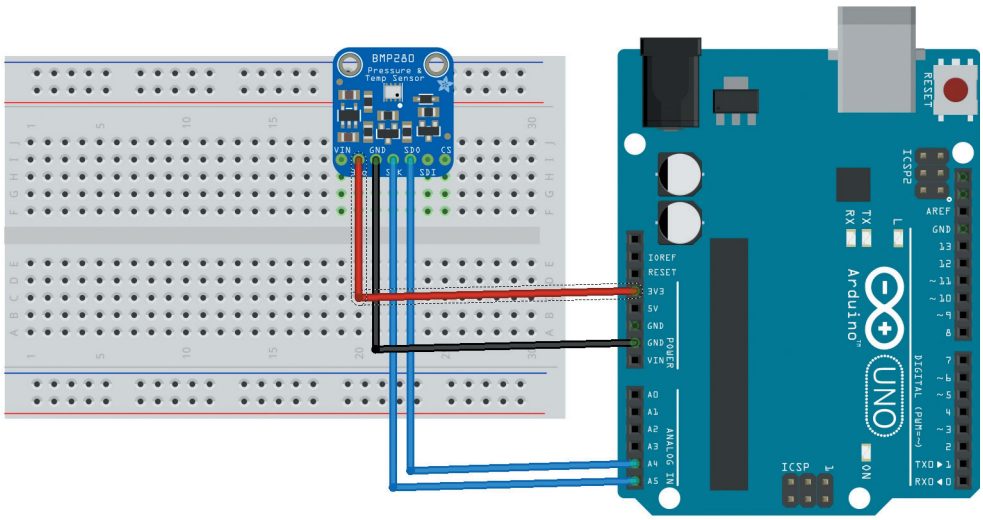


Рис. 41.1. Схема соединений для подключения датчика BMP180 к Arduino

Разбираемся в чем дело. Загрузим на плату Arduino скетч из листинга 40.1 для поиска устройств, подключенных к шине I2C, и откроем монитор последовательного порта (рис. 4132). Теперь открываем с помощью Блокнота файл

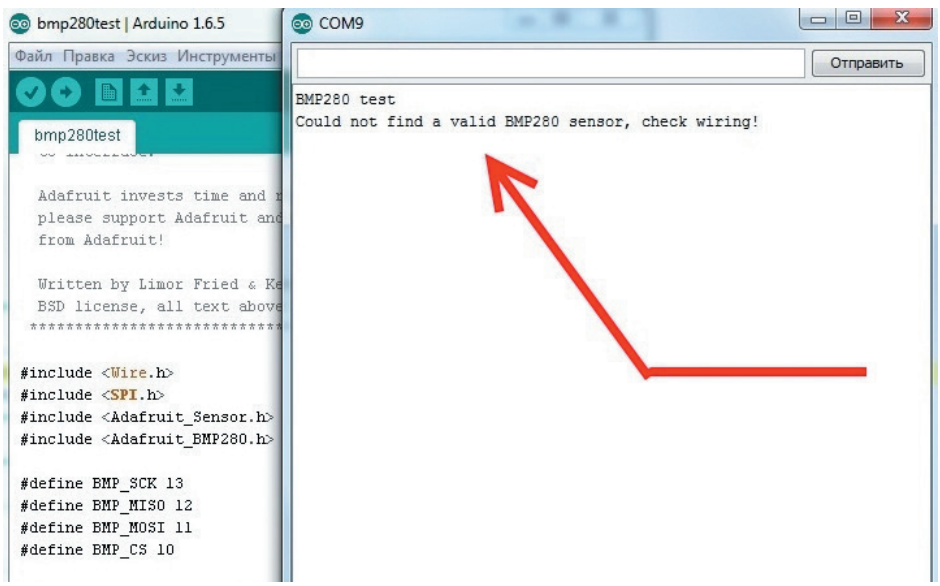


Рис. 41.3. I2C-адрес датчика BMP280

Adafruit_BMP280.h, находим строку

```
#define BMP280_ADDRESS                (0x77)
```

и меняем значение 0x77 на 0x76 (рис. 41.4).

```
/*=========================================================================
    I2C ADDRESS/BITS/SETTINGS
    -----------------------------------------------------------------------*/
    #define BMP280_ADDRESS                (0x77)
    #define BMP280_CHIPID                 (0x58)
/*=========================================================================*/

/*=========================================================================
    REGISTERS
    -----------------------------------------------------------------------*/
    enum
```

Рис. 41.4. Редактирование файла Adafruit_BMP280.h

Сохраняем. Загрузим пример из библиотеки bmp280test. Теперь все нормально – Arduino находит датчик BMP280 и считывает данные (рис. 41.5).

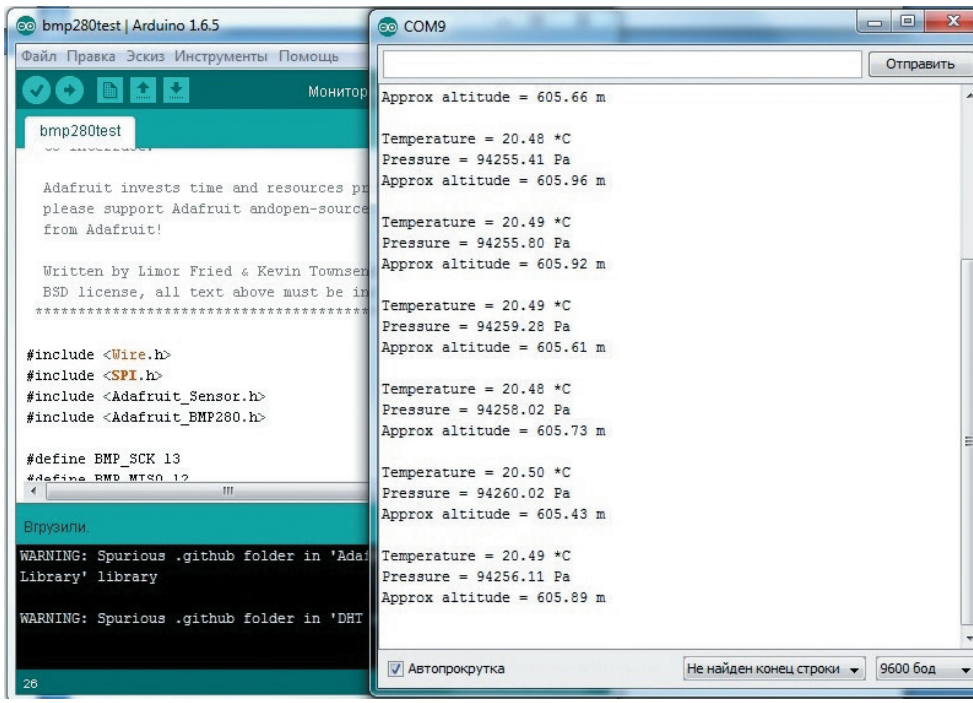


Рис. 41.5. Пример bmp280test – данные с датчика BMP280

Создадим домашнюю метеостанцию на датчиках BMP280 и DHT11. Для визуализации данных будем использовать LCD Keypad shield. Схема соединений показана на рис. 41.6

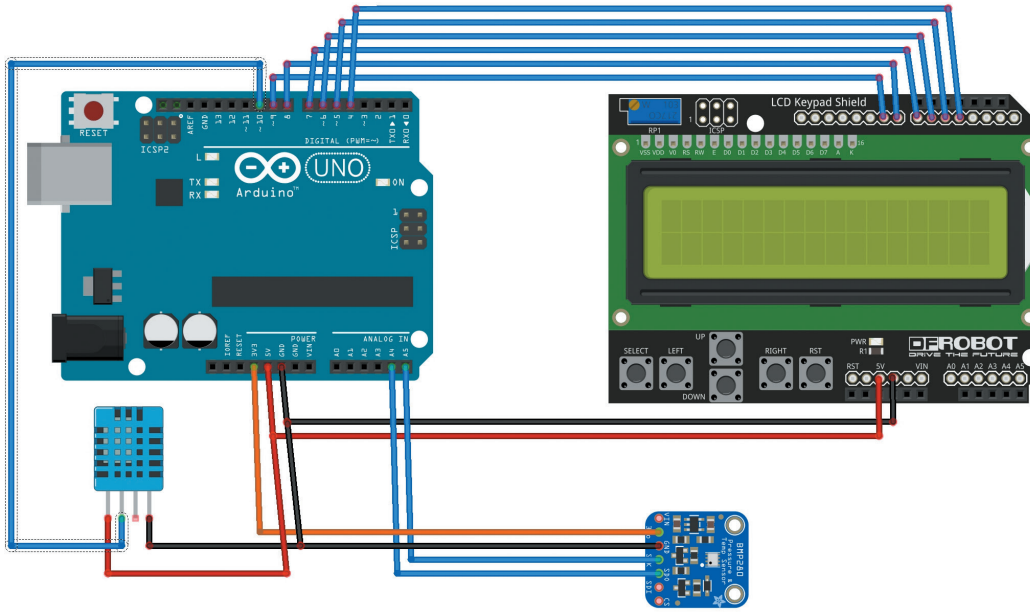


Рис. 41.6. Схема соединений для метеостанции на датчиках BMP180 и DHT11

Приступим к написанию скетча. Данные будем получать раз в 30 секунд, и выводить на экран. Содержимое скетча показано в листинге 41.1.

Листинг 41.1.

```
// подключение библиотек
#include <LiquidCrystal.h>
#include "DHT.h"
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
// пин для подключения датчика DHT
#define DHTPIN 12
// тип датчика DHT
#define DHTTYPE DHT11   // DHT 11
// создание экземпляров
Adafruit_BMP280 bmp;
LiquidCrystal lcd(8,9,4,5,6,7);
DHT dht(DHTPIN, DHTTYPE);
// для опроса
unsigned long millist=0;

void setup()
{
```

```
  // подключение последовательного порта
  Serial.begin(9600);
  // инициализация дисплея
  lcd.begin(16,2);
  // очистить
  lcd.clear();
  // запуск датчиков DHT,BMP20
  dht.begin();
  bmp.begin();
}

void loop() {
  if(millis()-millist>=30000)
     {
     // получение данных
     int h = dht.readHumidity();
     int t = bmp.readTemperature();
     int p = bmp.readPressure();
     // вывод на дисплей
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("H=");
     lcd.print(h);lcd.print("%");
     lcd.setCursor(7,0);
     lcd.print("T=");
     lcd.print(t);lcd.print("*C");
     lcd.setCursor(0,1);
     lcd.print("P= ");
     lcd.print(p);lcd.print("Pa");
     //
     millist=millis();
     }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_41_01.zip.**

Загружаем скетч на плату Arduino и на экране дисплея видим показания атмосферного давления и температуры с датчика BMP280 и относительной влажности воздуха с датчика DHT11.

Эксперимент 42.
Часы реального времени DS3231. Установка (корректировка) времени

В этом эксперименте рассмотрим подключение к плате Arduino модуля часов реального времени на микросхеме DS3231, а также корректировку времени по последовательному порту

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Модуль DS3231 – 1;
- ❑ Провода MM – 4;

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Если вы создаете устройство, которому нужно знать точное время, вам пригодится модуль часов реального времени RTC (Real Time Clock). Данные модули отсчитывают точное время и могут сохранять его даже при отключении основного питания при использовании резервного питания (батарейка CR2032), которого хватит на несколько лет.

Еще совсем недавно основным модулем RTC в среде Ардуинщиков являлся модуль на микросхеме DS1307. В этом модуле использовался внешний кварцевый генератор частотой 32 кГц, при изменении температуры менялась частота кварца, что приводило к погрешности в подсчете времени.

Новые модули RTC построены на микросхеме DS3231, внутрь которой установлен кварцевый генератор и датчик температуры, который компенсирует изменения температуры, поэтому время отсчитывается более точно. Погрешность составляет ±2 минуты за год.

Рассмотрим, как получать данные с модуля реального времени и как устанавливать точное время, если требуется его корректировка.

Модуль DS3231 подключается к плате Arduino по интерфейсу I2C, используются выводы SDA и SCL. Схема подключения показана на рис. 42.1.

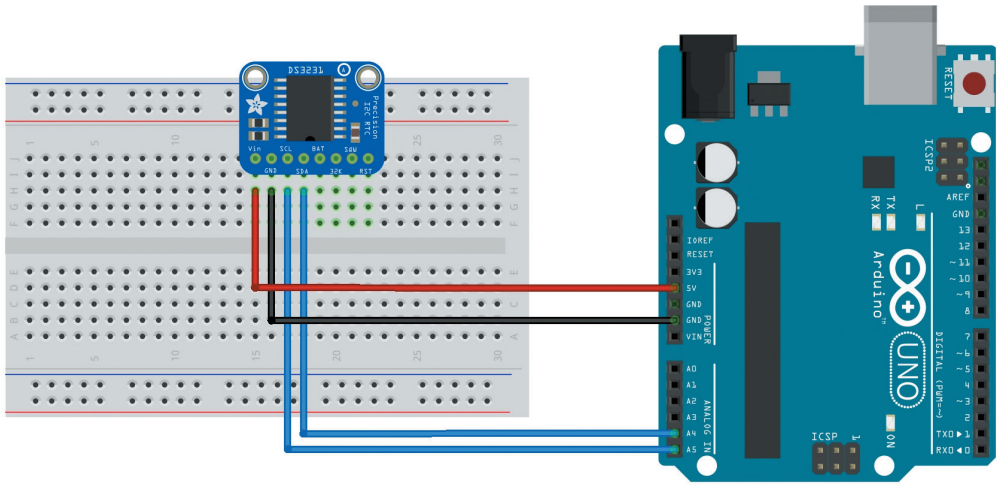


Рис. 42.1. Схема соединений для подключения DS3231 к Arduino

Для программирования будем использовать библиотеки DS1307 и TimeLib, которые можно скачать с сайта Arduino-kit по ссылкам https://arduino-kit.ru/scetches/TimeLib.zip и https://arduino-kit.ru/scetches/DS1307RTC.zip. Скетч получения данных с DS3231 и вывода в последовательный порт показан в листинге 42.1.

Листинг 42.1.

```
// подключение библиотек
#include <Wire.h>
#include <DS1307RTC.h>
#include <Time.h>
#include <TimeLib.h>

tmElements_t datetime;

void setup() {
   Serial.begin(9600);
}

void loop() {
   // получение данных из ds3231
   if (RTC.read(datetime))  {
      print2(datetime.Hour, ":");
      print2(datetime.Minute, ":");
      print2(datetime.Second, " ");
      print2(datetime.Day, "/");
      print2(datetime.Month, "/");
```

```
        print2(tmYearToCalendar(datetime.Year) ,"");
        Serial.println();
    }
    else {
          Serial.println("error");
          delay(5000);
    }
    delay(1000);
}
// добавление до 2 цифр
void print2(int nn,String str) {
    if (nn >= 0 && nn < 10)
        { Serial.print("0");}
    Serial.print(nn);
    Serial.print(str);
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_42_01.zip.

Загружаем скетч на плату Arduino и открываем монитор последовательного порта (рис. 42.2).

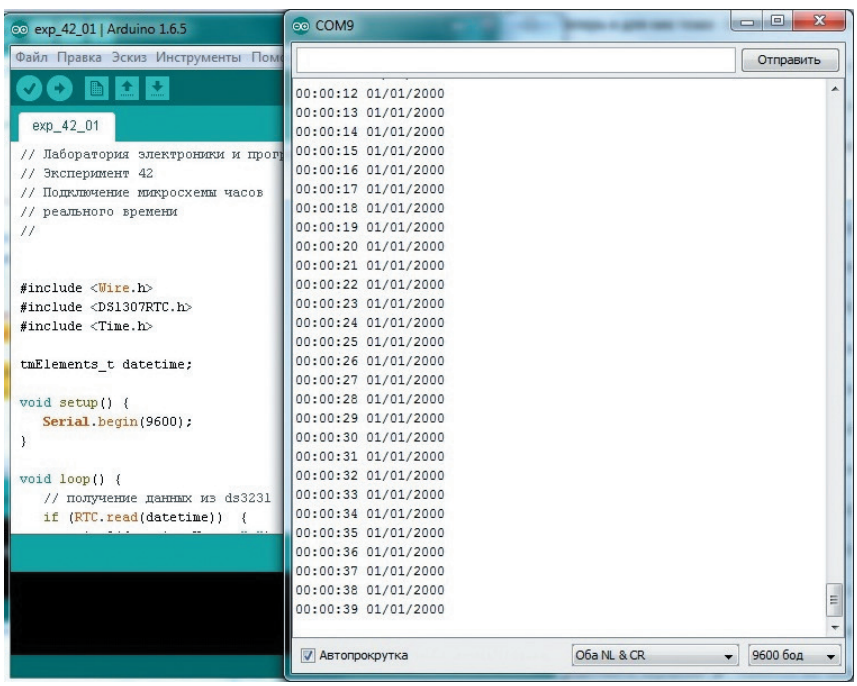


Рис. 42.2. Вывод, получаемых с DS3231, данных в последовательный порт

Если отсчет времени идет неправильно, то данные неверные. Необходимо на модуле DS3231 установить точное время, и если модуль использует внешнее питание, то это время будет сохраняться даже после отключения от платы Arduino. Напишем скетч установки времени отправкой строки вида "dd/mm/YYYY hh:mm:ss" в последовательный порт Arduino. Содержимое скетча показано в листинге 42.2.

Листинг 42.2.

```
// подключение библиотек
#include <DS1307RTC.h>
#include <Time.h>
#include <TimeLib.h>
#include <Wire.h>
// служебные переменные
String inSer = "";
boolean strFull = false;
tmElements_t datetime;

void setup() {
   Serial.begin(9600);
}

void loop() {
   // пришли данные по serial
   if (strFull) {
      datetime.Hour=(int(inSer[11])-48)*10+(int(inSer[12])-48);
      datetime.Minute=(int(inSer[14])-48)*10+(int(inSer[15])-48);
      datetime.Second=(int(inSer[17])-48)*10+(int(inSer[18])-48);
      datetime.Day=(int(inSer[0])-48)*10+(int(inSer[1])-48);
      datetime.Month=(int(inSer[3])-48)*10+(int(inSer[4])-48);
     datetime.Year=CalendarYrToTm((int(inSer[6])-48)*1000+(int(inSer
[7])-48)*100+(int(inSer[8])-48)*10+(int(inSer[9])-48));
      RTC.write(datetime); // записать данные в DS3231
      // очистить строку
      inSer = "";
      strFull = false;
   }
    // получение данных из ds3231
   if (RTC.read(datetime))  {
      print2(datetime.Hour, ":");
      print2(datetime.Minute, ":");
      print2(datetime.Second, " ");
      print2(datetime.Day, "/");
      print2(datetime.Month, "/");
      print2(tmYearToCalendar(datetime.Year) ,"");
      Serial.println();
   }
   else {
```

```
            Serial.print("error");
            delay(5000);
      }
      delay(1000);
   }
   // добавление до 2 цифр
   void print2(int nn,String str) {
      if (nn >= 0 && nn < 10)
         { Serial.print("0");}
      Serial.print(nn);
      Serial.print(str);
   }
   void serialEvent() {
      while (Serial.available()) {
         // получить очередной байт:
         char c = (char)Serial.read();
         // добавить в строку
         inSer += c;
         // /n - конец передачи
         if (c == '\n')
            { strFull = true;}
      }
   }
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_42_02.zip.

Загружаем скетч на плату Arduino, открываем монитор последовательного порта, набираем строку вида "dd/mm/YYYY hh:mm:ss" и время становится верным (рис. 42.3).

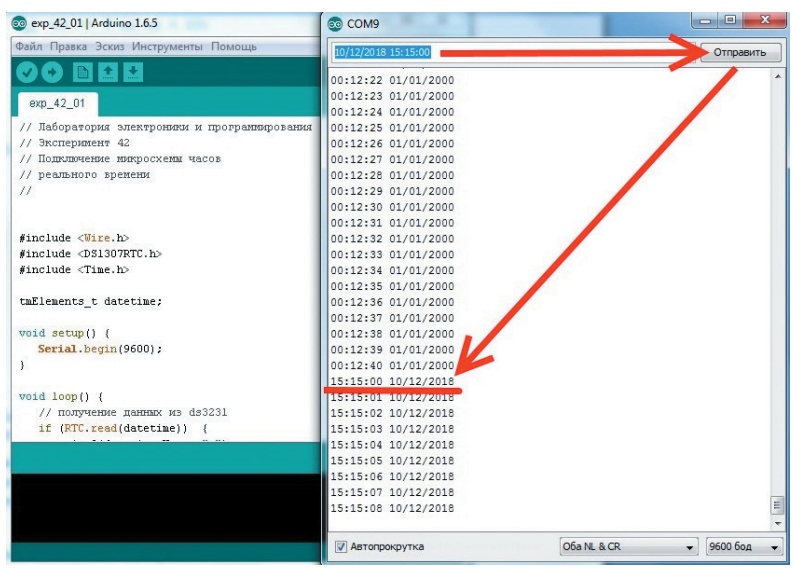


Рис. 42.3 Изменение времени по последовательному порту

Эксперимент 43.
Часы на 4-х разрядной светодиодной матрице

В этом эксперименте рассмотрим использование экрана 4-х разрядной светодиодной матрицы в качестве часов

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Модуль DS3231 – 1;
- ❑ 4-х разрадная светодиодная матрица – 1;
- ❑ Провода MF – 5.
- ❑ Провода MM – 7

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

В эксперименте 42 мы научились использовать модуль DS3231 для получения реального времени. В этом эксперименте создадим часы с использовангием модуля DS3231, используя в качестве экрана 4-х разрядную светодиодную матрицу.

Схема подключения показана на рис. 43.1.

Приступим к написанию скетча. Необходимо получить данные о времени (часы, минуты) с модуля DS3231, и вывести 4 цифры на матрицы, кроме того необходимо организовать разделитель часов и минут – мигание точек между показаниями с частотой 1 секунда. Содержимое скетча показано в листинге 43.1.

Листинг 43.1.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>
#include <Wire.h>
#include <DS1307RTC.h>
#include <Time.h>
#include <TimeLib.h>
```

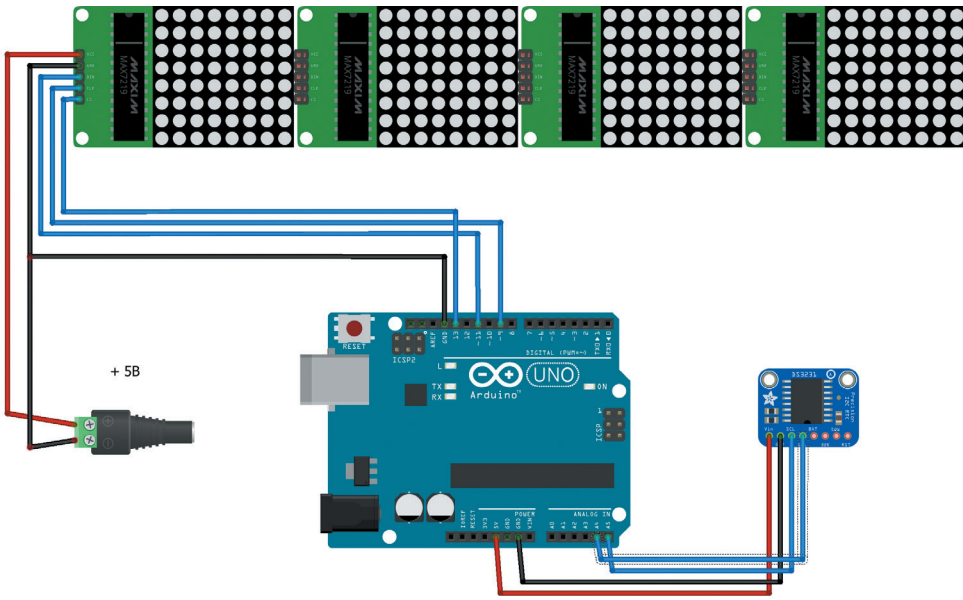


Рис. 43.1. Схема соединений для часов на DS3231 и 4-х разрядной светодиодной матрице

```
// пин CS
int pinCS = 9;
// количество матриц по-горизонтали
int numberOfHorizontal = 4;
// количество матриц по-вертикали
int numberOfVertical = 1;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal,
                      numberOfVertical);
// текст для вывода ошибки
String texterror = "ERROR";
unsigned long millist=0;
// переменная времени
tmElements_t datetime;
// показ разделителя
boolean blinkview=true;

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

```
void loop() {
  // прошло 500 мсек?
  if(millis()-millist>=500) {
     millist=millis();
     // получение данных из ds3231
     if (RTC.read(datetime))  {
        matrix.setRotation( 0, 1 );
        matrix.drawChar(1, 0, datetime.Hour/10+0x30, HIGH, LOW, 1);
        matrix.setRotation( 1, 1 );
        matrix.drawChar(9, 0, datetime.Hour%10+0x30, HIGH, LOW, 1);
        matrix.setRotation( 2, 1 );
       matrix.drawChar(17, 0, datetime.Minute/10+0x30, HIGH, LOW, 1);
        matrix.setRotation( 3, 1 );
       matrix.drawChar(25, 0, datetime.Minute/10+0x30, HIGH, LOW, 1);
        matrix.write();
        // разделитель
        matrix.drawPixel(15, 2, blinkview);
        matrix.drawPixel(15, 5, blinkview);
        blinkview=!blinkview;
     }
     else {
        for ( int i = 0 ; i < texterror.length(); i++ ) {
           matrix.setRotation( i, 1 );
           matrix.drawChar(i*6, 0, texterror[i], HIGH, LOW, 1);
        }
        matrix.write();
    }
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_43_01.zip.**

Загружаем скетч на плату Arduino и получаем часы на светодиодной матрице (рис. 42.2).

Эксперимент 44.
Часы с бегущей строкой на 4-х разрядной светодиодной матрице

В этом эксперименте добавим функционал для часов на модуле DS3231 и 4-х разрядной светодиодной матрице.

В эксперименте мы будем использовать компоненты из эксперимента 43. Схема соединений согласно рис. 43.1.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

В эксперименте 43 мы создали часы на модуле DS3231, используя в качестве экрана 4-х разрядную светодиодную матрицу. В этом эксперименте добавим функционал часам, организуем вывод на экран:

- даты (день-месяц);
- года;
- дня недели.

Смена экранов будем производить "бегущей строкой".

Приступим к написанию скетча. Создадим две функции:

- viewData() – вывод данных;
- viewrunningData() – вывод данных бегущей строки

```
// вывод неподвижного текста
void viewData(String txt,int m[8],boolean f) {
   Serial.println(txt);
   // вывод текста
   for ( int i = 0 ; i < txt.length(); i++ ) {
      matrix.setRotation( i, 1 );
      matrix.drawChar(i*8+2, 0, txt[i], HIGH, LOW, 1);
   }
   // вывод разделителя
   for(int i=0;i<8;i++) {
      matrix.drawPixel(16, i, m[i]&f);
   }
   matrix.write();
```

```
}
// вывод бегущей строки
void viewrunningData(String txt,int speedstr) {
   int offset=0;
   Serial.println(txt);
   // 48 шагов
   for(int offset=0;offset<48;offset++) {
      matrix.fillScreen(LOW);
      // вывод строки с позиции offset
      for ( int i = 0 ; i < txt.length(); i++ ) {
         matrix.setRotation( i, 1 );
         matrix.drawChar(i*8-offset+1, 0, txt[i], HIGH, LOW, 1);
      }
      matrix.write();
      delay(speedstr);
   }
}
```

Данные для экрана и бегущей строки формируются в зависимости от значения переменной pos (0-7), согласно таблице

pos	*Данные для экрана*
0	*"hh:ss"*
1	*Бегущая строка "hh:ss dd/mm"*
2	*"dd:mm"*
3	*Бегущая строка "dd:mm YYYY"*
4	*"YYYY"*
5	*Бегущая строка "YYYY Wday"*
6	*"Wday"*
7	*Бегущая строка "Wday dd/mm"*

Содержимое скетча показано в листинге 44.1.

Листинг 44.1.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>
#include <Wire.h>
#include <DS1307RTC.h>
#include <Time.h>
#include <TimeLib.h>
// пин CS
```

```
int pinCS = 9;
// количество матриц по-горизонтали
int numberOfHorizontal = 15;
// количество матриц по-вертикали
int numberOfVertical = 1;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal,
                      numberOfVertical);
// текст для вывода ошибки
String text ="";
unsigned long millist=0;
// переменная времени
tmElements_t datetime;
// показ разделителя
boolean blinkview=true;
// текущий режим
// 0 - hh:ss
// 1 - hh:ss --> dd/mm
// 2 - dd:mm
// 3 - dd:mm --> YYYY
// 4 - YYYY
// 5 - YYYY --> Wday
// 6 - Wday
// 7 - Wday --> dd/mm

int pos=0;
// для отсчета времени
unsigned long millispos=0;
String weekdays[]={
     {"Sun "},{"Mon "},{"Tue "},{"Wed "},
     {"Thu "},{"Fri "},{"Sat "}
};
// разделители
int bl0[]={0,0,1,0,0,1,0,0};
int bl1[]={0,0,1,1,1,0,0,0};

void setup() {
    Serial.begin(9600);
    // яркость от 0 до 15
    matrix.setIntensity(7);
}


void loop() {
```

```
switch(pos) {
  case 0:
          millispos=millis();
          while(millis()-millispos<4000) {
             if(millis()-millist>=500) {
                millist=millis();
                RTC.read(datetime);
                text="";
                text+=char(datetime.Hour/10+0x30);
                text+=char(datetime.Hour%10+0x30);
                text+=char(datetime.Minute/10+0x30);
                text+=char(datetime.Minute%10+0x30);
                viewData(text,bl0,blinkview);
                blinkview=!blinkview;
             }
          }
          pos=1;
          Serial.println("pos=1");
     break;
  case 1:
           RTC.read(datetime);
           text="";
           text+=char(datetime.Hour/10+0x30);
           text+=char(datetime.Hour%10+0x30);
           text+=char(datetime.Minute/10+0x30);
           text+=char(datetime.Minute%10+0x30);
           text+="  ";
           text+=char(datetime.Day/10+0x30);
           text+=char(datetime.Day%10+0x30);
           text+=char(datetime.Month/10+0x30);
           text+=char(datetime.Month%10+0x30);
           viewrunningData(text,20);
           pos=2;
           Serial.println("pos=2");
     break;
  case 2:
          millispos=millis();
          while(millis()-millispos<3000) {
             if(millis()-millist>=500) {
                millist=millis();
                RTC.read(datetime);
                text="";
                text+=char(datetime.Day/10+0x30);
                text+=char(datetime.Day%10+0x30);
```

```
            text+=char(datetime.Month/10+0x30);
            text+=char(datetime.Month%10+0x30);
            viewData(text,bl1,1);
         }
      }
      pos=3;
      Serial.println("pos=3");
   break;
case 3:
       RTC.read(datetime);
       text="";
       text+=char(datetime.Day/10+0x30);
       text+=char(datetime.Day%10+0x30);
       text+=char(datetime.Month/10+0x30);
       text+=char(datetime.Month%10+0x30);
       text+="  ";
       text+=String(tmYearToCalendar(datetime.Year));
       viewrunningData(text,20);
       pos=4;
       Serial.println("pos=4");
   break;
case 4:
      millispos=millis();
      while(millis()-millispos<3000) {
         if(millis()-millist>=500) {
            millist=millis();
            RTC.read(datetime);
            text=String(tmYearToCalendar(datetime.Year));
            viewData(text,bl1,0);
         }
      }
      pos=5;
      Serial.println("pos=5");
   break;
case 5:
       RTC.read(datetime);
       text="";
       text+=String(tmYearToCalendar(datetime.Year));
       text+="  ";
       text+=weekdays[datetime.Wday-1];
       viewrunningData(text,20);
       pos=6;
       Serial.println("pos=6");
   break;
```

```
    case 6:
            millispos=millis();
            while(millis()-millispos<3000) {
               if(millis()-millist>=500) {
                  millist=millis();
                  RTC.read(datetime);
                  Serial.println(datetime.Wday);
                  Serial.println(weekdays[datetime.Wday]);
                  text=weekdays[datetime.Wday-1];
                  viewData(text,bl1,0);
               }
            }
            pos=7;
            Serial.println("pos=7");
       break;
    case 7:
             RTC.read(datetime);
             text="";
             text+=weekdays[datetime.Wday-1];
             text+="  ";
             text+=char(datetime.Hour/10+0x30);
             text+=char(datetime.Hour%10+0x30);
             text+=char(datetime.Minute/10+0x30);
             text+=char(datetime.Minute%10+0x30);
             viewrunningData(text,20);
             pos=0;
             Serial.println("pos=0");
       break;
    default:
       break;
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_44_01.zip.**

Загружаем скетч на плату Arduino и получаем часы на светодиодной матрице, где циклически на экране показывается время, дата, год и день недели.

Эксперимент 45.
Часы на ЖК-дисплее LCD Keypad shield

В этом эксперименте создадим часы на модуле DS3231 с экраном на ЖК-дисплее

В эксперименте мы будем использовать следующие компоненты:

- Плата Arduino UNO – 1;
- Кабель USB – 1;
- Плата прототипирования – 1;
- Модуль DS3231 – 1;
- LCD Keypad shield – 1;
- Провода ММ – 12.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

В этом эксперименте создадим часы на модуле DS3231, используя в качестве экрана ЖК-дисплей LCD Keypad shield.

Схема подключения показана на рис. 45.1.

Приступим к написанию скетча. На первой строке ЖК-дисплея будем показывать текущее время в формате "HH-mm-ss". На второй строке ЖК-дисплея будем поочередно с периодичностью 5 секунд показывать либо текущую дату в формате "dd-mm-YYYY", либо день недели. Для программирования будем использовать библиотеки DS1307, Time, которые мы уже загрузили в эксперименте 42, и стандартную библиотеку LiquidCrystal. Оставляем возможность корректировки данных DS3231 по последовательному порту. Содержимое скетча показано в листинге 45.1.

Листинг 45.1.

```
// подключение библиотек
#include <Wire.h>
#include <DS1307RTC.h>
#include <Time.h>
#include <TimeLib.h>
#include <LiquidCrystal.h>
// создание экземпляра дисплея
```

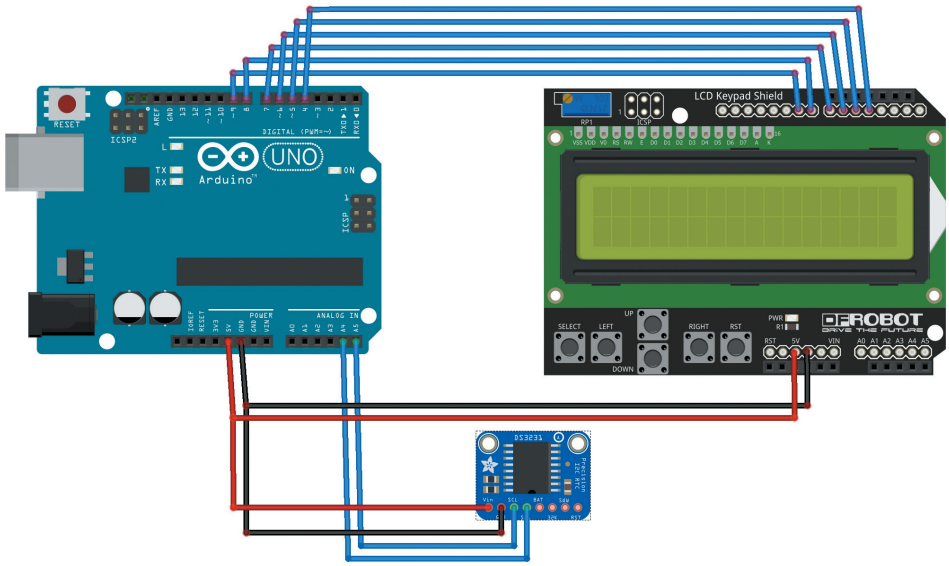


Рис. 45.1. Схема соединений для часов на DS3231 и LCD Keypad shield

```
LiquidCrystal lcd(8,9,4,5,6,7);
// переменная времени-даты
tmElements_t datetime;
// массив дней недели
String weekdays[]={
     {"  Sunday  "},{"  Monday  "},{" Tuesday  "},{"Wednesday "},
     {" Thursday "},{"  Friday  "},{" Saturday "}
};
// для смены данных второй строки
boolean week=false;
unsigned long millis1=0;

void setup() {
   Serial.begin(9600);
   // запуск дисплея
   lcd.begin(16,2);
   // вывод заставки
   lcd.setCursor(3,0);
   lcd.print("Clock DS3231");
   lcd.setCursor(2,1);
   lcd.print("Arduino-kit.ru");
   delay(3000);
   lcd.clear();
}

void loop() {
```

```
    // получение данных из ds3231
    if (RTC.read(datetime))  {
       // время HH:mm:ss
       lcd.setCursor(4,0);
       lcd.print(print2(datetime.Hour, ":"));
       lcd.print(print2(datetime.Minute, ":"));
       lcd.print(print2(datetime.Second, " "));
       // дата или день недели
       lcd.setCursor(3,1);
       Serial.println();
       if(week) { // день недели
          lcd.print(weekdays[datetime.Wday]);
       }
       else {    // дата dd/mm/YYYY
          lcd.print(print2(datetime.Day, "/"));
          lcd.print(print2(datetime.Month, "/"));
          lcd.print(tmYearToCalendar(datetime.Year));
       }
       // изменение week каждые 5 секунд
       if(millis()-millis1>5000) {
          week=!week;
          millis1=millis();
       }
    }
    else {
          lcd.print("error");
          delay(5000);
    }
    delay(1000);
}
String print2(int nn,String str) {
    String s="";
    if (nn >= 0 && nn < 10)
       { s=s+"0";}
    s=s+String(nn);
    s=s+str;
    return s;
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_45_01.zip.**

Загружаем скетч на плату Arduino и на экране LCD Keypad shield получаем часы с отображением времени, даты и дня недели.

Эксперимент 46.
Добавляем часам на ЖК-дисплее LCD Keypad shield функционал будильника

В этом эксперименте добавим часам на ЖК-дисплее LCD Keypad shield функционал будильника

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Модуль DS3231 – 1;
- ❑ LCD Keypad shield – 1;
- ❑ Динамик – 1;
- ❑ Провода MM – 12.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

В эксперименте 45 мы создали часы на модуле DS3231. Добавим данным часам функционал будильника. Схема соединений показана на рис. 46.1.

Приступим к написанию скетча. Создадим структуру, описывающую будильник.

```
struct ALARM{
  int Hour;         // час срабатывания будильника
  int Minute;       // минута срабатывания будильника
  int Day;          // день срабатывания будильника
  int Month;        // месяц срабатывания будильника
  int Year;         // год срабатывания будильника
  boolean repeat;   // true – повторять, false -одноразово
  int wdays[7];     // дни недели
};
```

И создадим список необходимых будильников:

```
ALARM alarms[]={{7,30,14,12,2018,1,{1,1,1,1,1,0,0}},
                {12,35, 14,12,2018,0,{0,0,0,0,0,0,0}},
                {18,00, 5,12,2018,1,{1,1,1,1,1,0,0}},
                {22,10, 10,12,2018,1,{1,1,1,1,1,1,1}},
       };
```

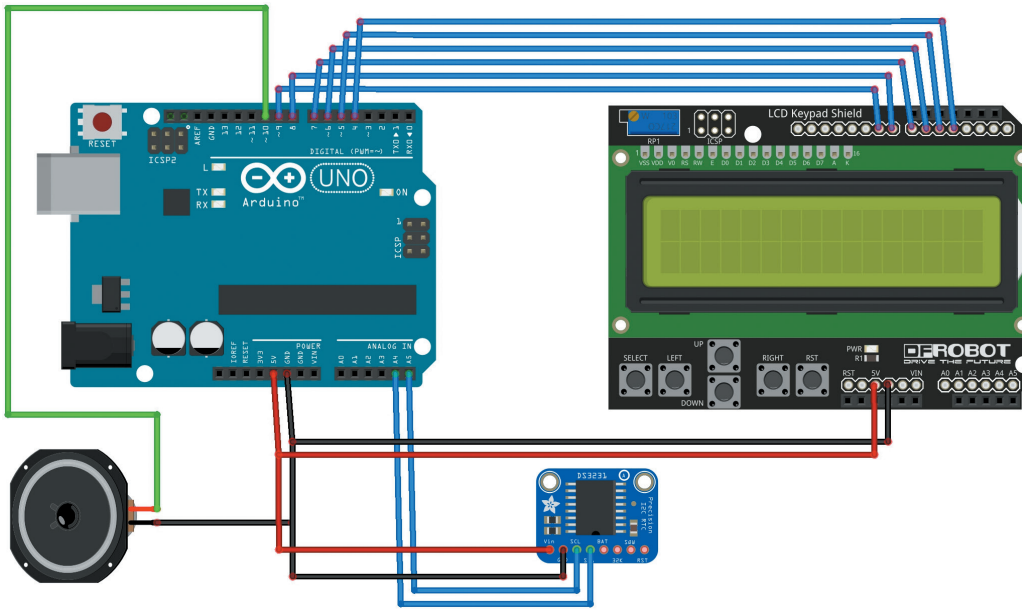


Рис. 46.1. Схема соединений для часов с будильником на DS3231 и LCD Keypad shield

В цикле loop() скетча необходимо проводить проверку наступления события по расписанию и необходимых действий при наступлении события (воспроизведение звука, прерываемого удержанием более 1 сек кнопки RIGHT на клавиатуре LCD Keypad shield). Содержимое скетча показано в листинге 46.1.

Листинг 46.1.

```
// подключение библиотек
#include <Wire.h>
#include <DS1307RTC.h>
#include <Time.h>
#include <TimeLib.h>
#include <LiquidCrystal.h>
// создание экземпляра дисплея
LiquidCrystal lcd(8,9,4,5,6,7);
// переменная времени-даты
tmElements_t datetime;
// массив дней недели
String weekdays[]={
     {"  Sunday  "},{"  Monday  "},{" Tuesday  "},{"Wednesday "},
     {" Thursday "},{"  Friday  "},{" Saturday "}
```

```
};
// для смены данных второй строки
boolean week=false;
unsigned long millis1=0;
// структура, описывающая будильник
struct ALARM{
  int Hour;        // час срабатывания будильника
  int Minute;      // минута срабатывания будильника
  int Day;         // день срабатывания будильника
  int Month;       // месяц срабатывания будильника
  int Year;        // год срабатывания будильника
  boolean repeat;  // true - повторять, false - одноразово
  int wdays[7];    // дни недели
};
// список будильников
ALARM alarms[]={{7,30,14,12,2018,1,{1,1,1,1,1,0,0}},
                {12,35, 14,12,2018,0,{0,0,0,0,0,0,0}},
                {18,00, 5,12,2018,1,{1,1,1,1,1,0,0}},
                {22,10, 10,12,2018,1,{1,1,1,1,1,1,1}},
              };
// будильник работает?
boolean alarmyes=false;
// пин подключения динамика
int pinSpeaker=10;

void setup() {
   Serial.begin(9600);
   // запуск дисплея
   lcd.begin(16,2);
   // вывод заставки
   lcd.setCursor(3,0);
   lcd.print("Clock DS3231");
   lcd.setCursor(2,1);
   lcd.print("Arduino-kit.ru");
   delay(3000);
   lcd.clear();
   // пин динамика
   pinMode(pinSpeaker,OUTPUT);
}

void loop() {
   // выключение будильника
   if(alarmyes==true) {
      if(analogRead(A0)<100) { // кнопка RIGHT нажата?
```

```
            alarmyes=false;
            noTone(pinSpeaker);
            Serial.println("ALARM STOP");
         }
      }

      // получение данных из ds3231
      if (RTC.read(datetime))  {
         // время HH:mm:ss
         lcd.setCursor(4,0);
         lcd.print(print2(datetime.Hour, ":"));
         lcd.print(print2(datetime.Minute, ":"));
         lcd.print(print2(datetime.Second, " "));
         // дата или день недели
         lcd.setCursor(3,1);
         Serial.println();
         if(week) { // день недели
            lcd.print(weekdays[datetime.Wday]);
         }
         else {    // дата dd/mm/YYYY
            lcd.print(print2(datetime.Day, "/"));
            lcd.print(print2(datetime.Month, "/"));
            lcd.print(tmYearToCalendar(datetime.Year));
         }
         // изменение week каждые 5 секунд
         if(millis()-millis1>5000) {
            week=!week;
            millis1=millis();
         // проверка времение срабатывания будильника
         isalarm();
         }
      }
      else {
            lcd.print("error");
            delay(5000);
      }
      delay(1000);
   }
   String print2(int nn,String str) {
      String s="";
      if (nn >= 0 && nn < 10)
         { s=s+"0";}
      s=s+String(nn);
      s=s+str;
```

```
    return s;
}
// проверка времение срабатывания будильника
void isalarm() {
    // проход по всем будильникам
    for(int i=0;i<4;i++) {
        if(alarms[i].repeat==false) { //одноразовый
            if(tmYearToCalendar(datetime.Year)==alarms[i].Year &&
                datetime.Month==alarms[i].Month &&
                datetime.Day==alarms[i].Day &&
                datetime.Hour==alarms[i].Hour &&
                datetime.Minute==alarms[i].Minute &&
                datetime.Second<2) {
                tone(pinSpeaker,349,30000);
                alarmyes=true;
                Serial.println("ALARM OK");
            }
        }
        else {
            if(alarms[i].wdays[datetime.Wday]==1 &&
                datetime.Hour==alarms[i].Hour &&
                datetime.Minute==alarms[i].Minute &&
                datetime.Second<3) {
                tone(pinSpeaker,349,30000);
                alarmyes=true;
                Serial.println("ALARM OK1");
            }
        }
    }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_46_01.zip.**

Загружаем скетч на плату Arduino. На экране LCD Keypad shield видим часы с отображением времени, даты и дня недели. При срабатывании будильника слышим звуковой сигнал продолжительностью 30 секунд, прерываемый удержанием более 1 сек кнопки RIGHT на клавиатуре LCD Keypad shield.

Эксперимент 47.
Память EEPOM. Запись в EEPROM данных для будильников

В этом эксперименте рассмотрим работу с энергонезависимой памятью Arduino – EEPROM и создадим программу для загрузки и удаления из памяти EEPROM данных для будильников

В эксперименте мы будем использовать только плату Arduino+ WiFi и кабель USB для подключения к компьютеру. Переключатели на плате установлены следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

В эксперименте 46 мы создали часы на модуле DS3231 с функционалом будильника. Список будильников вписан в скетч. Но очень часто необходимо удалять старые будильники и добавлять новые. Где же в таком случае хранить информацию о будильниках? Микроконтроллер ATmega328 имеет на борту энергонезависимую память EEPROM объемом 1024 байта, которая не потеряет записанные в нее данные даже после отключения питания. Память типа EEPROM допускает несколько десятков тысяч циклов записи и стирания данных. Ее мы и будем использовать для сохранения данных о будильниках для часов.

Для работы с этой памятью в составе Arduino IDE имеется удобная библиотека EEPROM. Библиотека содержит две функции: чтения и записи в память данных. И чтение и запись происходят побайтно.

Структура, описывающая будильник из эксперимента 46:

```
struct ALARM{
  int Hour;         // час срабатывания будильника
  int Minute;       // минута срабатывания будильника
  int Day;          // день срабатывания будильника
  int Month;        // месяц срабатывания будильника
  int Year;         // год срабатывания будильника
  boolean repeat;   // true - повторять, false - одноразово
  int wdays[7];     // дни недели
};
```

Для хранения каждого будильника требуется 14 байт (на год – 2 байта, на остальные параметры по одному). Необходимо в памяти сохранять и общее количество будильников (нулевой байт памяти EEPROM). Тогда память EEPROM 1-1023 может использоваться для хранения информации о 73 будильниках!

Предварительно очистим память EEPROM, загрузив на Arduino скетч Файл → Примеры → EEPROM → eeprom_clear.

Добавление, удаление будильников будем производить по последовательному порту, отправляя соответствующие команды. Список команд:

№	Наименование	Формат
1	Добавить будильник	1;hour;minute;day;month;Year;repeat;s;m;t;w;t;f;s$
2	Удалить будильник	2;num$
3	Получить количество будильников	3$
4	Вывести список будильников	4$
5	Очистить	5$

Arduino собирает поступающие по последовательному порту данные, пока не придет символ '$'. Далее идет анализ поступивших данных и Arduino отправляет в последовательный порт ответ OK или ERROR и дополнительную информацию.

Содержимое скетча показано в листинге 47.1.

Листинг 47.1.

```
// подключение библиотек
#include <EEPROM.h>
// данные, пришедшие из последовательного порта
String inputString = "";
// строка пришла
boolean stringComplete = false;

void setup() {
   // запуск последовательного порта
   Serial.begin(9600);
   // резервирование 30 bytes для inputString:
   inputString.reserve(50);
   Serial.println("wait.....");
}

void loop() {
    // проверка прихода строки из последовательного порта
    if (stringComplete) {
       Serial.println(inputString);
       // обработка строки
```

```
        parse();
        // очистить строку
        inputString = "";
        stringComplete = false;
        Serial.println("wait.....");
    }
}
// получение данных по последовательному порту
void serialEvent() {
  boolean flag1=false;

  while (Serial.available() && flag1==false) {
    // получить байт:
    char inChar = (char)Serial.read();
    // добавление в строку
    inputString += inChar;
    if (inChar == '$') {
      stringComplete = true;
      flag1=true;
    }
  }
}
// парсинг поступивших данных
void parse() {
   int data[15];
   int index=0;
   String str1="";
   unsigned int count;

   // очистить
   for(int i=0;i<15;i++) {
      data[i]=0;
   }
   for(int i=0;i<inputString.length();i++) {
      if(inputString[i]==';' || inputString[i]=='$') {
         data[index]=str1.toInt();
         str1="";
         index++;
      }
      else {
         str1=str1+inputString[i];
      }
   }
   // выполнить команды
   count=EEPROM.read(0);
   switch(data[0]) {
      case 1:  Serial.println("OK");
               // добавление будильника
               count=EEPROM.read(0);
               EEPROM.write(count*14+1,data[1]);
               EEPROM.write(count*14+2,data[2]);
               EEPROM.write(count*14+3,data[3]);
               EEPROM.write(count*14+4,data[4]);
```

```
                EEPROM.write(count*14+5,highByte(data[5]));
                EEPROM.write(count*14+6,lowByte(data[5]));
                EEPROM.write(count*14+7,data[6]);
                EEPROM.write(count*14+8,data[7]);
                EEPROM.write(count*14+9,data[8]);
                EEPROM.write(count*14+10,data[9]);
                EEPROM.write(count*14+11,data[10]);
                EEPROM.write(count*14+12,data[11]);
                EEPROM.write(count*14+13,data[12]);
                EEPROM.write(count*14+14,data[13]);
                count++;
                EEPROM.write(0,count);
                Serial.println("add alarm");
                Serial.print("all alarms = ");Serial.println(count);
            break;
        case 2:  Serial.println("OK");
                // удалить если <= count
                if(data[1]<=count && data[1]>0) {
                   // перенести будильники от count до data[1]
                   for(int i=data[1];i<count;i++) {
                      for(int j=1;j<15;j++) {
                       EEPROM.write((i-1)*14+j,EEPROM.read(i*14+j));
                      }
                   }
                   count--;
                   EEPROM.write(0,count);
                   Serial.println("delete alarm");
                }
                else {
                   Serial.println("delete alarm ERROR !");
                }
                Serial.println("delete alarm");
                Serial.print("all alarms = ");Serial.println(count);
            break;
        case 3:  Serial.println("OK");
                Serial.print("all alarms = ");Serial.println(count);
            break;
        case 4:  Serial.println("OK");
                for(int i=0;i<count;i++) {
                   Serial.print(EEPROM.read(i*14+1));
                   Serial.print(" ");
                    Serial.print(EEPROM.read(i*14+2));Serial.print("
");
                    Serial.print(EEPROM.read(i*14+3));Serial.print("
");
                    Serial.print(EEPROM.read(i*14+4));Serial.print("
");
                   Serial.print((EEPROM.read(i*14+5)<<8)
                                +EEPROM.read(i*14+6));
                   Serial.print(" ");
                    Serial.print(EEPROM.read(i*14+7));Serial.print("
");
```

```
                Serial.print(EEPROM.read(i*14+8));Serial.print(" ");
                Serial.print(EEPROM.read(i*14+9));Serial.print(" ");
                  Serial.print(EEPROM.read(i*14+10));
                  Serial.print(" ");
                  Serial.print(EEPROM.read(i*14+11));
                  Serial.print(" ");
                  Serial.print(EEPROM.read(i*14+12));
                  Serial.print(" ");
                  Serial.print(EEPROM.read(i*14+13));
                  Serial.print(" ");
                  Serial.print(EEPROM.read(i*14+14));
                  Serial.print(" ");
                  Serial.println();
                }
                Serial.print("all alarms = ");Serial.println(count);
          break;
       case 5:  Serial.println("OK");
                count=0;
                EEPROM.write(0,count);
                Serial.println("clear alarms");
                Serial.print("all alarms = ");Serial.println(count);
          break;
       default:  Serial.println("ERROR");
          break;
    }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_47_01.zip.**

Загружаем скетч на плату Arduino, по последовательному порту добавляем и удаляем будильники, а также получаем список и количество будильников (рис. 47.1).

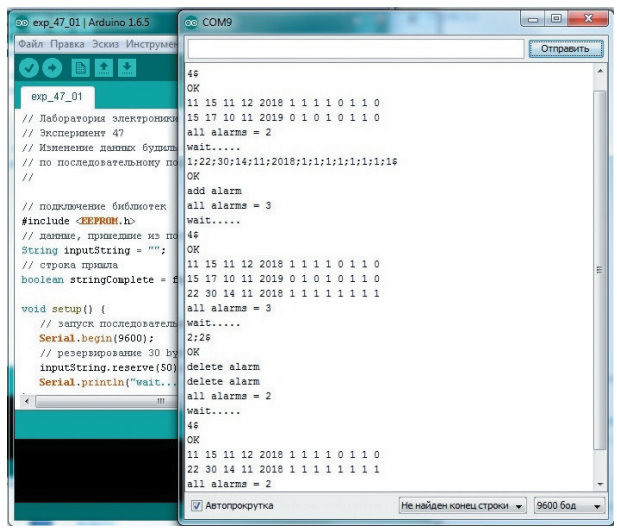


Рис. 47.1. Добавление, удаление будильников в EEPROM по последовательному порту.

Эксперимент 48.
Часы с будильниками на EEPROM

В этом эксперименте добавим часам на ЖК-дисплее LCD Keypad shield возможность управления будильниками, которые будут храниться в EEPROM

В эксперименте мы будем использовать компоненты и схему соединений из эксперимента 46. Только теперь наши часы получат возможность изменять будильники динамически – отправкой команд создания и удаления по последовательному порту, и хранить информацию о будильниках в памяти EEPROM. Команды добавления и удаления будильников и хранения их в EEPROM мы рассмотрели в эксперименте 47.

Приступим к написанию скетча. За основу берем скетч из листинга 46.1 (Эксперимент 46). Добавим подключение библиотеки EEPROM.h и переменные для работы со строками:

```
#include <EEPROM.h>
// данные, пришедшие из последовательного порта
String inputString = "";
// строка пришла
boolean stringComplete = false;
Добавим переменную для количества будильников:
int countalarms=0;
А также массив для хранения будильников:
ALARM alarms[73];
В процедуре setup() добавляем загрузку будильников в массив alarms[73]
из EEPROM – вызов процедуры getAlarms():
void getAlarms() {
   countalarms=EEPROM.read(0);
   for(int i=0;i<countalarms;i++) {
        alarms[i].Hour=EEPROM.read(i*14+1);
        alarms[i].Minute=EEPROM.read(i*14+2);
        alarms[i].Day=EEPROM.read(i*14+3);
        alarms[i].Month=EEPROM.read(i*14+4);
        alarms[i].Year=(EEPROM.read(i*14+5)<<8)+EEPROM.read(i*14+6);
        alarms[i].repeat=EEPROM.read(i*14+7);
        for(int j=0;j<7;j++) {
           alarms[i].wdays[j]=EEPROM.read(i*14+8+j);
        }
```

```
    }
}
В цикл loop()  добавляем получение данных по последовательному порту.
// проверка прихода строки из последовательного порта
if (stringComplete) {
   Serial.println(inputString);
   // обработка строки
   parse();
   // очистить строку
   inputString = "";
   stringComplete = false;
   Serial.println("wait.....");
}
```

И процедуры для получения и обработки полученных данных serialEvent() и parse() из листинга 47.1.

Полный скетч можно скачать на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_48_01.zip.**

Загружаем скетч на плату Arduino. На экране LCD Keypad shield видим часы с отображением времени, даты и дня недели. При срабатывании будильника слышим звуковой сигнал продолжительностью 30 секунд, прерываемый удержанием более 1 сек кнопки RIGHT на клавиатуре LCD Keypad shield, список будильников можно посмотреть отправкой команды "4$" по последовательному порту. По последовательному порту также можно добавить и удалить будильники (рис. 48.1).

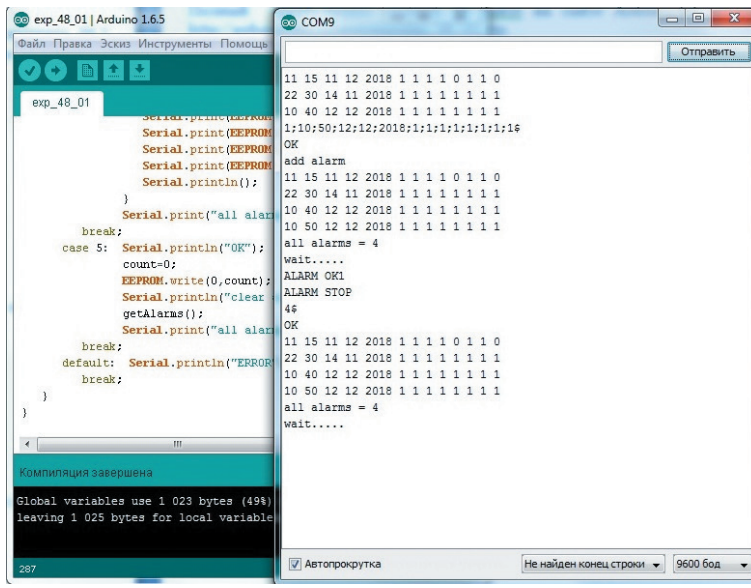


Рис. 48.1. Просмотр, добавление, удаление будильников в EEPROM по последовательному порту

Эксперимент 49.
Работа с SD-картой

В этом эксперименте рассмотрим использование SD-карты в качестве внешней памяти для сохранения данных в Arduino-проектах

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Модуль SD-карты – 1;
- ❑ SD-карта или microSD-карта;
- ❑ Провода MF – 6

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Платы Arduino оснащены сравнительно небольшой внутренней памятью. Объем памяти EEPROM для Arduino UNO всего 1 кБ и EEPROM. Этой памяти не хватит для записи больших объемов данных. Подключение SD карты к Arduino в качестве внешнего накопителя позволяет многократно увеличить место для хранения любой информации. Подключить SD-карту к Arduino напрямую не получится, но мы можем использовать, входящий в набор, дополнительный модуль. Модуль считывания SD-карт подключается по интерфейсу SPI. Схема подключения показана на рис. 49.1

Для работы с SD-картами в Arduino IDE есть встроенная библиотека SD. Загрузим на плату Arduino пример CardInfo (Файл → Примеры → SD → CardInfo) и откроем монитор последовательного порта. В последовательный порт выводится информация о файловой системе, емкости SD-карты и список файлов на SD-карте (рис. 49.2).

Arduino-библиотека SD содержит различные функции, с помощью которых можно управлять данными.

Некоторые функции класса SD:

- ❑ mkdir() – создает папку на карте памяти SD;
- ❑ rmdir() – удаляет папку с карты памяти SD, папка должна быть пустой;
- ❑ open() – открывает файл на карте памяти SD. Если файл не существует и открывается для записи, то он будет создан;

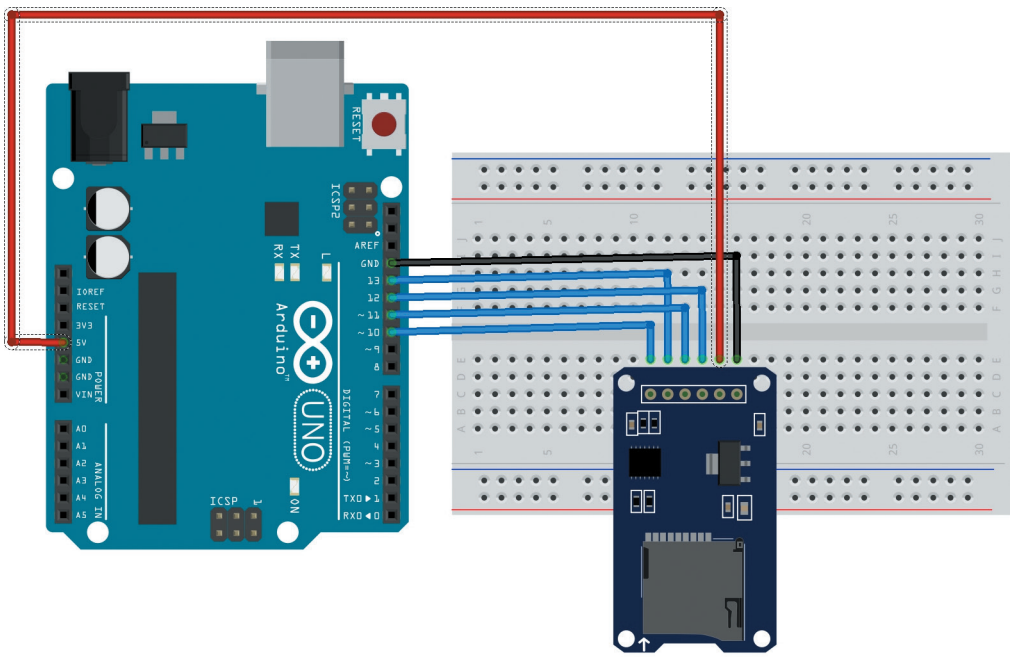


Рис. 49.1. Схема соединений для подключения модуля считывания SD-карт к Arduino

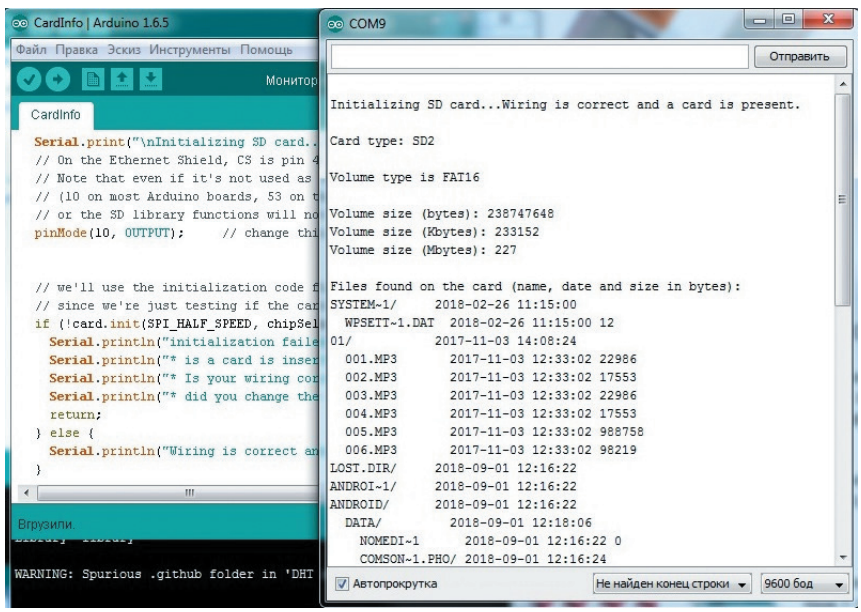


Рис. 49.2. Вывод информации с SD-карты (пример CardInfo).

- remove() – удаляет файл с карты памяти SD.

Для работы с файлами существует класс File. В него входят функции, которые предназначены для записи и чтения информации с карты:

- available() – проверяет наличие в файле байт, которые доступны для чтения. В ответ приходит количество места, которое доступно для чтения;
- close() – закрывает файл и сохраняет записанные на него данные на карту SD;
- flush() – гарантирует, что любые байты, записанные в файл, физически сохранятся на SD-карте (при закрытии файла это делается автоматически);
- name() – возвращает указатель на имя;
- peek() – считывает байт из файла из текущей позиции, не продвигаясь к следующей позиции;
- position() – возвращает текущую позицию в файле, куда следующий байт будет записываться или откуда считываться;
- print() – записывает данные в файл, который был открыт для записи;
- println() – записывает данные в файл, который был открыт для записи, данные дополняются символами перевода строки;
- seek() – устанавливает новую позицию в файле для чтения или записи байта;
- size() – возвращает размер файла в байтах;
- read() – возвращает байт из открытого файла;
- write() – записывает данные в открытый для записи файл;
- isDirectory() – проверяет, является ли текущий файл каталогом или нет;
- openNextFile() – возвращает имя следующего по позиции файла из каталога;
- rewindDirectory() – возвращает к первому файлу в директории.

Создадим пример создания файла и записи в него информации. Данные, получаемые по последовательному порту, будем записывать в файл serial.txt.

Содержимое скетча показано в листинге 49.1.

Листинг 49.1.

```
// подключение библиотек
#include <SPI.h>
#include <SD.h>
// экземпляр объекта
File file1;
// пин CS
int pinCS = 10;
// данные, пришедшие из последовательного порта
String inputString = "";
// строка пришла
boolean stringComplete = false;

void setup() {
    // запуск последовательного порта
```

```
    Serial.begin(9600);
   // резервирование 50 bytes для inputString:
   inputString.reserve(50);
   // инициализация карты
   Serial.print("Initializing SD card...");
   pinMode(10, OUTPUT);
   if (!SD.begin(pinCS)) {
      Serial.println("initialization failed!");
      return;
  }
  Serial.println("initialization done.");
}

void loop() {
    // проверка прихода строки из последовательного порта
    if (stringComplete) {
       Serial.println(inputString);
       // запись строки в файл
       // открыть файл для записи
       file1 = SD.open("serial.txt", FILE_WRITE);
       if (file1) {
          Serial.println("Writing to serial.txt");
          file1.print(inputString);
          // закрыть файл
          file1.close();
       }
       else {
           Serial.println("Error open serial.txt");
       }
       // Чтение содержимого файла
       file1 = SD.open("serial.txt");
       if (file1) {
          Serial.println("Writing to serial.txt");
          // чтение файла:
          while (file1.available()) {
             Serial.write(file1.read());
          }
          // закрыть файл
          file1.close();
       }
       else {
           Serial.println("Error open serial.txt");
       }
       // очистить строку
       inputString = "";
       stringComplete = false;
    }
```

```
}

//
void serialEvent() {
  boolean flag1=false;

  while (Serial.available() && flag1==false) {
    // получить байт:
    char inChar = (char)Serial.read();
    if (inChar == '\n') {
      stringComplete = true;
      flag1=true;
    }
    // добавление в строку
    inputString += inChar;
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/ exp_49_01.zip.**

Загружаем скетч на плату Arduino и загружаем по последовательному порту данные в файл serial.txt построчно (рис. 49.3).

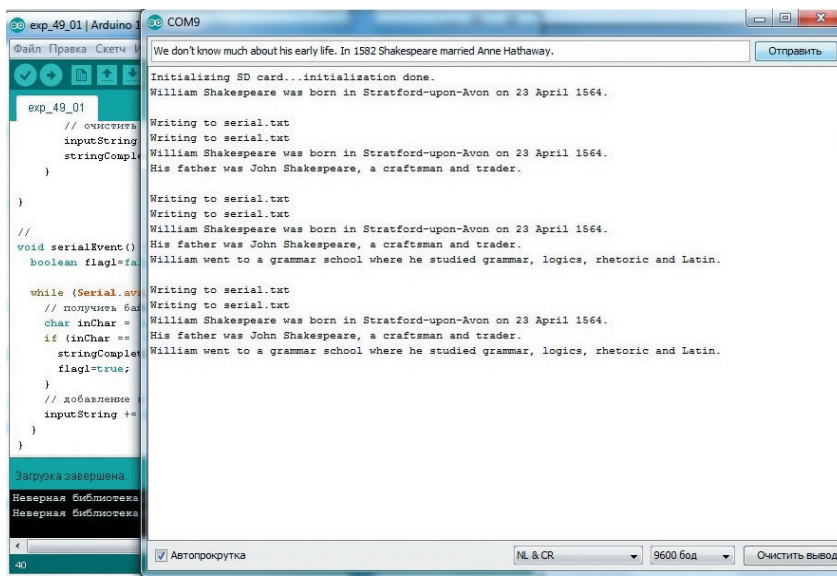


Рис. 49.3. Загрузка данных по последовательному порту в файл serial.txt

Эксперимент 50.
Сохранение данных метеостанции на SD-карте

В этом эксперименте мы рассмотрим сохранение данных домашней метеостанции на SD-карте

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Датчик BMP280 – 1;
- ❑ Датчик DHT11 – 1;
- ❑ LCD Keypad shield – 1;
- ❑ Модуль DS3231 – 1;
- ❑ Модуль SD-карты – 1;
- ❑ SD-карта или microSD-карта;
- ❑ Провода MM – 15.

Переключатели на плате Arduino+WiFi установите следующим образом

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

В эксперименте 41 мы создали домашнюю метеостанцию на датчиках BMP280 и DHT11. Данные с датчиков мы выводили на экран LCD Keypad shield. Однако часто необходимо знать, как менялись данные с датчиков во времени. В этом эксперименте мы будем сохранять данные с датчиков в файлах на SD-карте. Для фиксации времени получения данных будем использовать модуль часов реального времени DS3231.

Схема соединений показана на рис. 50.1.

Приступим к написанию скетча. Данные будем получать раз в 30 секунд, и выводить на экран.

Каждые 60 секунд получаем текущее время (в формате H:s), показания влажности, температуры и атмосферного давления. Формируем строку для записи в файл текущего дня (формат y-m-d.txt). Выводим строку в монитор последовательного порта, а данные на экран LCD Keypad shield.

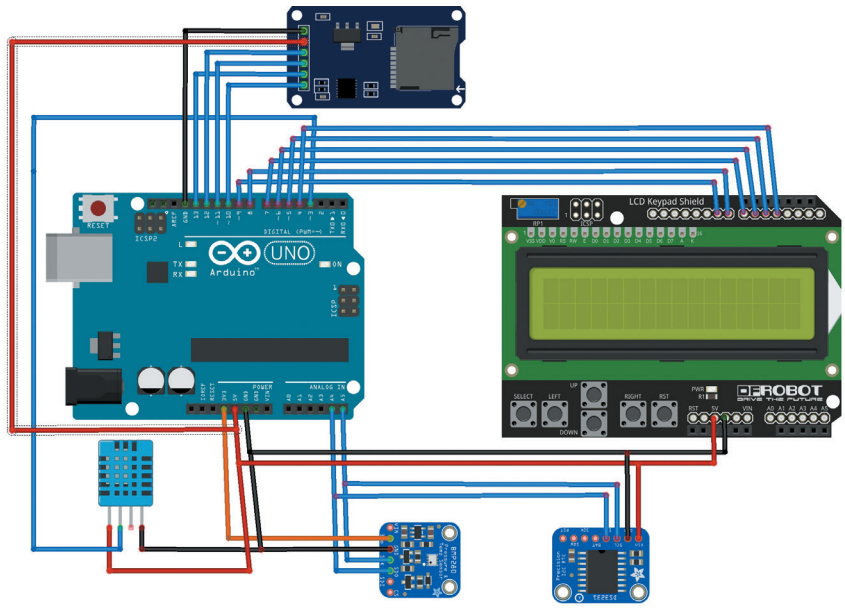


Рис. 50.1. Схема соединений для метеостанции на датчиках BMP180 и DHT11 сохранение данных на SD-карте

Содержимое скетча показано в листинге 50.1.

Листинг 50.1.

```
// подключение библиотек
#include <LiquidCrystal.h>
#include "DHT.h"
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <DS1307RTC.h>
#include <Time.h>
#include <TimeLib.h>
#include <SPI.h>
#include <SD.h>
// пин для подключения датчика DHT
#define DHTPIN 12
// тип датчика DHT
#define DHTTYPE DHT11   // DHT 11
// создание экземпляров
Adafruit_BMP280 bmp;
LiquidCrystal lcd(8,9,4,5,6,7);
DHT dht(DHTPIN, DHTTYPE);
// для опроса
unsigned long millist=0;
// экземпляр объекта
```

```
File file1;
// пин CS
int pinCS = 10;
//
tmElements_t datetime;
// имя файла текущего дня ymd
String sfilename;
char filename[20];
String record="";

void setup()
{
  // подключение последовательного порта
  Serial.begin(9600);
  // инициализация дисплея
  lcd.begin(16,2);
  // очистить
  lcd.clear();
  // инициализация карты
  Serial.print("Initializing SD card...");
  pinMode(10, OUTPUT);
  if (!SD.begin(pinCS)) {
     Serial.println("initialization failed!");
     return;
  }
  Serial.println("initialization done.");
  // запуск датчиков DHT,BMP20
  dht.begin();
  bmp.begin();
}

void loop() {
  if(millis()-millist>=60000)
     {
     // получение данных
     int h = dht.readHumidity();
     int t = bmp.readTemperature();
     int p = bmp.readPressure();
     // вывод на дисплей
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("H=");
     lcd.print(h);lcd.print("%");
     lcd.setCursor(7,0);
     lcd.print("T=");
     lcd.print(t);lcd.print("*C");
     lcd.setCursor(0,1);
     lcd.print("P= ");
     lcd.print(p);lcd.print("Pa");
     // получение имени файла
     sfilename=get_file_name();
     sfilename.toCharArray(filename,20);
     //
     file1 = SD.open(filename, FILE_WRITE);
     // получить время H:m
     // создать запись для файла
     record=get_time();
```

```
      record+=" ";
      record+=String(h);
      record+=" ";
      record+=String(t);
      record+=" ";
      record+=String(p);
      Serial.println(record);
      file1.println(record);
      file1.close();
      //
      millist=millis();
      }
}

// получение времени дня
String get_time()
 {
  String time1;
  RTC.read(datetime);
  if(datetime.Hour<10)
   time1="0"+String(datetime.Hour,DEC);
  else
   time1=String(datetime.Hour,DEC);
  if(datetime.Minute<10)
   time1+=":0"+String(datetime.Minute,DEC);
  else
   time1+=":"+String(datetime.Minute,DEC);
  return time1;
 }
// получение имени файла для текущего дня
String get_file_name()
 {
  String filename1;
  RTC.read(datetime);
  filename1+=String(tmYearToCalendar(datetime.Year),DEC);
  if(datetime.Month<10)
   filename1+="-0"+String(datetime.Month,DEC);
  else
   filename1+="-"+String(datetime.Month,DEC);
  if(datetime.Day<10)
   filename1+="-0"+String(datetime.Day,DEC);
  else
   filename1+="-"+String(datetime.Day,DEC);
  filename1+=".txt";
  return filename1;
 }
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_50_01.zip.**

Загружаем скетч на плату Arduino и на экране дисплея видим показания атмосферного давления и температуры с датчика BMP280 и относительной влажности воздуха с датчика DHT11. Через некоторое время проверяем наличие файла текущего дня с показаниями датчиков.

Эксперимент 51.
Подключение исполнительных устройств.

В этом эксперименте рассмотрим подключение электромеханических реле, которые позволяют управлять с Arduino различными электроприборами

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Релейный модуль – 1;
- ❑ LCD Keypad shield – 1;
- ❑ Светодиод – 2;
- ❑ Резистор 220 Ом – 2;
- ❑ Провода MM – 5;
- ❑ Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Для управления электроприборами, которые питаются от бытовой электросети, мы пользуемся различными клавишными выключателями и тумблерами. Чтобы управлять такими электроприборами с помощью микроконтроллера существует специальный тип выключателей — электромеханические реле. В наборе имеется релейный модуль, который содержит два таких реле и позволяет Arduino управлять двумя электроприборами.

В релейных модулях обычно используется n-канальное управление. При таком управлении реле включается подачей на вывод Arduino низкого уровня LOW, а выключается подачей высокого уровня HIGH.

Создадим проект включения/выключения двух реле с помощью кнопок. Будем использовать кнопки от LCD Keypad shield. Схема соединений для подключения релейного модуля к плате Arduino показана на рис. 51.1.

По нажатию кнопки переключаем состояние реле согласно таблице.

Светодиоды служат для индикации состояния реле. Содержимое скетча представлено в листинге 51.1.

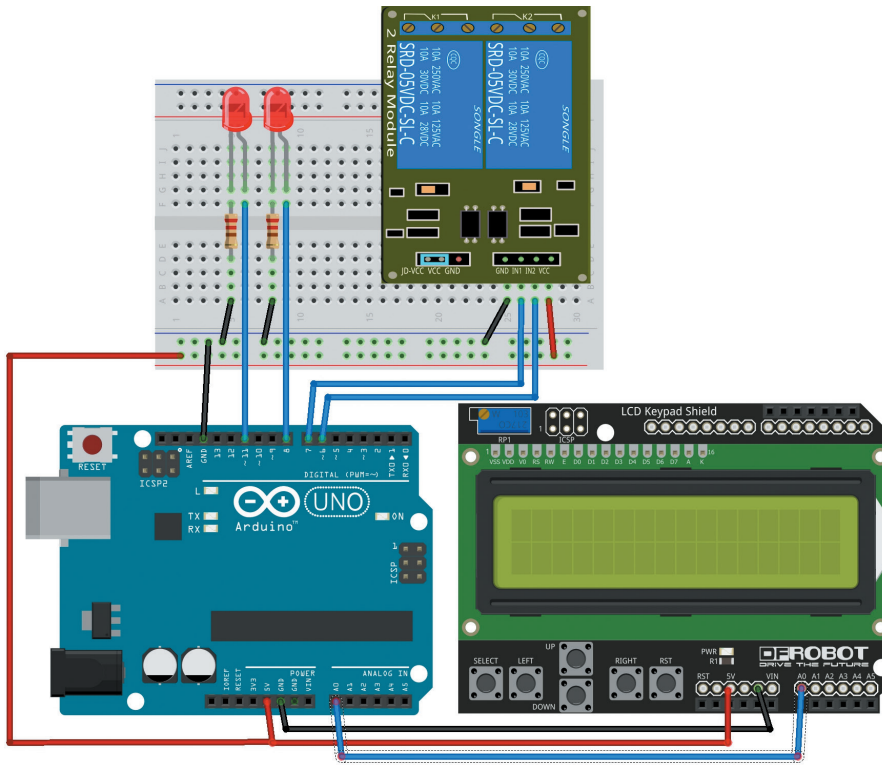


Рис. 51.1. Схема соединений для подключения релейного модуля к Arduino

Листинг 51.1.

```
// переменная для хранения значения A0
int valA0 = 0;
// контакты подключения светодиодов
const int pinLed1=11;
const int pinLed2=8;
// контакты подключения реле
const int pinRelay1=7;
const int pinRelay2=6;
// состояние реле
boolean relayOn1=false;
boolean relayOn2=false;

void setup() {
   // конфигурация пинов
   pinMode(pinLed1,OUTPUT);
   pinMode(pinLed2,OUTPUT);
   pinMode(pinRelay1,OUTPUT);
   pinMode(pinRelay2,OUTPUT);
   // начальная установка
   digitalWrite(pinLed1,LOW);
   digitalWrite(pinLed2,LOW);
```

	Relay1	Relay2
SELECT	OFF	
LEFT	ON	
DOWN		OFF
UP		ON

```
    digitalWrite(pinRelay1,HIGH);
    digitalWrite(pinRelay2,HIGH);
}

void loop() {
    // чтение данных A0
    valA0 = analogRead(A0);
    // определение нажатия кнопки
    if(valA0<100) { // RIGHT
        ;
    }
    else if(valA0<200) { // UP
        relayOn1=true;
    }
    else if(valA0<400) { // DOWN
        relayOn1=false;
    }
    else if(valA0<600) { // LEFT
        relayOn2=true;
    }
    else if(valA0<800) { // SELECT
        relayOn2=false;
    }
    // установка светодиодов
    digitalWrite(pinLed1,relayOn1);
    digitalWrite(pinLed2,relayOn2);
    // установка реле
    digitalWrite(pinRelay1,relayOn1);
    digitalWrite(pinRelay2,relayOn2);

}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_51_01.zip.**

Загружаем скетч на плату Arduino. При переключении реле слышен щелчок. Теперь остается подсоединить к реле реальные электроприборы.

ВНИМАНИЕ!

Работа с высоким напряжением опасна для вашего здоровья и жизни. На плате существуют области, прикосновение к которым приведет к поражению электрическим током. Это винты контактных колодок и места пайки выводов контактных колодок и реле. Не работайте с платой, если она подключена к бытовой сети. Для готового устройства используйте изолированный корпус.

Если вы сомневаетесь как подключить к реле электроприбор, работающий от общей сети 220 В и у вас есть сомнения, вопросы на тему того как это делается, остановитесь: вы можете устроить пожар или убить себя. Убедитесь, что у вас в голове кристальное понимание принципа работы устройства и опасностей, которые связаны с высоким напряжением

Эксперимент 52.
Подключение 4-фазного шагового двигателя

В этом эксперименте рассмотрим подключение к плате Arduino шагового двигателя

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Шаговый двигатель 28BYJ-48 – 1;
- ❑ Драйвер двигателя на микросхеме ULN2003 – 1;
- ❑ Провода MM – 4;
- ❑ Провода MF – 4

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Шаговые двигатели применяют в механических системах точного позиционирования – ЧПУ станках, 3d-принтерах, принтерах, роботах-манипуляторах. Шаговые двигатели преобразуют электрические импульсы в перемещение вала на определенный угол. Минимально возможный угол перемещения шагового двигателя, называется шагом.

В наборе присутствует шаговый двигатель 28BYJ-48, а также драйвер двигателя на микросхеме ULN2003, необходимый для подключения шагового двигателя к плате Arduino.

Принципиальная схема шагового двигателя 28BYJ-48 приведена на рис. 52.1.

Для управления шаговым двигателем 28BYJ-48 используют один из двух режимов подключения.

- ❑ полношаговый режим – 4 ступени импульсов на 1 шаг (рис. 52.2);
- ❑ полушаговый режим – 8 ступеней импульсов на 1 шаг (рис. 52.2).

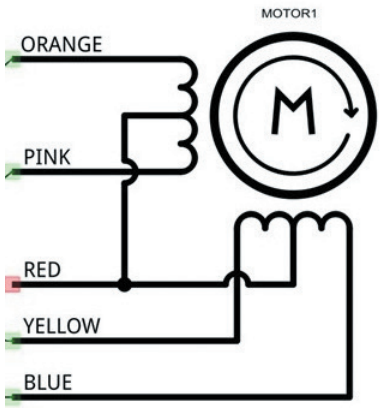


Рис. 52.1. Принципиальная схема шагового двигателя 28BYJ-48

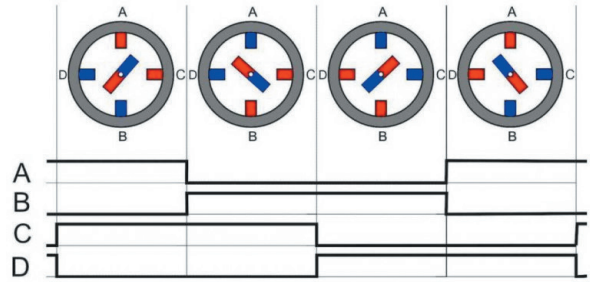


Рис. 52.2. Полношаговый режим.

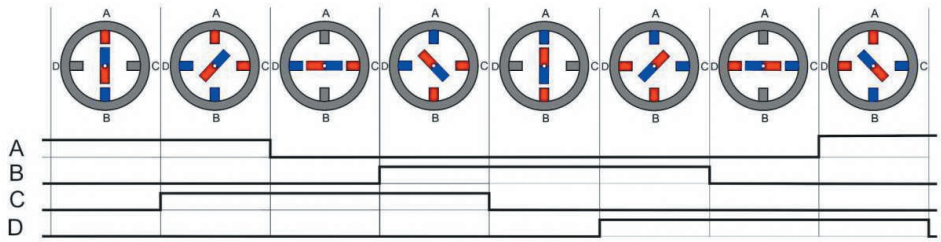


Рис. 52.3. Полушаговый режим

Полношаговый режим:

Контакт	Такты			
	1	2	3	4
A - orange	1	0	0	1
B - yellow	0	1	1	0
C - pink	1	1	0	0
D - blue	0	0	1	1

Полушаговый режим:

Контакт	Такты							
	1	2	3	4	5	6	7	8
A-orange	1	1	0	0	0	0	0	1
B-yellow	0	0	0	1	1	1	0	0
C-pink	0	1	1	1	0	0	0	0
D-blue	0	0	0	0	0	1	1	1

Драйвер двигателя состоит из 7 пар транзисторов Дарлингтона и является усилителем. Выводы IN1 – IN7 предназначены для подключения к микроконтроллеру, GND и VCC – для питания шагового двигателя. Схема соединений для подключения драйвера к плате Arduino показана на рисунке 52.4.

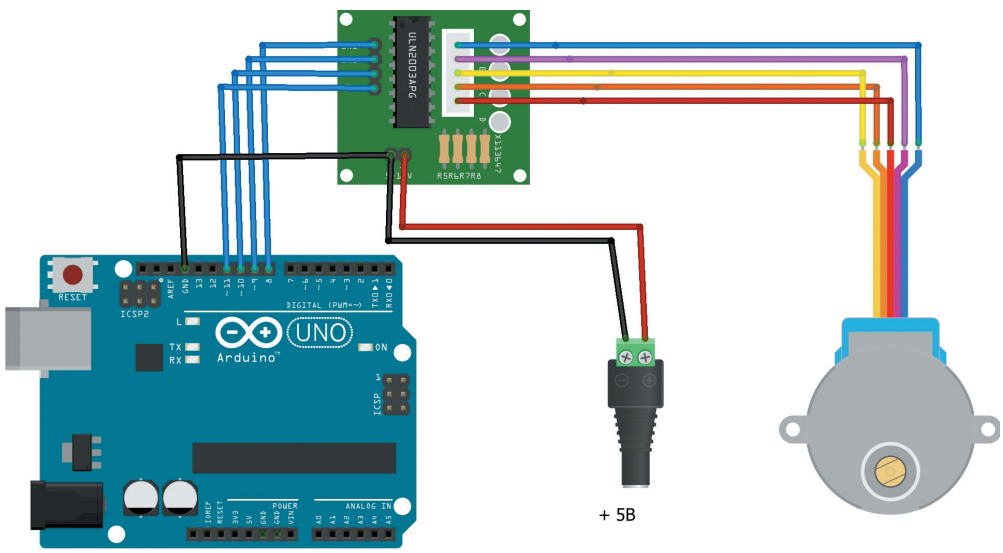


Рис. 52.4. Схема соединений для подключения драйвера к плате Arduino

Для управления шаговыми двигателями в Arduino IDE есть встроенная библиотека – Stepper. Данная библиотека осуществляет только полношаговый режим коммутации. Поэтому будем использовать Arduino-библиотеку Accel Stepper, которую можно скачать с сайта Arduino-kit по ссылке http://arduino-kit.ru/scetches/AccelStepper.zip. Accel Stepper. Библиотека Accel Stepper поддерживает не только равномерное движение, но и замедление/ускорение двигателя и работу

с несколькими двигателями. Установим библиотеку и напишем скетч циклического движения шагового двигателя, сначала на 1 оборот в направлении против часовой стрелки, затем на 1 оборот в направлении часовой.

Содержимое скетча показано в листинге 52.1.

Листинг 52.1.

```
// подключение библиотеки
#include<AccelStepper.h>
// контакты подключения
#define IN1 8
#define IN2 9
#define IN3 10
#define IN4 11
// создаем экземпляр AccelStepper
AccelStepper motor(8, IN1, IN3, IN2, IN4);

void setup(){
  motor.setMaxSpeed(900.0);
  motor.setAcceleration(100.0);
  motor.setSpeed(200);
  // перемещение на 2000 шагов
  motor.moveTo(2000);
}

void loop(){
  // Изменяем направление, если пройдено заданное число шагов
  if(motor.distanceToGo()==0)
    motor.moveTo(-motor.currentPosition());
  motor.run();
}
```

Скачать данный скетч можно на сайте на сайте Arduino-kit по ссылке **http://arduino-kit.ru/scetches/exp_52_01.zip.**

Загружаем скетч на плату Arduino и наблюдаем за движением шагового двигателя.

Эксперимент 53.
Управление скоростью и направлением движения 4-фазного шагового двигателя с LCD Keypad shield

В этом эксперименте организуем управление шаговым двигателем с помощью кнопок LCD Keypad shield

В эксперименте мы будем использовать следующие компоненты:
- Плата Arduino UNO – 1;
- Кабель USB – 1;
- Плата прототипирования – 1;
- Шаговый двигатель 28BYJ-48 – 1;
- Драйвер двигателя на микросхеме ULN2003 – 1;
- LCD Keypad shield – 1;
- Провода MM – 13;
- Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

В этом эксперименте создадим систему управления скоростью и направлением движения шагового двигателя. Выбирать скорость и направление движения шагового двигателя будем с помощью кнопок LCD Keypad shield. На экран будем выводить информацию.

Схема соединений показана на рисунке 53.1.

Для кнопок UP и DOWN необходимо реализовать задержку действия клавиши и автоповтор:

```
if(millis()-millisDown>500) {
        // DOWN - скорость уменьшить
        speed=max(minSpeed,speed-stepspeed);
        motor.setSpeed(speed*dir);
        lcd.setCursor(6,1);lcd.print("     ");
```

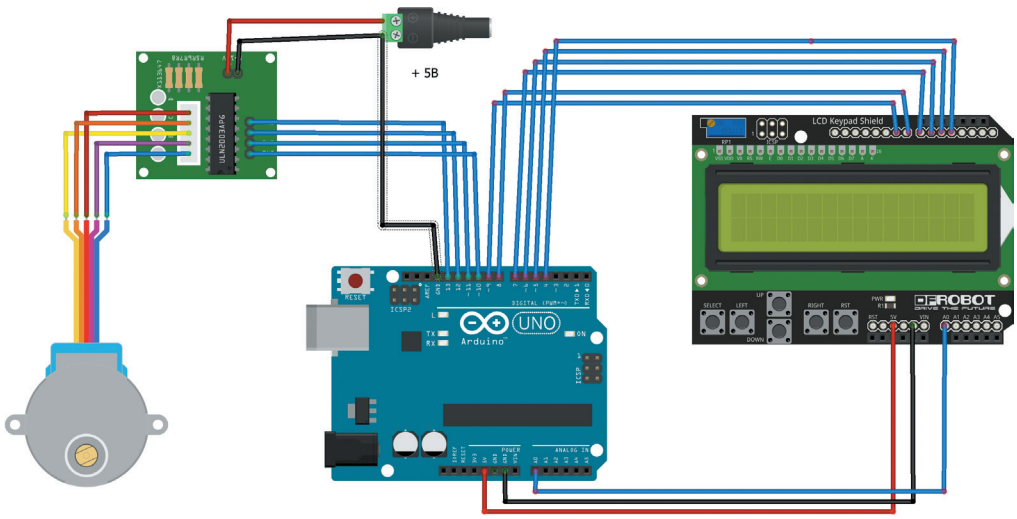


Рис. 53.1. Схема соединений для управления шаговым двигателем

Кнопка	Назначение
RIGHT	Движение по часовой стрелке
LEFT	Движение против часовой стрелки
UP	Увеличение скорости (автоповтор для кнопки 500 мсек)
DOWN	Уменьшение скорости (автоповтор для кнопки 500 мсек)
SELECT	Останов двигателя

```
        lcd.setCursor(6,1);lcd.print(speed);
        millisDown=millis();
    }
```

А также нижнюю и верхнюю границу для скорости, и шаг изменения скорости:

```
int minSpeed=50;
int maxSpeed=900;
int stepspeed=50;
```

Содержимое скетча показано в листинге 53.1.

Листинг 53.1.

```
// подключение библиотек
#include <LiquidCrystal.h>
#include<AccelStepper.h>
// контакты подключения
#define IN1 10
```

```
#define IN2 11
#define IN3 12
#define IN4 13
// создаем экземпляров объектов
AccelStepper motor(8, IN1, IN3, IN2, IN4);
LiquidCrystal lcd(8,9,4,5,6,7);
// переменные
int minSpeed=50;
int maxSpeed=900;
int speed=300;
int stepspeed=50;
int dir=0;
// для автоповтора
unsigned long millisUp=0;
unsigned long millisDown=0;
// переменная для хранения значения A0
int valA0 = 0;

void setup(){
  lcd.begin(16,2);
  // очистить
  lcd.clear();
  // запуск датчика DHT
  lcd.setCursor(0,0);
  lcd.print("dir   ");
  lcd.print("0    ");
  lcd.setCursor(0,1);
  lcd.print("speed ");
  lcd.print(speed);
  // начальные установки
  motor.setMaxSpeed(900.0);
  motor.setAcceleration(100.0);
  motor.setSpeed(200);
}

void loop(){
   // чтение данных A0
   valA0 = analogRead(A0);
   // определение нажатия кнопок
   if(valA0<80) { // RIGHT
      dir=-1;
      lcd.setCursor(6,0);
      lcd.print("right");
      motor.setSpeed(speed*dir);
   }
   else if(valA0<200) {
      if(millis()-millisUp>500) {
         // UP - скорость увеличить
```

```
            speed=min(maxSpeed,speed+stepspeed);
            motor.setSpeed(speed*dir);
            lcd.setCursor(6,1);lcd.print("     ");
            lcd.setCursor(6,1);lcd.print(speed);
            millisUp=millis();
         }
      }
      else if(valA0<400) {
         if(millis()-millisDown>500) {
            // DOWN - скорость уменьшить
            speed=max(minSpeed,speed-stepspeed);
            motor.setSpeed(speed*dir);
            lcd.setCursor(6,1);lcd.print("     ");
            lcd.setCursor(6,1);lcd.print(speed);
            millisDown=millis();
         }
      }
      else if(valA0<600) { // LEFT
         dir=1;
         lcd.setCursor(6,0);
         lcd.print("left ");
         motor.setSpeed(speed*dir);
      }
      else if(valA0<800) { // SELECT - останов
         dir=0;
         lcd.setCursor(6,0);
         lcd.print("0    ");
      }
     // движение мотора
     if(dir!=0)
        motor.runSpeed();
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_53_01.zip.**

Загружаем скетч на плату Arduino и управляем скоростью и направлением движения шагового двигателя, данные выводятся на экран.

Эксперимент 54.
Беспроводная связь по инфракрасному каналу

В этом эксперименте мы рассмотрим беспроводную передачу данных по инфракрасному каналу

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ ИК-приемик – 1;
- ❑ ИК пульт – 1;
- ❑ Провода MM – 3.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Шаговые двигатели применяют в механических системах точного Инфракрасный пульт дистанционного управления — один из самых простых способов взаимодействия с электронными приборами. Практически в каждом доме есть несколько таких устройств: телевизор, музыкальный центр, видеоплеер, кондиционер. Но если мы хотим использовать инфракрасный пульт для дистанционного управление устройством на плате Arduino, нам понадобится еще и инфракрасный приемник. В составе набора имеется ИК приемник на плате с подтягивающими резисторами, индикатором питания, разъемами и ИК пульт, имеющий 17 функциональных клавиш. Дальность передачи сигнала пультом до 8 м. Работает пульт от батарейки 3В типа CR2025.

Реализуем на плате Arduino получение данных, отправляемых с пульта. Схема соединений для подключения ИК приемника к плате Arduino показана на рисунке 54.1.

Для получения кодов, отправляемых с пульта, будем использовать Arduino-библиотеку IRRemote, которую можно скачать с сайта Arduino-kit по ссылке **https://arduino-kit.ru/scetches/IRRemote.zip.** Эта библиотека умеет не только принимать и декодировать сигналы, но и работать с ИК передатчиком. Загрузим

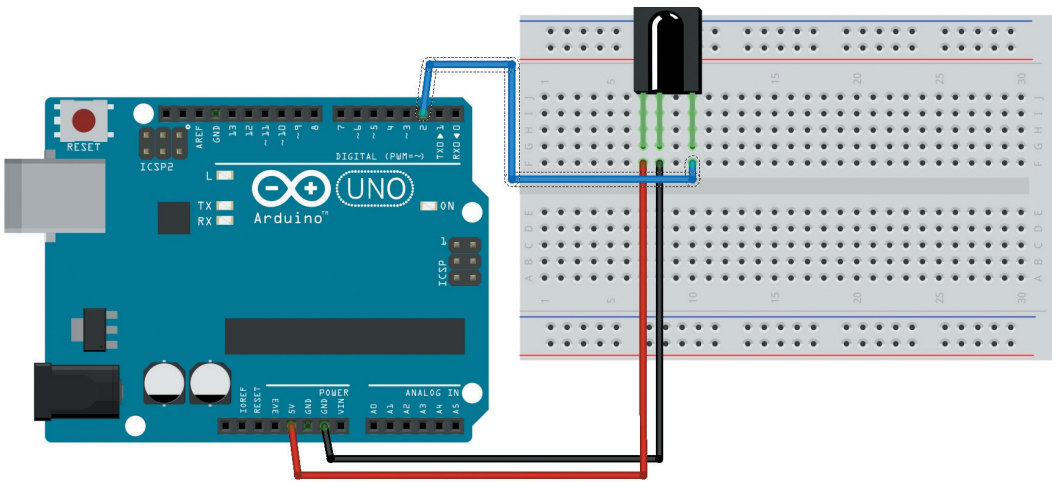


Рис. 54.1. Схема соединений для подключения ИК приемника.

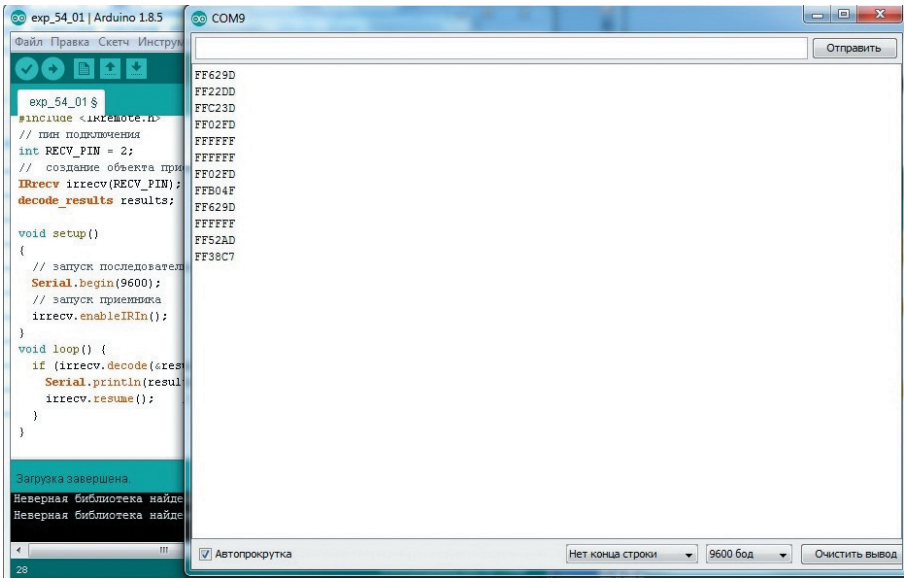


Рис. 54.2. Получаемые с ИК пульта коды клавиш.

на плату Arduino скетч для получения данных с пульта и выдачи кода в последовательный порт Arduino.

Содержимое скетча показано в листинге 54.1.

Листинг 54.1.

```
// подключение библиотеки
#include <IRremote.h>
// пин подключения
int RECV_PIN = 2;
//  создание объекта приемника
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  // запуск последовательного порта
  Serial.begin(9600);
  // запуск приемника
  irrecv.enableIRIn();
}
void loop() {
  // получение кода
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume();    // ждать следующее нажатие
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_54_01.zip.

После загрузки скетча открываем монитор последовательного порта и смотрим коды, приходящие при нажатии кнопок пульта (рис. 54.2). Код FFFFFFFF означает повтор последней нажатой кнопки, т.е. кнопка зажата и не отпущена.

В таблице представлены коды, отправляемые при нажатии клавиш пульта.

Кнопка пульта	Код	Кнопка пульта	Код
↑	0xFF629D	5	0xFF18E7
←	0xFF22DD	6	0xFF7A85
→	0xFFC23D	7	0xFF10EF
↓	0xFFA857	8	0xFF38C7
OK	0xFF02FD	9	0xFF5AA5
1	0xFF6897	0	0xFF4AB5
2	0xFF9867	*	0xFF42BD
3	0xFFB04F	#	0xFF52AD
4	0xFF30CF		

Эксперимент 55.
Управление скоростью и направлением движения 4-фазного шагового двигателя по ИК каналу

В этом эксперименте организуем управление шаговым двигателем с ИК пульта

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Шаговый двигатель 28BYJ-48 – 1;
- ❑ Драйвер двигателя на микросхеме ULN2003 – 1;
- ❑ LCD Keypad shield – 1;
- ❑ ИК-приемик – 1;
- ❑ ИК пульт – 1;
- ❑ Провода MM – 13;
- ❑ Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

В эксперименте 53 мы создали систему управления скоростью и направлением вращения шагового двигателя с помощью кнопок LCD Keypad shield. В этом эксперименте мы будем управлять шаговым двигателем с ИК пульта. Выводить информацию будем также на экран LCD Keypad shield.

Схема соединений показана на рисунке 55.1.

Содержимое скетча показано в листинге 55.1.

Листинг 55.1.

```
// подключение библиотек
#include <LiquidCrystal.h>
#include<AccelStepper.h>
```

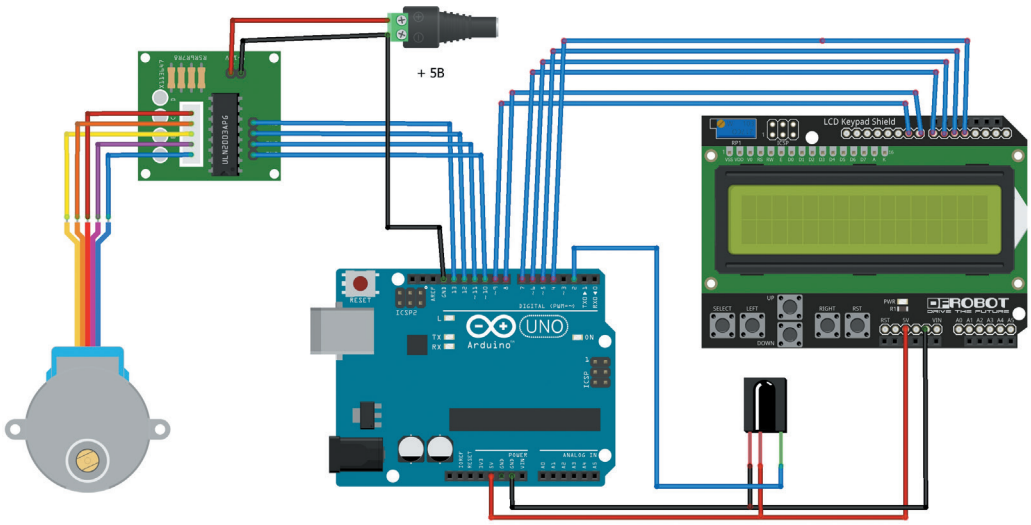


Рис. 55.1. Схема соединений для управления шаговым двигателем.

Назначение кнопок пульта

Кнопка пульта	Назначение	Код кнопки
→	Движение по часовой стрелке	0xFFC23D
←	Движение против часовой стрелки	0xFF22DD
↑	Увеличение скорости	0xFF629D
↓	Уменьшение скорости	0xFFA857
OK	Останов двигателя	0xFF02FD
1	0xFF6897	0
2	0xFF9867	*
3	0xFFB04F	#
4	0xFF30CF	

```
#include <IRremote.h>
// контакты подключения
#define IN1 10
#define IN2 11
#define IN3 12
#define IN4 13
int RECV_PIN = 2;
// создаем экземпляров объектов
AccelStepper motor(8, IN1, IN3, IN2, IN4);
```

```
LiquidCrystal lcd(8,9,4,5,6,7);
IRrecv irrecv(RECV_PIN);
decode_results results;
// переменные
int minSpeed=50;
int maxSpeed=900;
int speed=300;
int stepspeed=50;
int dir=0;
//


void setup(){
  lcd.begin(16,2);
  // очистить
  lcd.clear();
  // запуск датчика DHT
  lcd.setCursor(0,0);
  lcd.print("dir   ");
  lcd.print("0    ");
  lcd.setCursor(0,1);
  lcd.print("speed ");
  lcd.print(speed);
  // начальные установки
  motor.setMaxSpeed(900.0);
  motor.setAcceleration(100.0);
  motor.setSpeed(200);
  // запуск приемника
  irrecv.enableIRIn();
}

void loop(){
   // получение кода с пульта
   if (irrecv.decode(&results)) {
      Serial.println(results.value, HEX);
      // проверка на коды управления
      switch(results.value) {
         case 0xFFC23D: // вправо (по часовой стрелке)
                dir=-1;
                lcd.setCursor(6,0);
                lcd.print("right");
                motor.setSpeed(speed*dir);
            break;
         case 0xFF22DD: // влево (против часовой стрелки)
```

```
                dir=1;
                lcd.setCursor(6,0);
                lcd.print("left ");
                motor.setSpeed(speed*dir);
            break;
          case 0xFF629D: // увеличить
                speed=min(maxSpeed,speed+stepspeed);
                motor.setSpeed(speed*dir);
                lcd.setCursor(6,1);lcd.print("     ");
                lcd.setCursor(6,1);lcd.print(speed);
            break;
          case 0xFFA857: // уменьшить)
                speed=max(minSpeed,speed-stepspeed);
                motor.setSpeed(speed*dir);
                lcd.setCursor(6,1);lcd.print("     ");
            break;
          case 0xFF02FD: // стоп
                dir=0;
                lcd.setCursor(6,0);
                lcd.print("0    ");
            break;
          default:
            break;
        }
        irrecv.resume();
     }
    // движение мотора
    if(dir!=0)
       motor.runSpeed();
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_55_01.zip.**

Загружаем скетч на плату Arduino и управляем с пульта скоростью и направлением вращения шагового двигателя. Данные выводятся на экран.

Эксперимент 56.
Ультразвуковой датчик расстояния HC-SR04

В этом эксперименте рассмотрим ультразвуковой датчик расстояния HC-SR04

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Ультразвуковой датчик HC-SR04;
- ❑ Провода MM – 13;
- ❑ Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Ультразвуковой дальномер модуль HC-SR04 — это помещенные на одну плату приемник и передатчик ультразвукового сигнала. Принцип действия HC-SR04 основан на явлении эхолокации. Излучатель формирует акустический сигнал, который отразившись от преграды, возвращается к датчику и регистрируется приемником. Зная скорость распространения ультразвука в воздухе и время запаздывания между излученным и принятым сигналом, легко рассчитать расстояние до акустической преграды. Диапазон дальности его измерения составляет от 2 до 400 см. В отличие от инфракрасных дальномеров на ультразвуковой датчик HC-SR04 не влияют источники света или цвет препятствия. Могут возникнуть затруднения при определении расстояния до пушистых или тонких объектов.

Кроме приемника и передатчика на плате находится необходимая обвязка. Подсоединим датчик к плате Arduino. Схема соединений показана на рисунке 56.1.

Последовательность действий по измерению расстояния следующая:

1. подаем импульс продолжительностью 10 мкс на вывод Trig;

2. на плате модуля входной импульс преобразуется в 8 импульсов частотой 40 кГц и посылается через излучатель T;

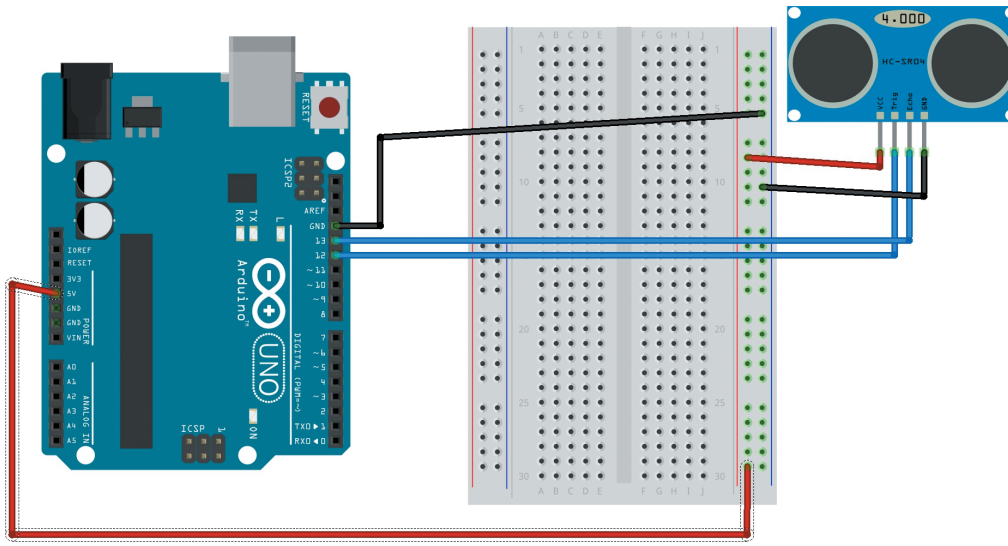


Рис. 56.1. Схема соединений для подключения датчика HC-SR04

3. дойдя до препятствия, посланные импульсы отражаются и принимаются приемником R, в результате получаем выходной сигнал на выводе Echo.

4. На стороне контроллера переводим полученный сигнал в расстояние по формуле:

ширина импульса (мкс) / 58 = дистанция (см).

Для работы Arduino с датчиком HC-SR04 есть готовая библиотека Ultrasonic, которую можно скачать с сайта Arduino-kit по ссылке https://arduino-kit.ru/scetches/Ultrasonic.zip. Загрузим на плату Arduino скетч для получения данных с датчика HC-SR04 и выдачи результата в последовательный порт Arduino.

Содержимое скетча показано в листинге 56.1.

Листинг 56.1.

```
// константы для выводов
#define PIN_TRIG 12
#define PIN_ECHO 13
// подключение библиотеки для HC SR04
#include "Ultrasonic.h"
// создание объекта Ultrasonic
// Trig - 12, Echo - 13
Ultrasonic ultrasonic(PIN_TRIG, PIN_ECHO);
// переменная для хранения измеренного расстояния
float dist_cm=0;

void setup()  {
  // запуск последовательного порта
```

```
  Serial.begin(9600);
}

void loop() {
    // получить данные с дальномера
    dist_cm = ultrasonic.Ranging(CM);
    // вывести в последовательный порт
    Serial.println(dist_cm);
    // пауза перед следующим измерением
    delay(200);
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_56_01.zip.**

Загружаем скетч на плату Arduino, открываем монитор последовательного порта и смотрим значение рассояния (рис. 56.2).

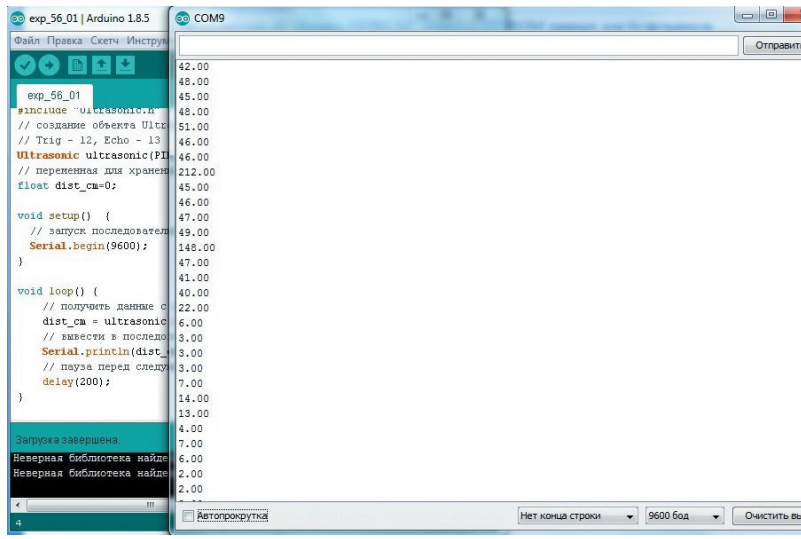


Рис. 56.2. Вывод данных датчика HC-SR04 в последовательный порт

Эксперимент 57.
Радар на шаговом двигателе и датчике HC-SR04

В этом эксперименте мы создадим радар на датчике расстояния HC-SR04 и шаговом двигателе

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Шаговый двигатель 28BYJ-48 – 1;
- ❑ Драйвер двигателя на микросхеме ULN2003 – 1;
- ❑ LCD Keypad shield – 1;
- ❑ Двухкоординатный джойстик – 1;
- ❑ Ультразвуковой датчик HC-SR04;
- ❑ Провода MM – 13;
- ❑ Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

В этом эксперименте создадим радар на ультразвуковом датчике расстояния HC-SR04. Датчик будет крепиться к валу шагового двигателя. Для управления движением шагового двигателя будем использовать двухкоординатный джойстик. Данные о расстоянии будем выводить на экран LCD Keypad shield.

Схема соединений показана на рисунке 57.1.

С помощью двухкоординатного джойстика устанавливаем направление вращения шагового двигателя (используем одну из осей – ось X):

```
analogRead(A0) < 300                // влево
analogRead(A0 > 800                 // вправо
```

Периодически получаем данные с ультразвукового датчика HC-SR04 и выводим на экран LCD Keypad shield.

Содержимое скетча показано в листинге 57.1.

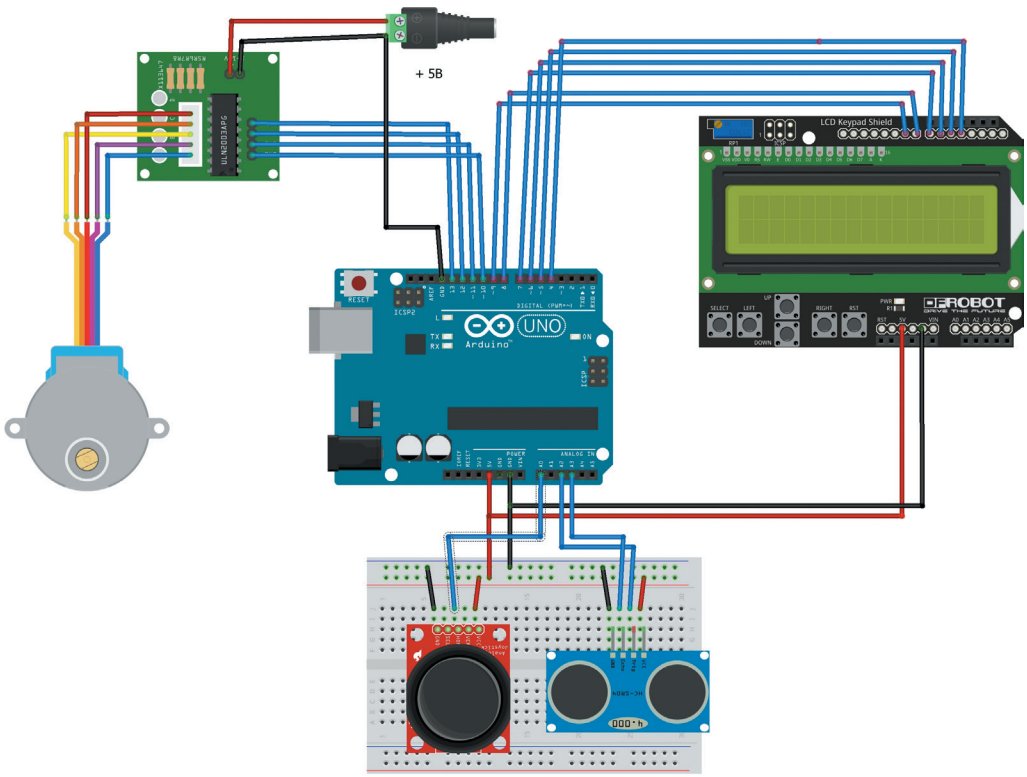


Рис. 57.1. Схема соединений для радара

Листинг 57.1.

```
// подключение библиотек
#include <LiquidCrystal.h>
#include<AccelStepper.h>
#include "Ultrasonic.h"
// контакты подключения
#define IN1 10
#define IN2 11
#define IN3 12
#define IN4 13
#define PIN_TRIG A2
#define PIN_ECHO A3
// создаем экземпляров объектов
AccelStepper motor(8, IN1, IN3, IN2, IN4);
LiquidCrystal lcd(8,9,4,5,6,7);
Ultrasonic ultrasonic(PIN_TRIG, PIN_ECHO);
//
int minSpeed=50;
int maxSpeed=900;
int speed=400;
int stepspeed=50;
int dir=0;
//
```

```
unsigned long millisHC04=0;
// переменная для хранения значений джойстика
int valX = 0;
// переменная для хранения измеренного расстояния
float dist_cm=0;

void setup(){
  lcd.begin(16,2);
  // очистить
  lcd.clear();
  lcd.setCursor(4,0);
  lcd.print("distance");

  // запуск датчика DHT
  lcd.setCursor(0,0);
  // начальные установки
  motor.setMaxSpeed(900.0);
  motor.setAcceleration(100.0);
  motor.setSpeed(speed);
}

void loop(){
   // чтение данных джойстика
   valX = analogRead(A0);
   // определение нажатия кнопок
   if(valX<300) { // вправо
      dir=-1;
      motor.setSpeed(speed*dir);
   }
   else if(valX>800) { // влево
      dir=1;
      motor.setSpeed(speed*dir);
   }
   else  { // останов
      dir=0;

   }
  // движение мотора
  if(dir!=0)
     motor.runSpeed();
  if(millis()- millisHC04>500) {
     // получить данные с дальномера
     dist_cm = ultrasonic.Ranging(CM);
     // вывести на дисплей
     lcd.setCursor(6,1);
     lcd.print("        ");
     lcd.setCursor(6,1);
     lcd.print(dist_cm);
     millisHC04-millis();
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_57_01.zip.

Загружаем скетч на плату Arduino и проверяем работу радара.

Эксперимент 58.
Компас на шаговом двигателе и модуле GY273 HMC5883

В этом эксперименте мы создадим компас на шаговом двигателе с использованием магнитного датчика

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Шаговый двигатель 28BYJ-48 – 1;
- ❑ Драйвер двигателя на микросхеме ULN2003 – 1;
- ❑ Модуль GY273 HMC5883 – 1;
- ❑ Стрелка – 1;
- ❑ Провода MM – 13;
- ❑ Провода MF – 4.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Микросхема HMC5883L представляет собой 3-х осевой цифровой компас. В качестве сенсоров используется три магниторезистивных датчика.

Датчик может использоваться в мобильных телефонах, планшетах, навигационном оборудовании и прочей бытовой электронике, но для радиолюбителей он может быть интересен тем, что цифровой компас может очень пригодится при конструировании роботов и радиоуправляемых моделей. Точность определения направления 1° – 2°, может применяться в сильных магнитных полях (до ±8 Гаусс), работает по протоколу I2C.

В наборе присутствует модуль GY273 на микросхеме HMC5883L. На плате удобные выводы для подключения и они распаяны подтягивающими резисторами для контактов SDA и SCL.

Схема соединений для подключения модуля GY273 к плате Arduino показана на рисунке 58.1.

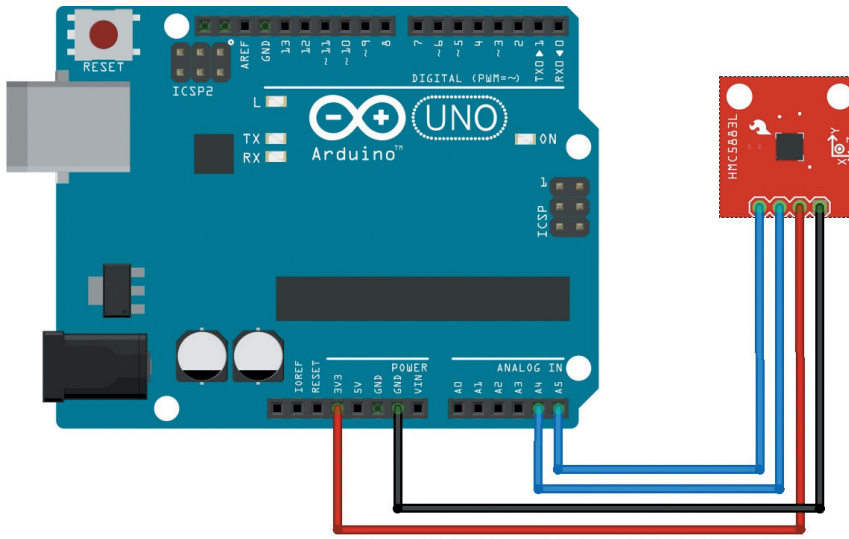


Рис. 58.1. Схема соединений для подключения модуля GY273 к плате Arduino

При программировании будем использовать Arduino-библиотеку HMC5883, которую можно скачать с сайта Arduino-kit по ссылке **https://arduino-kit.ru/scetches/HMC5883.zip.** Загрузим на плату Arduino скетч для получения положения датчика относительно магнитного полюса Земли в плоскости XY.

Содержимое скетча показано в листинге 58.1.

Листинг 58.1.

```
// Подключение библиотек
#include "Wire.h"
#include "HMC5883L.h"
// Создание объекта
HMC5883L compass;

void setup(){
  Serial.begin(9600);
  Wire.begin();
  // создаем экземпляр HMC5883L библиотеки
  compass = HMC5883L();
  // инициализация HMC5883L
setupHMC5883L();
}

void loop(){
  int heading = getHeading();
  Serial.println(heading);
  delay(250);
}

void setupHMC5883L(){
  // инициализация HMC5883L и проверка наличия ошибок
```

```
  int error;
  // чувствительность датчика
  error = compass.SetScale(0.88);
  if(error != 0)
    Serial.println(compass.GetErrorText(error));
  // установка режима измерений как Continuous (продолжительный)
  error = compass.SetMeasurementMode(Measurement_Continuous);
  if(error != 0)
     Serial.println(compass.GetErrorText(error));
}

float getHeading(){
  // считываем данные и рассчитываем  направление
  MagnetometerScaled scaled = compass.ReadScaledAxis();
  // высчитываем направление
  float heading = atan2(scaled.YAxis, scaled.XAxis);

  // корректируем значения с учетом знаков
  if(heading < 0) heading += 2*PI;
  if(heading > 2*PI) heading -= 2*PI;
  // переводим радианы в градусы
  return heading * RAD_TO_DEG;
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_58_01.zip.**

Открываем монитор последовательного порта и смотрим показания датчика при вращении (рис. 58.2). Вращая датчик вокруг своей оси, будет изменяться градус поворота. 0° – это будет север, а 180° – юг. Датчик необходимо располагать строго в горизонтальной плоскости. Стоит его наклонить и тогда данные будут неверные.

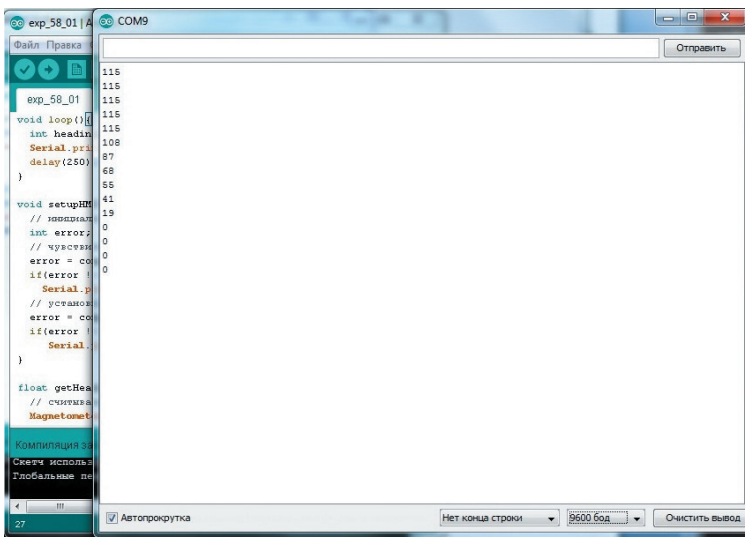


Рис. 58.2. Вывод показаний с датчика HMC5883L в последовательный порт

В этом эксперименте мы создадим компас на шаговом двигателе, который будет стрелкой указывать на север. Для этого необходимо установить на макетной плате модуль GY273, а стрелку на валу шагового двигателя в горизонтальной плоскости. Шаговый двигатель и модуль GY273 подключить к плате Arduino.

Схема соединений показана на рисунке 58.3.

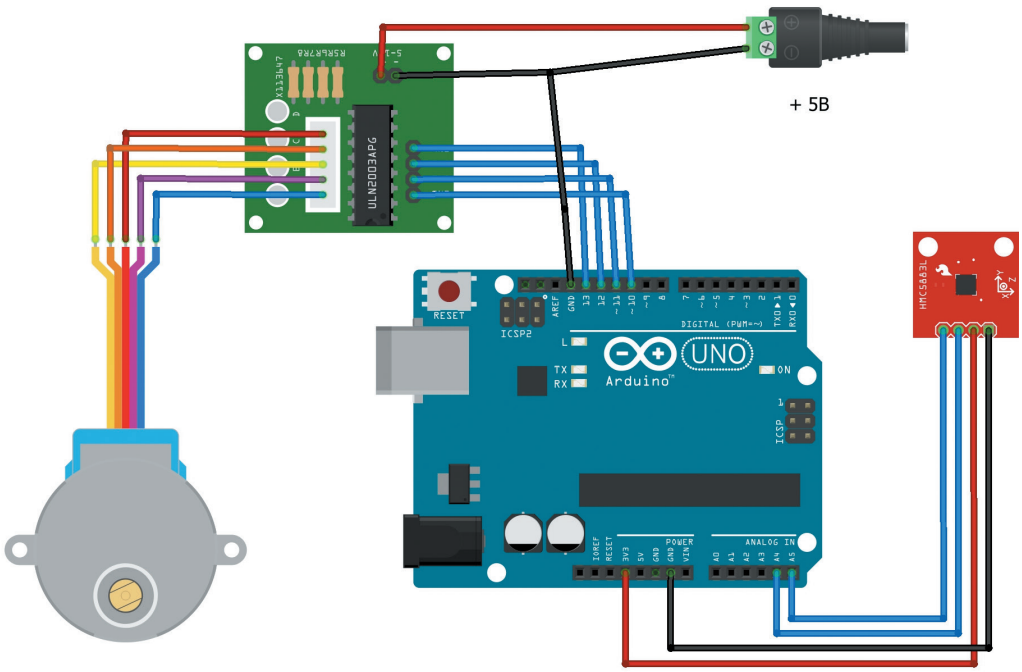


Рис. 58.3. Схема соединений для компаса на шаговом двигателе.

При отлонении показаний от 0 будем запускать шаговый двигатель. Содержимое скетча показано в листинге 58.2.

Листинг 58.1.

```
// подключение библиотек
#include<AccelStepper.h>
#include "Wire.h"
#include "HMC5883L.h"
// контакты подключения
#define IN1 10
#define IN2 11
#define IN3 12
#define IN4 13
// создаем экземпляры объектов
AccelStepper motor(8, IN1, IN3, IN2, IN4);
```

```
HMC5883L compass;
// переменные
int minSpeed=50;
int maxSpeed=900;
int speed=300;
int stepspeed=50;
int dir=0;
//


void setup(){
  Serial.begin(9600);
  Wire.begin();
  // создаем экземпляр HMC5883L библиотеки
  compass = HMC5883L();
  // инициализация HMC5883L
  setupHMC5883L();
  // начальные установки двигателя
  motor.setMaxSpeed(900.0);
  motor.setAcceleration(100.0);
  motor.setSpeed(200);
}

void loop(){
  int heading = getHeading();
  //Serial.println(heading);
   if(heading>3 && heading<=180) { // вправо
      dir=-1;
      motor.setSpeed(speed*dir);
   }
   else if(heading>180 && heading<=356) { // влево
      dir=1;
      motor.setSpeed(speed*dir);
   }
   else  { // останов
      dir=0;
   }
  // движение мотора
  if(dir!=0)
     motor.runSpeed();

}

void setupHMC5883L(){
  // инициализация HMC5883L, и проверка наличия ошибок
  int error;
  // чувствительность датчика
  error = compass.SetScale(0.88);
```

```
  if(error != 0)
    Serial.println(compass.GetErrorText(error));
  // установка режима измерений как Continuous (продолжительный)
  error = compass.SetMeasurementMode(Measurement_Continuous);
  if(error != 0)
     Serial.println(compass.GetErrorText(error));
}

float getHeading(){
  // считываем данные и рассчитываем  направление
  MagnetometerScaled scaled = compass.ReadScaledAxis();
  // высчитываем направление
  float heading = atan2(scaled.YAxis, scaled.XAxis);

  // корректируем значения с учетом знаков
  if(heading < 0) heading += 2*PI;
  if(heading > 2*PI) heading -= 2*PI;
  // переводим радианы в градусы
  return heading * RAD_TO_DEG;
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_58_02.zip.**
Загружаем скетч на плату Arduino и проверяем работу компаса.

Эксперимент 59.
RFID-идентификация. Считыватель RFID RC522

В этом эксперименте познакомимся с RFID – технологией автоматической бесконтактной идентификации объектов при помощи радиочастотного канала связи, и рассмотрим работу Arduino с RFID-модулем RC522, работающем с метками 13.56 МГц

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Модуль RC522 – 1;
- ❑ RFID-метки 13.56 МГц – 2;
- ❑ Провода MM – 7.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Радиочастотная RFID — это технология автоматической бесконтактной идентификации объектов при помощи радиочастотного канала связи. Любая RFID-система состоит из:

- ❑ RFID-метки;
- ❑ считывателя информации (RFID-ридера);
- ❑ микроконтроллера или компьютера для дальнейшей обработки информации.

Идентификация объектов производится по уникальному цифровому коду, который считывается из памяти электронной метки, прикрепляемой к объекту идентификации.

Считыватель содержит в своем составе передатчик и антенну, и посылает в эфир электромагнитные сигналы определенной частоты. RFID-метки "отвечают" собственным сигналом, который содержит информацию об идентификационном

номере данной метки и данных об объекте, оснащенном данной меткой. Этот сигнал улавливается антенной считывателя, информация расшифровывается и передается для дальнейшей обработки.

В состав набора входит модуль RC522 – RFID-модуль 13.56 МГц с SPI-интерфейсом. В комплекте к модулю идет две RFID-метки — в виде карты и брелока. В RFID-метках, работающих на данной частоте, реализована криптографическая защита, что обеспечивает защиту от копирования и подделки.

Схема соединений для подключения RFID-модуля RC522 к плате Arduino. показана на рисунке 59.1. Обратите внимание, что питание модуля 3.3В!

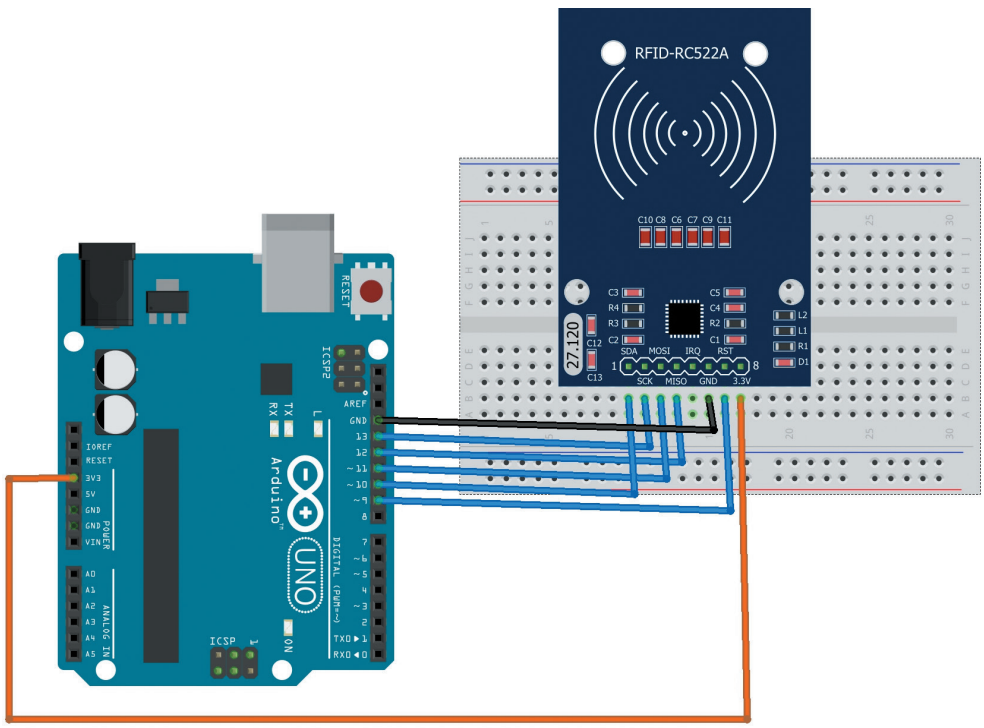


Рис. 59.1. Схема соединений для подключения RFID-модуля RC522 к плате Arduino

Для взаимодействия Arduino с модулем RC522 будем использовать Arduino-библиотеку MFRC522, которую можно скачать с сайта Arduino-kit по ссылке https://arduino-kit.ru/scetches/MFRC522.zip. Установим библиотеку и напишем скетч вывода UID-метки и ее типа в последовательный порт. Скетч представлен в листинге 59.1.

Содержимое скетча показано в листинге 59.1.

Листинг 59.1.

```
// Подключение библиотек
#include <SPI.h>
#include <MFRC522.h>
// константы подключения контактов SS и RST
#define RST_PIN 9
#define SS_PIN 10
// Инициализация MFRC522
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.

void setup() {
   Serial.begin(9600); // инициализация последовательного порта
   SPI.begin(); // инициализация SPI
   mfrc522.PCD_Init(); // инициализация MFRC522
}

void loop() {
   if ( ! mfrc522.PICC_IsNewCardPresent())
      return;
   // чтение карты
   if ( ! mfrc522.PICC_ReadCardSerial())
      return;
   // показать результат чтения UID и тип метки
   Serial.print(F("Card UID:"));
   dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
   Serial.println();
   Serial.print(F("PICC type: "));
   byte piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
   Serial.println(mfrc522.PICC_GetTypeName(piccType));
   delay(2000);
}
// Вывод результата чтения данных в HEX-виде
void dump_byte_array(byte *buffer, byte bufferSize) {
   for (byte i = 0; i < bufferSize; i++) {
      Serial.print(buffer[i] < 0x10 ? " 0" : " ");
      Serial.print(buffer[i], HEX);
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_59_01.zip.**

Загружаем скетч на плату Arduino. При поднесении меток (брелоков и карт) в монитор последовательного порта выводится UID (уникальный идентификатор) и тип метки (рис. 59.2).

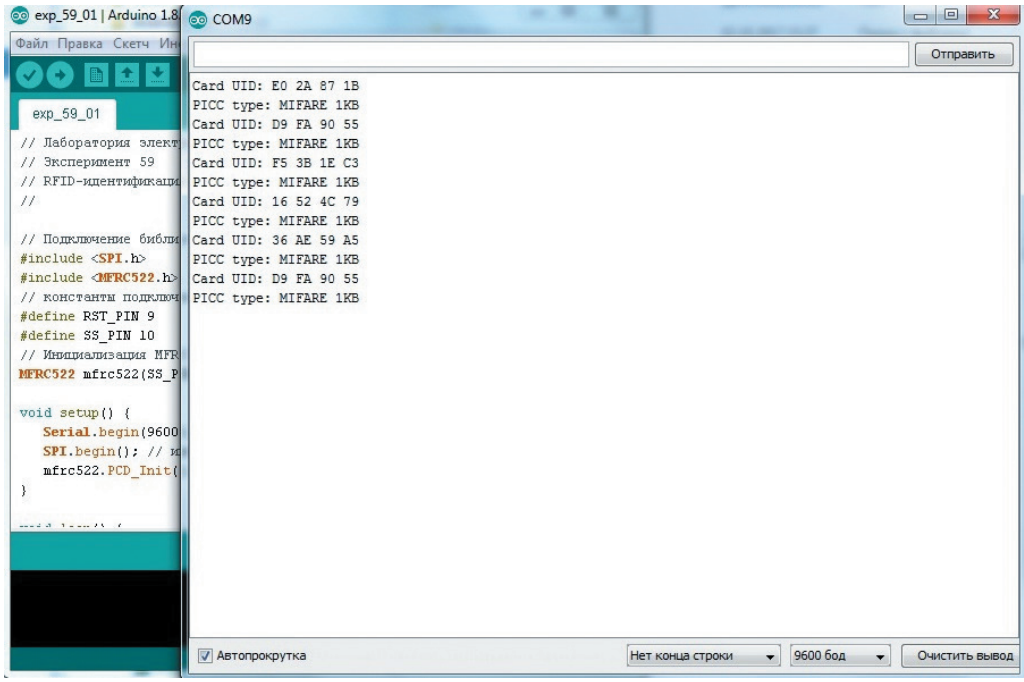


Рис. 59.2. Вывод в монитор последовательного порта выводится UID (уникального идентификатора) и тип метки

Эксперимент 60.
Организация контроля доступа по RFID-меткам

В этом эксперименте создадим проект организации доступа по RFID-меткам.

В эксперименте мы будем использовать следующие компоненты:

- Плата Arduino UNO – 1;
- Кабель USB – 1;
- Плата прототипирования – 1;
- Модуль RC522 – 1;
- RFID-метки 13.56 МГц – 2;
- LCD Keypad shield – 1;
- Динамик – 1;
- Релейный модуль – 1;
- Провода MM – 7.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

В этом эксперименте создаем проект организации доступа по RFID-меткам. Arduino будет отправлять сигнал «открытие» на реле (например, открытие электромеханического замка). Доступ осуществляется только для меток из списка. Схема соединений для подключения RFID-модуля RC522 к плате Arduino.показана на рисунке 60.1.

Приступим к написанию скетча.

Уникальные идентификаторы карт или брелоков, по которым доступен вход запишем в массиве uid_ok:

```
byte uidok[][4]={
                  {0x11,0x22,0x33,0x44},
        {0x11,0x22,0x33,0x45}
};
```

Пин для подключения реле:

```
#define RELAY_RC522_PIN 4
```

В цикле будем считывать данные с поднесенной карты или брелока. Затем сверим

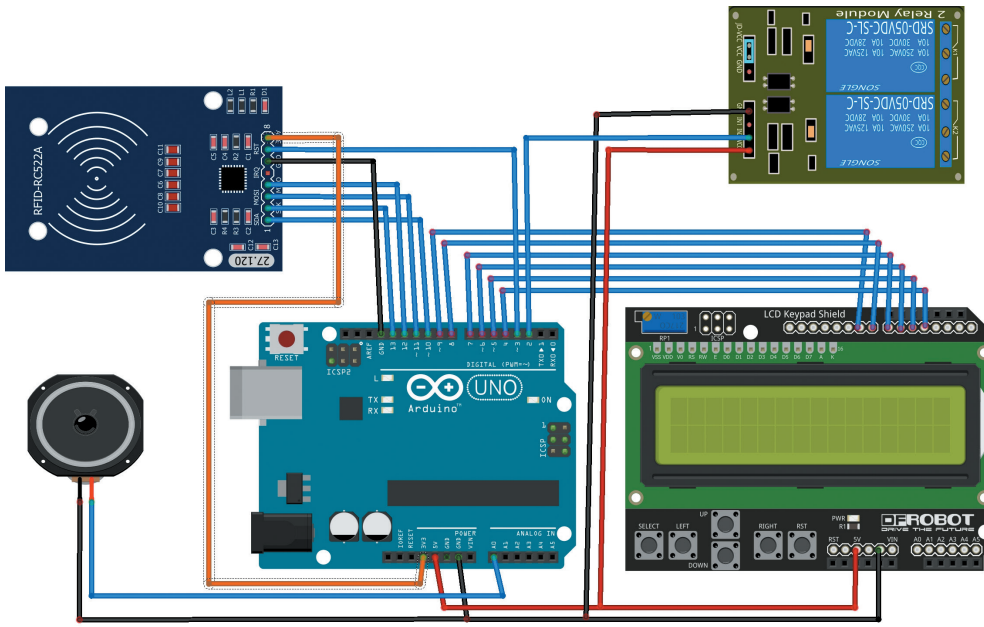


Рис. 60.1. Схема соединений для организации доступа с помощью RFID-меток

uid карты со списком разрешенных uid, и в случае присутствия карты в списке будем отправлять сигнал срабатывания на реле (уровень LOW).

Содержимое скетча показано в листинге 60.1.

Листинг 60.1

```
# // подключение библиотек
#include <LiquidCrystal.h>
#include <SPI.h>
#include <MFRC522.h>
// пин для подключения реле
#define RELAY_RC522_PIN 2
// пин для подключения динамика
#define SPEAKER_PIN A0

// уровни для включения/выключения реле
#define RELAY_ON 0
#define RELAY_OFF 1
// Инициализация MFRC522
// константы подключения контактов SS и RST
#define RST_PIN 3
#define SS_PIN 10
// Создание экземпляра MFRC522
MFRC522 mfrc522(SS_PIN, RST_PIN);
// создаем экземпляр объекта дисплея
LiquidCrystal lcd(8,9,4,5,6,7);
```

```
// массив разрешенных uid
byte uidok[][4]={
                 {0xE0, 0x2A, 0x87, 0x1B},
                 {0xD9, 0xFA, 0x90, 0x55},
                 {0x36, 0xAE, 0x59, 0xA5 }
};

void setup() {
   // инициализация дисплея
   lcd.begin(16,2);
   // очистить
   lcd.clear();
   // инициализация SPI
   SPI.begin();
   // инициализация MFRC522
   mfrc522.PCD_Init();
   // сконфигурировать вывод реле как OUTPUT
   pinMode(RELAY_RC522_PIN,OUTPUT);
   // и выключить
   digitalWrite(RELAY_RC522_PIN, RELAY_OFF);
   // сконфигурировать вывод динамика
   pinMode(SPEAKER_PIN,OUTPUT);

}

void loop() {
  if (mfrc522.PICC_IsNewCardPresent()) {
   // чтение карты
   if ( mfrc522.PICC_ReadCardSerial()) {
      // показать результат чтения UID
      if(compare_uid(mfrc522.uid.uidByte, mfrc522.uid.size)) {
          lcd.setCursor(4,1);
          lcd.print("OK !!!");
          tone(SPEAKER_PIN,494,300);
          delay(500);
          tone(SPEAKER_PIN,440,200);
          // включить реле
          digitalWrite(RELAY_RC522_PIN, RELAY_ON);
          delay(4000);
          digitalWrite(RELAY_RC522_PIN, RELAY_OFF);
      } else {
          lcd.setCursor(4,1);
          lcd.print("NO !!!");
          tone(SPEAKER_PIN,293,300);
          delay(500);
          tone(SPEAKER_PIN,329,500);
          delay(1000);
      }
   }
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_60_01.zip.**

Загрузим данный скетч на плату Arduino и проверим работу ограничения доступа с помощью RFID.

Эксперимент 61.
Запись информации на RFID-метку

В этом эксперименте рассмотрим запись данных на RFID-метку

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Модуль RC522 – 1;
- ❑ RFID-метки 13.56 МГц – 2;
- ❑ Провода MM – 7.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

В экспериментах 59, 60 мы получали UID (уникальный идентификатор) RFID-меток, работающих на частоте 13.56 МГц. Но каждая метка, кроме UID имеет память, из которой можно считывать данные и в которую можно записывать данные. Подключим RFID-считыватель согласно рис. 61.1 и загрузим на плату Arduino пример DumpInfo из библиотеки MFRC522 и поднесем к считывателю метку (рис. 61.2). Данная карта имеет 1Кб EEPROM.

Рассмотрим запись персональных данных на метку. Данные на карту записываются поблочно (по 16 байт). Отправлять данные будем по последовательному порту. Формат передачи данных:

```
*block;data$
```

block	data	Назначение	Block на RFID-метке
1	ssssssssssssssss (max 16 байт)	фамилия	4
2	ssssssssssssssss (max 16 байт)	имя	5
3	dd/mm/YYYY	дата рождения	6

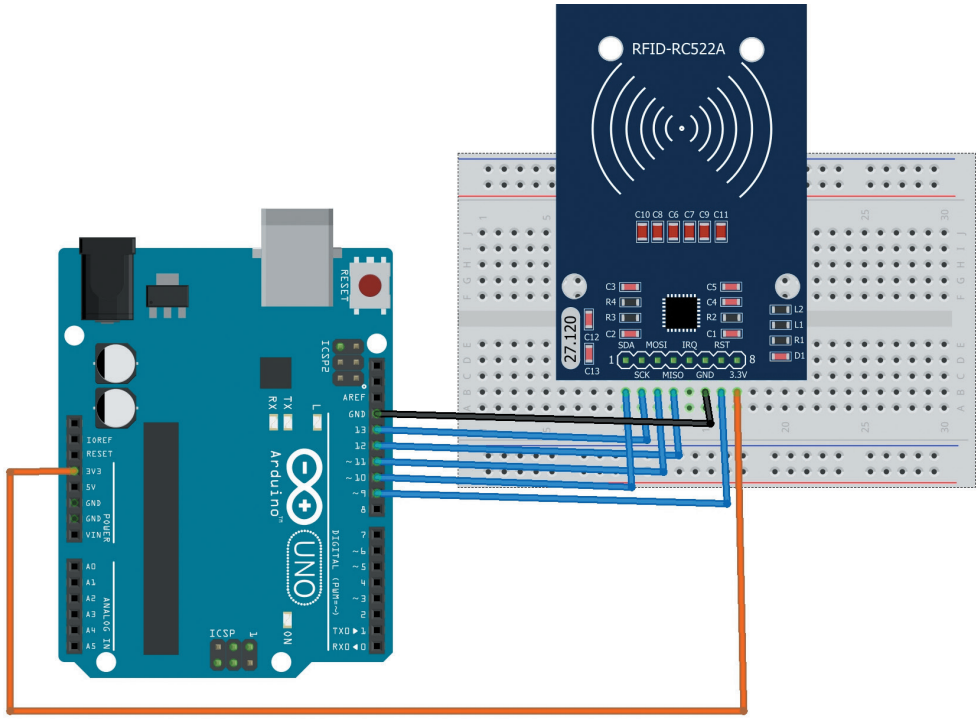


Рис. 61.1. Схема соединений для подключения RFID-модуля RC522 к плате Arduino

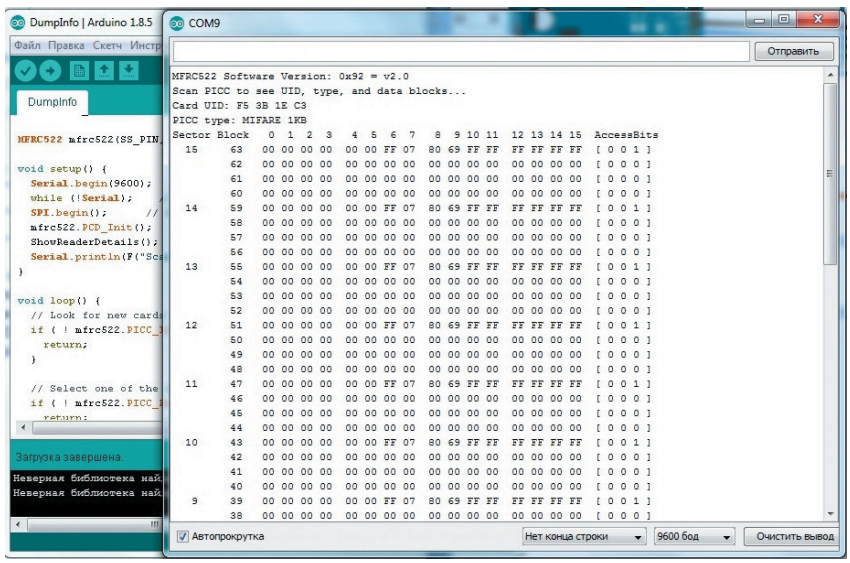


Рис. 61.2. Вывод поблочно данных памяти RFID-метки

Содержимое скетча представлено в листинге 61.1.

Листинг 61.1

```
// Подключение библиотек
#include <SPI.h>
#include <MFRC522.h>
// константы подключения контактов SS и RST
#define RST_PIN 9
#define SS_PIN 10
// Инициализация MFRC522
MFRC522 mfrc522(SS_PIN, RST_PIN);
MFRC522::StatusCode status;
MFRC522::MIFARE_Key key;
// строка для получения данных
String inputString = "";
// признак конца строки
boolean stringComplete = false;
// полученная команда
int block=0;
// полученные данные
byte data[16];

void setup() {
   Serial.begin(9600); // инициализация последовательного порта
   SPI.begin(); // инициализация SPI
   mfrc522.PCD_Init(); // инициализация MFRC522
}

void loop() {
  // ждем признака конца строки
  if (stringComplete) {
    Serial.println(inputString);
    if(parse_data()) {
       Serial.println("get data - OK!");
       Serial.println("Bring the card to the reader");
       Serial.print("Wait 10 sec ");
       if(write_data())
          Serial.println("Write OK");
       else
          Serial.println("Write error");
    }
    else {
       Serial.println("Wrong data");
    }
    // очистить строку
    inputString = "";
    stringComplete = false;
```

```
    }

}
// Вывод результата чтения данных в HEX-виде
void dump_byte_array(byte *buffer, byte bufferSize) {
   for (byte i = 0; i < bufferSize; i++) {
      Serial.print(buffer[i] < 0x10 ? " 0" : " ");
      Serial.print(buffer[i], HEX);
   }
}
// запись данных на карту
boolean write_data() {
   unsigned long millis1=millis();
   int sec=10;

   while(sec>0) {
      if(millis()-millis1>=1000) {
         Serial.print(".");
         sec--;
         millis1=millis();
      }
      // Проверка поднесена ли карта
      if ( mfrc522.PICC_IsNewCardPresent()) {
         // Считываем инфо о карте.
         if ( mfrc522.PICC_ReadCardSerial()) {
            Serial.print(F("Card UID:"));
            for (byte i = 0; i < 4; i++) {
               mfrc522.uid.uidByte[i];
               Serial.print(mfrc522.uid.uidByte[i],HEX);
            }
            Serial.println();
            //
            Serial.println(F("Authenticating using key A..."));
            for (byte i = 0; i < 6; i++)
               key.keyByte[i] = 0xFF;
            status = mfrc522.PCD_Authenticate(
                  MFRC522::PICC_CMD_MF_AUTH_KEY_A, block+4, &key,
                  &(mfrc522.uid));
            if (status != MFRC522::STATUS_OK) {
               Serial.print(F("PCD_Authenticate() failed: "));
               Serial.println(mfrc522.GetStatusCodeName(status));
               return false;
            }
            Serial.println(F("PCD_Authenticate() success: "));
            Serial.println("write data to card");
            byte trailerBlock   = 7;
            // Authenticate using key B
           Serial.println(F("Authenticating again using key B..."));
```

```
            status = (MFRC522::StatusCode) mfrc522.PCD_Authenticate(
                     MFRC522::PICC_CMD_MF_AUTH_KEY_B, trailerBlock,
                     &key, &(mfrc522.uid));
            if (status != MFRC522::STATUS_OK) {
               Serial.print(F("PCD_Authenticate() false: "));
               return false;
            }
            Serial.println(mfrc522.GetStatusCodeName(status));
            // Write data to the block
            Serial.print(F("Writing data into block "));
            Serial.print(block+4);
            Serial.println(F(" ..."));
            status = (MFRC522::StatusCode) mfrc522.MIFARE_Write
                     (block+4, data, 16);
            if (status != MFRC522::STATUS_OK) {
               Serial.print(F("MIFARE_Write() false: "));
               return false;
            }
            Serial.println(mfrc522.GetStatusCodeName(status));
         }
      mfrc522.PICC_HaltA();
      mfrc522.PCD_StopCrypto1();
      return true;
      }
   }
   return false;
}

// парсинг получаемых данных
boolean parse_data() {
   // первый символ
   if(inputString[0]!='*')
       return false;
   if(inputString[2]!=';')
       return false;
   if(inputString[1]<0x31 || inputString[1]>0x33)
       return false;
   block=int(inputString[1]-0x31);
   Serial.print("block=");Serial.println(block);
   for(int i=3;i<inputString.length()-1;i++)
      data[i-3]=inputString[i];
   // остальное пробелы
   for(int i=inputString.length()-4;i<16;i++)
      data[i]=0x20;
   //
   for(int i=0;i<16;i++) {
      Serial.print(data[i],HEX);Serial.print(" ");
   }
```

```
    Serial.println();
    return true;

}
// получение данных из последовательного порта
void serialEvent() {
  while (Serial.available()) {
    // получить байт
    char inChar = (char)Serial.read();
    // добавить в строку
    inputString += inChar;
    // если символ конца строки - '$'
    if (inChar == '$') {
      stringComplete = true;
    }
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_61_01.zip.**

Загрузим данный скетч на плату Arduino, а также загрузим по последовательному порту данные на RFID-метку (рис. 61.3).

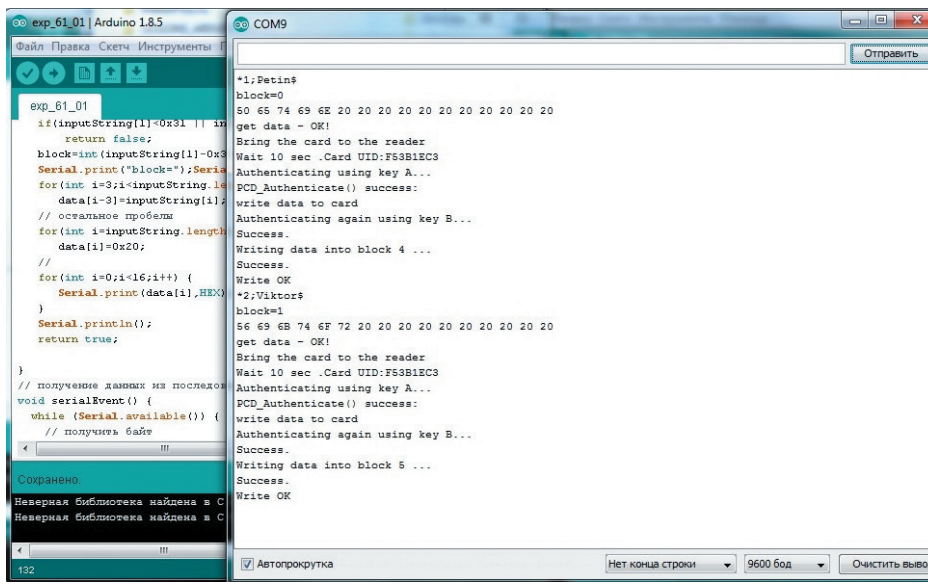


Рис. 61.3. Загрузка по последовательному порту данных на RFID-метку

Эксперимент 62.
Считывание данных с RFID-метки

В этом эксперименте рассмотрим считывание персональных данных с RFID-метки

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Модуль RC522 – 1;
- ❑ RFID-метки 13.56 МГц – 2;
- ❑ Провода MM – 7.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

В эксперименте 61 мы записали персональные данные на RFID-метку. Сейчас будем их считывать. На рис. 62.1. представлена схема соединений для данного эксперимента.

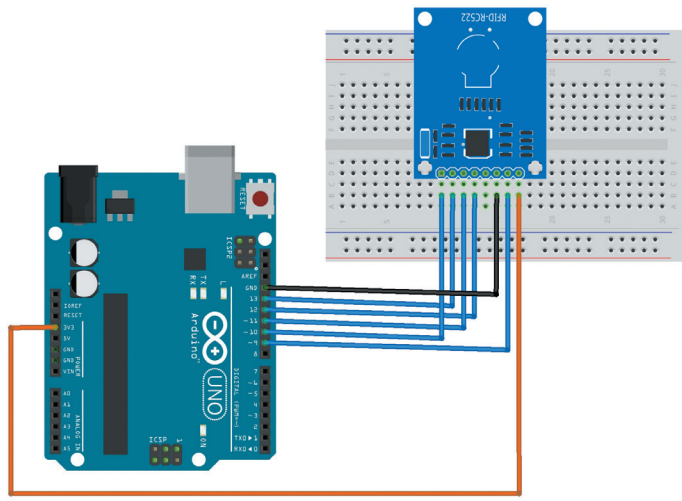


Рис. 62.1. Схема соединений для подключения RFID-модуля RC522 к плате Arduino

Приступим к написанию скетча. При обнаружении метки считываем данные из секторов 4, 5, 6 и выводим в последовательный порт.

Содержимое скетча показано в листинге 62.1.

Листинг 62.1

```
// подключение библиотек
#include <SPI.h>
#include <MFRC522.h>

// Инициализация MFRC522
// константы подключения контактов SS и RST
#define RST_PIN 9
#define SS_PIN 10
// Создание экземпляра MFRC522
MFRC522 mfrc522(SS_PIN, RST_PIN);
MFRC522::StatusCode status;
MFRC522::MIFARE_Key key;
// массив для заполнения данных
char persData[3][16];
// для смены изображения
unsigned long millis1=0;
int pos=0;

void setup() {
   // запуск последовательного порта
   Serial.begin(9600);
   // инициализация SPI
   SPI.begin();
   // инициализация MFRC522
   mfrc522.PCD_Init();

}

void loop() {
  if (mfrc522.PICC_IsNewCardPresent()) {
   // чтение карты
   if ( mfrc522.PICC_ReadCardSerial()) {
      // показать результат чтения UID и тип метки
      Serial.print(F("Card UID:"));
      Serial.print(F("Card UID:"));
      for (byte i = 0; i < 4; i++) {
         mfrc522.uid.uidByte[i];
         Serial.print(mfrc522.uid.uidByte[i],HEX);
      }
      Serial.println();
      Serial.println(mfrc522.GetStatusCodeName(status));
      // записать пробелы в persData
      for(int i=0;i<3;i++) {
         for(int j=0;j<16;j++)
            persData[i][j]=0x20;
      }
      // чтение данных
      for(int i=0;i<3;i++) {
         // аутентификация карты
         Serial.println(F("Authenticating using key A..."));
         for (byte i = 0; i < 6; i++)
```

```
            key.keyByte[i] = 0xFF;
         status = mfrc522.PCD_Authenticate(
                  MFRC522::PICC_CMD_MF_AUTH_KEY_A, 4, &key,
                  &(mfrc522.uid)); //line 834 of MFRC522.cpp file
         if (status == MFRC522::STATUS_OK) {
            Serial.print(F("Authentication ok: "));
         }
         // чтение данных с секторов 4,5,6
         byte len = 18;
         status = mfrc522.MIFARE_Read(i+4, persData[i], &len);
         if (status == MFRC522::STATUS_OK) {
            Serial.print("Reading ok: ");
            Serial.println(mfrc522.GetStatusCodeName(status));
            Serial.print(F("Reading OK! "));
            // распечатать данные сектора
            for (uint8_t j = 0; j < 16; j++) {
               Serial.print(persData[i][j],HEX);Serial.print(" ");
            }
            Serial.println(" ");
         }
         Serial.println(mfrc522.GetStatusCodeName(status));
         // распечатать данные
         Serial.println("Personal data:");
      }
      // вывод персональных данных
      Serial.println(persData[0]);
      mfrc522.PICC_HaltA();
      mfrc522.PCD_StopCrypto1();
   }
  }

}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_62_01.zip.**

Загрузим данный скетч на плату Arduino и проверим считывание персональных данных с RFID-метки (рис. 62.2).

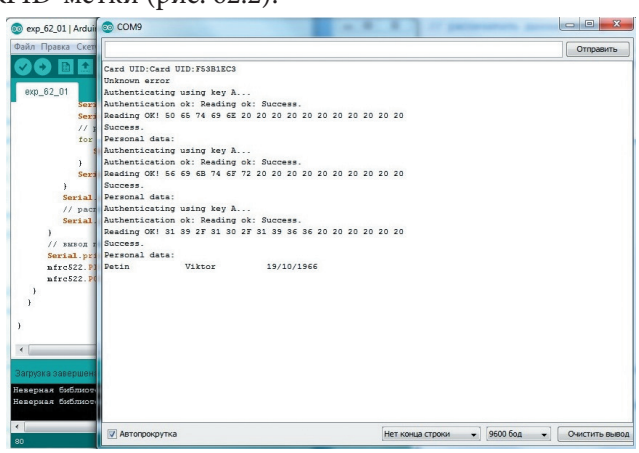


Рис. 62.2. Считывание персональных данных с RFID-метки

Эксперимент 63.
Подключение модуля TEA5767

В этом эксперименте рассмотрим подключение к плате Arduino модуля радио TEA5767

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Модуль FM-радио TEA5767 – 1;
- ❑ УНЧ стерео – 1;
- ❑ Провода MM – 17.

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Модуль TEA5767 позволяет собрать цифровой стерео радиоприемник УКВ-FM диапазона. Управление режимами работы выполняется микроконтроллером по шине I2C, соединяемой с контактами модуля. На выходе FM приемника установлена микросхема TDA1308 – звуковой усилитель для наушников. Модуль содержит гнезда для установки штекеров диаметром 3,5 мм. Подключаются антенна и наушники.

Модуль TEA5767 может работать в режиме поиска радиостанций. Поиск останавливается, когда обнаружена станция, имеющая уровень сигнала выше заданного порогового значения. В случае приема слабого сигнала происходит переключение из режима стерео в режим моно.

Чтобы получить на динамиках громкий звук необходимо использовать входящий в набор УНЧ.

Схема подключения показана на рис. 63.1.

Управление модулем осуществляется отправкой команд (5 байт) по протоколу I2C. Модуль так же позволяет читать из него информацию для реализации функции автопоиска и уровня сигнала. В листинге 63.1 показана настройка радио на определенную частоту.

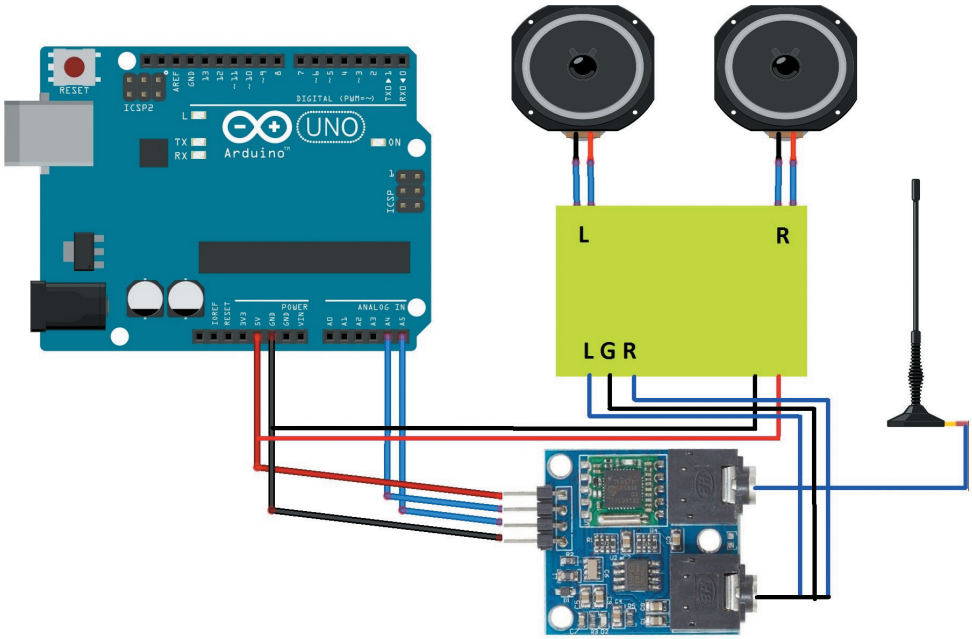


Рис. 63.1. Схема соединений для подключения модуля TEA5767 к плате Arduino.

Листинг 63.1

```
// Подключение библиотеки для I2C
#include <Wire.h>
// частота воспроизведения
float f = 103.2;

void setup() {
  Wire.begin();
  // запуск радио
  unsigned int freqB = 4 * (f * 1000000 + 225000) / 32768;
  byte freqH = freqB >> 8;
  byte freqL = freqB & 0XFF;
  Wire.beginTransmission(0x60);
  Wire.write(freqH);
  Wire.write(freqL);
  Wire.write(0xB0);
  Wire.write(0x10);
  Wire.write(0x00);
  Wire.endTransmission();
}

void loop() {;}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке
https://arduino-kit.ru/scetches/exp_63_01.zip.

Загрузим данный скетч на плату Arduino и проверим работу радио.

Для более удобной работы с модулем создана Arduino-библиотека TEA5767, которую можно скачать с сайта Arduino-kit по ссылке
https://arduino-kit.ru/scetches/TEA5767.zip.

В скетче (листинге 63.2) показана реализация работы радио на определенной частоте (выберите известную вам частоту).

Листинг 63.2

```
// Подключение библиотек
#include <TEA5767.h>
#include <Wire.h>

// Создание экземпляра объекта
TEA5767 Radio;

void setup() {
  Wire.begin();
  // запуск радио
  Radio.init();
  // установка частоты
  Radio.set_frequency(103.2);
}

void loop() {;}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке
https://arduino-kit.ru/scetches/exp_63_02.zip.

Загрузим данный скетч на плату Arduino и проверим работу радио.

Эксперимент 64.
Радиоприемник на модуле TEA5767

В этом эксперименте создадим радиоприемник на модуле TEA5767

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Модуль FM-радио TEA5767 – 1;
- ❑ УНЧ стерео – 1;
- ❑ LCD Keypad shield – 1;
- ❑ Провода MM – 27

Переключатели на плате Arduino+WiFi установите следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

В этом эксперименте создадим маленький радиоприемник с поиском доступных радиостанций. Данные о найденной радиостанции будем отображать на экране LCD Keypad shield. При нажатии на кнопки LEFT и RIGHT будем осуществлять поиск радиостанций. Модуль TEA5767 может работать в режиме поиска радиостанций. Поиск останавливается, когда обнаружена станция, имеющая уровень сигнала выше заданного порогового значения. В случае приема слабого сигнала происходит переключение из режима стерео в режим моно.

Схема подключения показана на рис. 64.1.

При написании скетча будем использовать библиотеку TEA5767.

Содержимое скетча показано в листинге 64.1.

Листинг 64.1

```
// Подключение библиотек
#include <TEA5767.h>
#include <Wire.h>
#include <LiquidCrystal.h>

// создание экземпляров
TEA5767 Radio;
```

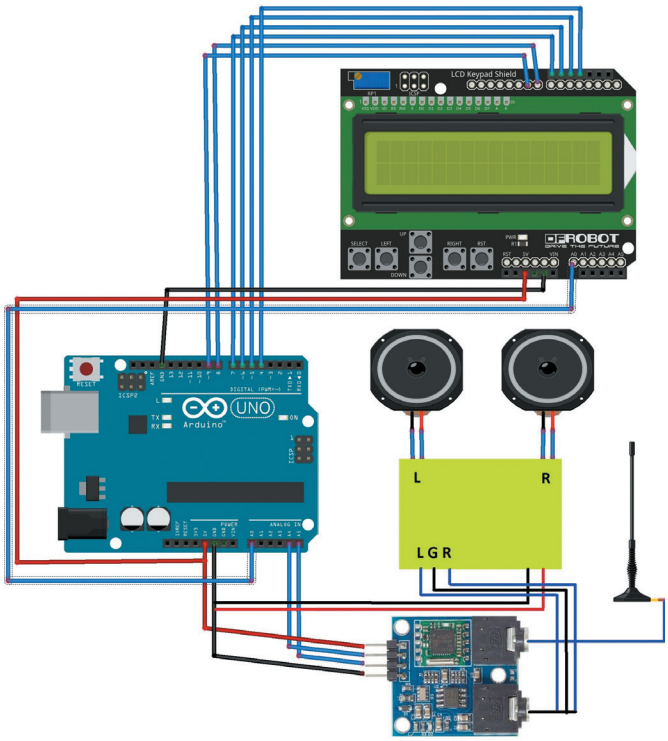


Рис. 64.1. Схема соединений радиоприемника на модуле TEA5767

```
LiquidCrystal lcd(8,9,4,5,6,7);

// служебные переменные
double old_frequency;
double frequency;
int search_mode = 0;
int search_direction;


void setup() {
  Wire.begin();
  // запуск радио
  Radio.init();
  // станция по умолчанию
  Radio.set_frequency(105.4);
  Serial.begin(9600);
  // запуск дисплея
  lcd.begin(16,2);
  lcd.clear();
}

void loop() {
```

```
  unsigned char buf[5];
  int stereo;
  int signal_level;
  double current_freq;
  unsigned long current_millis = millis();

  if (Radio.read_status(buf) == 1) {
    current_freq =  floor (Radio.frequency_available (buf)
                    / 100000 + .5) / 10;
    stereo = Radio.stereo(buf);
    signal_level = Radio.signal_level(buf);
    // вывод информации на дисплей
    lcd.setCursor(0,0);
    lcd.print("FM: "); lcd.print(current_freq);
    lcd.setCursor(0,1);
    if (stereo)
       {lcd.print("STEREO ");}
    else
       {lcd.print("MONO ");}

  }
  // поиск станции
  if (search_mode == 1) {
      if (Radio.process_search (buf, search_direction) == 1) {
          search_mode = 0;
          Serial.println("search ok");
      }
  }
  // поиск вверх по частоте
  if (analogRead(A0)<100) {
    Serial.println("RIGHT");
    search_mode = 1;
    search_direction = TEA5767_SEARCH_DIR_UP;
    Radio.search_up(buf);
    delay(300);
  }
  // поиск вниз по частоте
  if (analogRead(A0)>400 && analogRead(A0)<600) {
    Serial.println("LEFT");
    search_mode = 1;
    search_direction = TEA5767_SEARCH_DIR_DOWN;
    Radio.search_down(buf);
    delay(300);
  }
  delay(50);
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_64_01.zip.**

Загрузим данный скетч на плату Arduino и проверим работу радио.

Эксперимент 65.
Загрузка скетчей в модуль ESP8266 платы Arduino+WiFi

В этом эксперименте рассмотрим загрузку скетчей в WiFi-модуль ESP8266, который установлен на плате Arduino+WiFi

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;

На плате Arduino+WiFi интегрированы Arduino UNO R3 и WiFi-модуль ESP8266. Они могут работать вместе или каждый в отдельности. В этом эксперименте мы рассмотрим загрузку скетчей из Arduino IDE на модуль ESP8266.

Для загрузки скетчей переключатели на плате Arduino+WiFi необходимо установить следующим образом:

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

В Arduino IDE выбираем плату

Generic ESP8266 Module (Инструменты → Плата →) Generic ESP8266 Module. Выбираем любой пример, например Blink, подаем питание на плату Arduino+WiFi, выбираем порт подключения и загружаем скетч на плату (рис. 65.1).

Теперь загрузим на модуль ESP8266 скетч подключения модуля к доступной точке доступа. Перед загрузкой скетча нажимаем кнопку ESP reboot. Содержимое скетча показано в листинге 65.1. Вам необходимо изменить в нем значения переменных ssid и password на свои.

Листинг 65.1

```
// подключение библиотек
#include <ESP8266WiFi.h>
// данные SSID и пароль точки доступа
const char* ssid     = "Kassa1";
const char* password = "12345678";
```

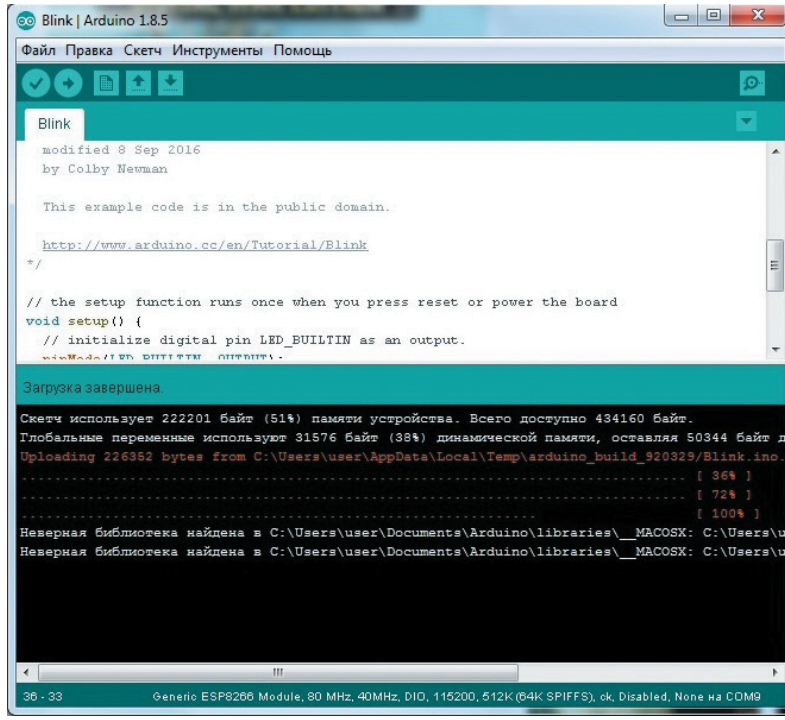


Рис. 65.1. Успешная загрузка скетча на модуль ESP8266 платы Arduino+WiFi

```
void setup() {
  Serial.begin(115200);
  delay(5000);
  //
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  // Подключение к точке доступа
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // успешно - вывести IP-адрес
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

```
void loop() {
  ;
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_65_01.zip.

Загрузим данный скетч в модуль ESP8266, откроем монитор последовательного порта и убедимся в подключении модуля к точке доступа WiFi (рис. 65.2).

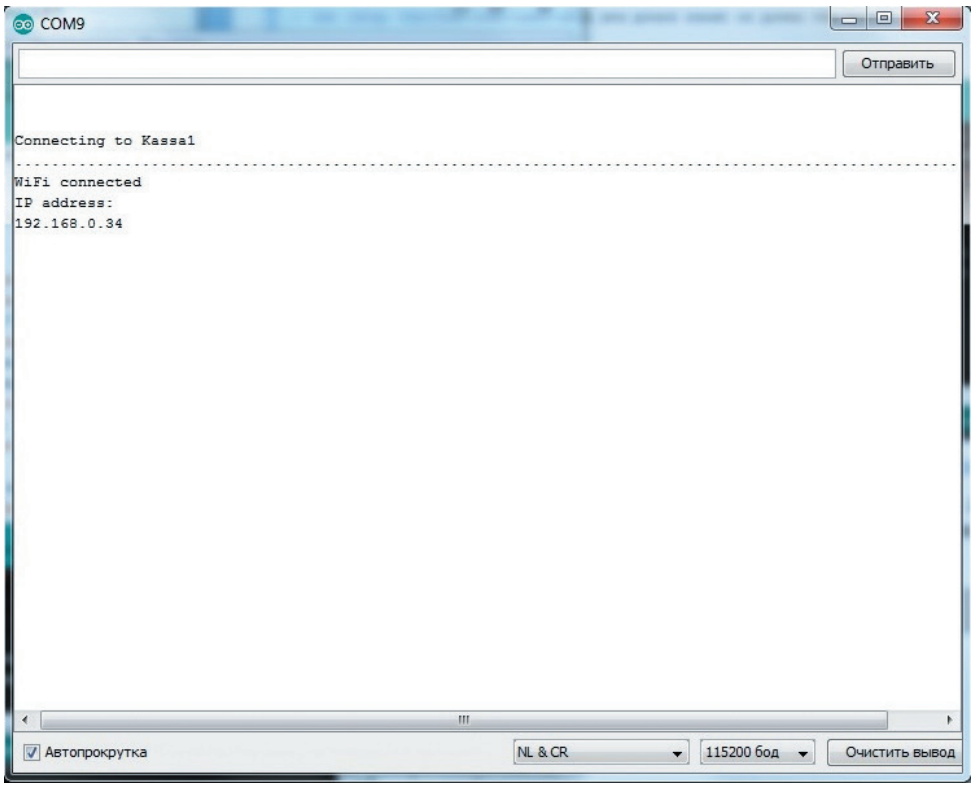


Рис. 65.2. Процесс подключения модуля ESP8266 к точке доступа WiFi

Теперь загрузим в модуль ESP8266 скетч создания сервера. Содержимое скетча показано в листинге 65.2. Вам необходимо изменить в нем значения переменных ssid и password на свои.

Листинг 65.2

```
// подключение библиотек
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
// данные SSID и пароль точки доступа
const char* ssid     = "Kassa1";
const char* password = "12345678";
// создание объекта сервера
ESP8266WebServer server(80);

// код вывода главной страницы
void handleRoot() {
  server.send(200, "text/html", "Arduino-KIT - <br>
                  Server Arduino+WiFi!");
}
// код вывода страницы about
void handleAbout() {
  server.send(200, "text/html", "<a href='https://arduino-kit.ru/'>
              Site Arduino-kit</a>");
}
// код вывода  страницы 404
void handleNotFound(){
  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET)?"GET":"POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i=0; i<server.args(); i++){
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);
}

void setup(void){

  Serial.begin(115200);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Подсоединение к точке доступа
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  // главная страница
  server.on("/", handleRoot);
  // страница /about
  server.on("/about", handleAbout);
  // страница 404 - не найдена
  server.onNotFound(handleNotFound);
  // запуск сервера
  server.begin();
  Serial.println("HTTP server started");
}
```

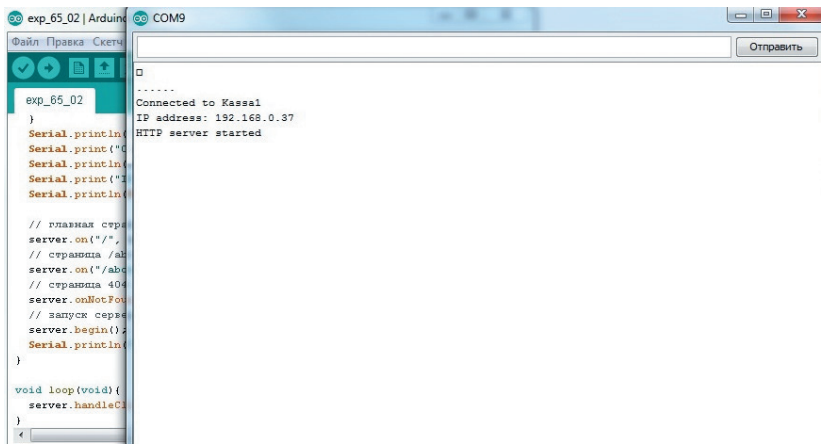


Рис. 65.3. Запуск WiFi сервера на модуле ESP8266

```
void loop(void){
  server.handleClient();
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_65_02.zip.**

Загрузим данный скетч в модуль ESP8266, откроем монитор последовательного порта и убедимся в подключении модуля к точке доступа WiFi и запуске сервера(рис. 65.3).

Заходим из браузера на страницу сервера (рис. 65.4, 65.5, 65.6).

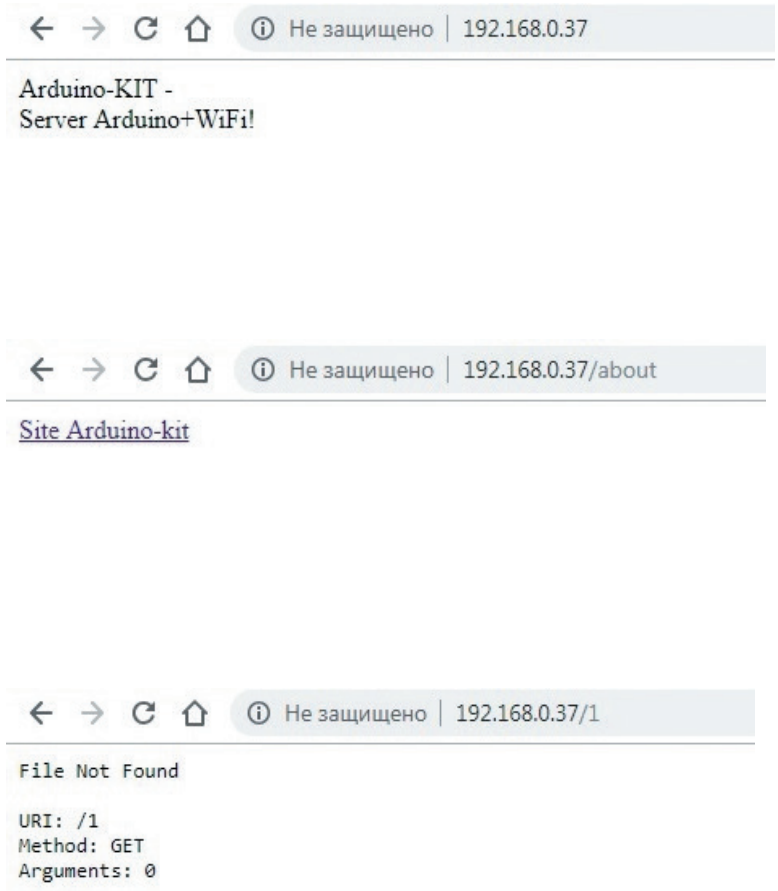


Рис. 65.4-6. Подключение к серверу

Эксперимент 66.
Обмен данными по последовательному порту между ESP8266 и Arduino UNO платы Arduino+WiFi

В этом эксперименте создадим связь двух модулей платы Arduino+WiFi (модуля ESP8266 и Arduino UNO) по последовательному порту

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования;
- ❑ USB-TTL адаптер – 1;
- ❑ Светодиод – 1;
- ❑ Резистор 220 Ом – 1;
- ❑ Провода MM – 5.

На плате Arduino+WiFi интегрированы Arduino UNO R3 и WiFi-модуль ESP8266. Рассмотрим режим соединения по последовательному порту. Чтобы иметь связь с компьютером для отладки подключения Arduino UNO и ESP8266 по последовательному порту, подключим к контактам 3 и 2 платы Arduino USB-TTL адаптер и создадим software-последовательный порт. Схема соединений показана на рис. 66.1.

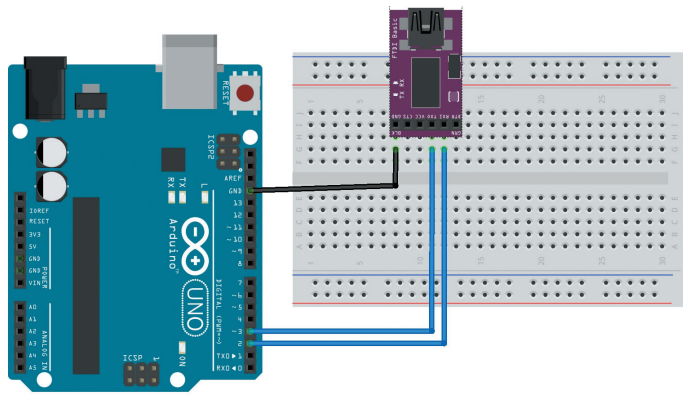


Рис. 66.1. Схема соединений для подключения USB-TTL-адаптера к плате Arduino

Загрузим на плату Arduino скетч из листинга 66.1.

Переключатели на плате Arduino+WiFi необходимо установить следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Листинг 66.1

```
// подключение библиотеки
#include <SoftwareSerial.h>
// создание экземпляра для USB-TTL
SoftwareSerial PCSerial(3, 2); // RX, TX

void setup()
{
  // запуск последовательного порта
  Serial.begin(115200);
  // запуск порта для USB-TTL
  PCSerial.begin(115200);
}

void loop()
{
  // получение USB-TTL -- Arduino
  if (PCSerial.available()) {
    Serial.write(PCSerial.read());
  }
  // получение ESP8266 -- Arduino
  if (Serial.available()) {
    PCSerial.write(Serial.read());
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_66_01.zip.**

Данный скетч необходим для обмена данными по последовательному порту между компьютером и Arduino UNO. Для проверки открываем терминальную программу, например Termite, которую можно скачать бесплатно из интернета. Отправляем данные из монитора последовательного порта Arduino IDE и получаем в программе Termite, и видим запрос-ответ (рис. 66.2).

Светодиод подключаем к контакту GPIO4 через резистор 220 Ом.

Загружаем в модуль ESP8266 скетч из листинга 66.2.

Переключатели на плате Arduino+WiFi необходимо установить следующим образом

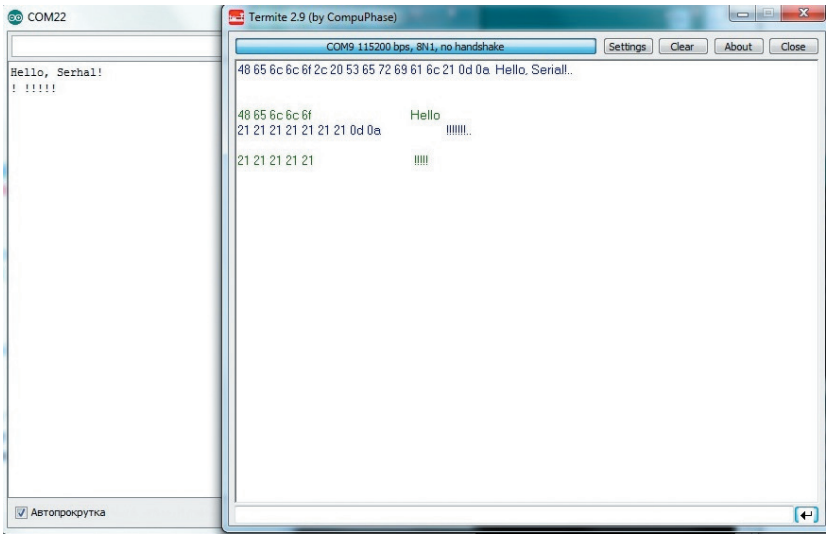


Рис. 66.2. Обмен данными между Serial (pin0 и pin1) и softwareSerial (pin2 и pin3)

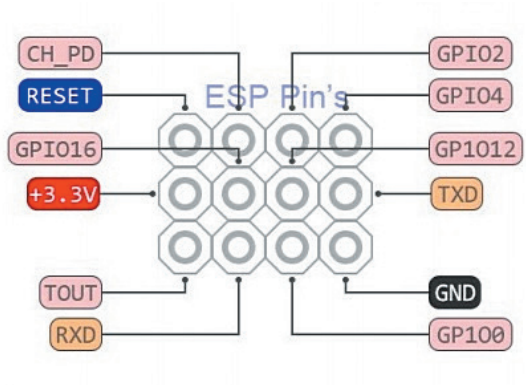


Рис. 66.3. Карта контактов ESP8266 платы Arduino+WiFi

Светодиод подключаем к контакту GPIO4 через резистор 220 Ом.
Загружаем в модуль ESP8266 скетч из листинга 66.2.
Переключатели на плате Arduino+WiFi необходимо установить следующим образом

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

Листинг 66.2

```
// пин подключения светодиода
int pinLed=4;     // GPIO4

void setup()
{
  // запуск последовательного порта:
  Serial.begin(115200);
  // конфигурация пина светодиода
  pinMode(pinLed,OUTPUT);
  // и выключить
  digitalWrite(pinLed,LOW);
}

void loop()
{
  // ожидание данных из Arduino
  if (Serial.available()) {
    char c=Serial.read();
    if(c=='0') {
       digitalWrite(pinLed,LOW);
       // ответ
       Serial.println("GPIO4=0");
    }
    else if(c=='1') {
       digitalWrite(pinLed,HIGH);
       // ответ
       Serial.println("GPIO4=1");
    }

  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_66_02.zip.**

После загрузки скетча на ESP8266, необходимо перевести плату в режим ATmega328 ← →ESP8266. Переключатели необходимо установить следующим образом

1	2	3	4	5	6	7
ON	ON	OFF	OFF	OFF	OFF	OFF

Теперь Arduino и ESP8266 соединены по последовательному порту. Отправляя с компьютера (из программы Termite) '0' или '1' мы изменяем состояние светодиода на контакте GPIO4 и получаем сообщение из ESP8266. Процесс обмена показан на рис. 66.4.

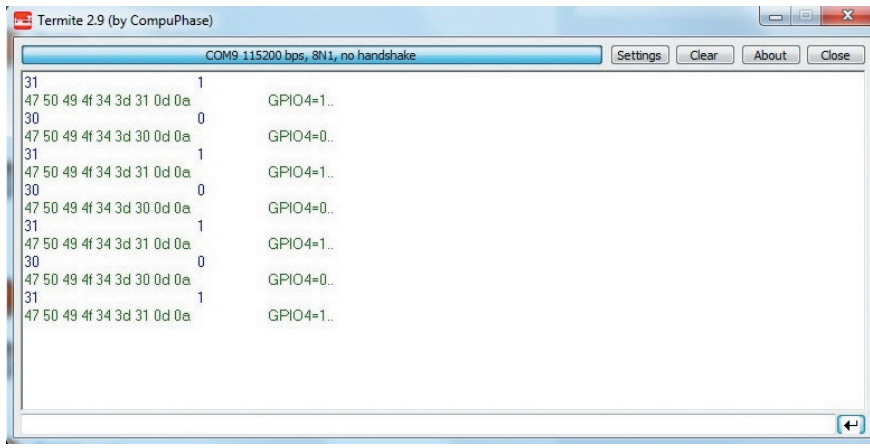


Рис. 66.4. Процесс обмена по последовательному порту между ATmega328 и ESP8266 платы Arduino+WiFi

Режим связи между ATmega328 и ESP8266 позволяет их использовать совместно, например, в проектах Интернет вещей.

Эксперимент 67.
Web-сервер с отображением данных метеостанции

В этом эксперименте создадим web-сервер, для отображения данных метеостанции на датчиках BMP280 и DHT11

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Датчик BMP280 – 1;
- ❑ Датчик DHT11 – 1;
- ❑ USB-TTL адаптер – 1;
- ❑ Провода MM – 15.

В эксперименте 41 мы создавали домашнюю метеостанцию на датчиках BMP280 и DHT11 с выводом данных на ЖК-дисплей. В этом эксперименте данные с датчиков будут отображаться на web-странице сервера, который мы создадим на модуле ESP8266 нашей платы Arduino WiFi. Сами датчики будут подсоединены к плате Arduino, и данные с них будут отправляться по последовательному порту на модуль ESP8266 для отображения на web-странице. Для отладки (вывода данных в последовательный порт компьютера) будем использовать USB-TTL адаптер, подключенный к плате Arduino. Схема соединений показана на рис. 67.1.

Приступим к написанию скетча для платы Arduino. Данные будем отправлять по последовательному порту Serial на модуль ESP8266 в формате:

P=xxx Pa
T=xxxx *C
H=xxxx %$

Для отладки выводим данные в последовательный порт компьютера (со стороны Arduino softwareSerial на контактах 2 и 3).

Загрузим на плату Arduino скетч из листинга 67.1.

Переключатели на плате Arduino+WiFi необходимо установить следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

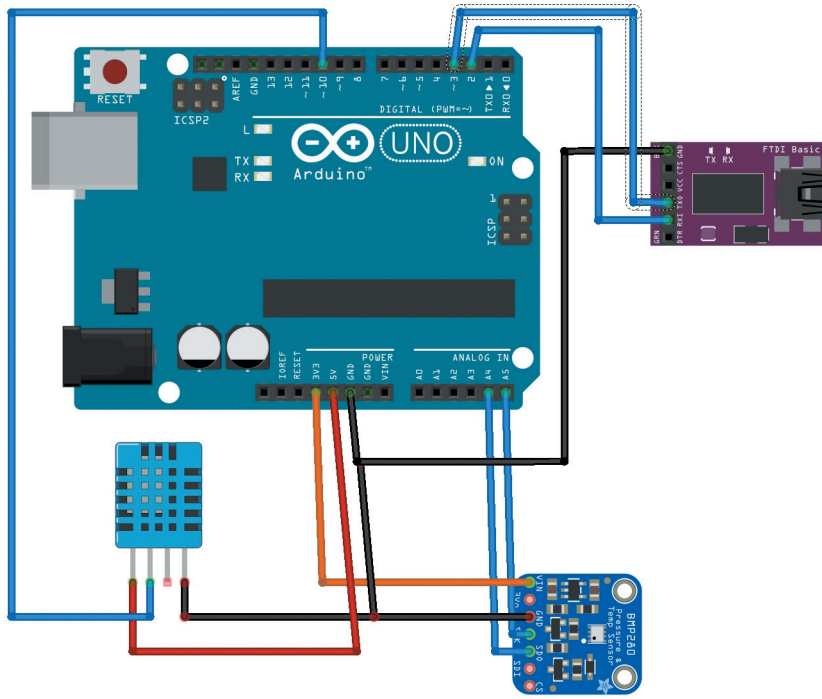


Рис. 67.1. Схема соединений подключения датчиков метеостанции

Листинг 67.1.

```
// подключение библиотек
#include "DHT.h"
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <SoftwareSerial.h>
// пин для подключения датчика DHT
#define DHTPIN 12
// тип датчика DHT
#define DHTTYPE DHT11   // DHT 11
// создание экземпляров
Adafruit_BMP280 bmp;
SoftwareSerial PCSerial(3, 2); // RX, TX
DHT dht(DHTPIN, DHTTYPE);
// для опроса
unsigned long millist=0;
//
```

```
String str="";

void setup()
{
  // запуск последовательного порта
  Serial.begin(115200);
  // запуск порта для USB-TTL
  PCSerial.begin(115200);
  // запуск датчиков DHT,BMP20
  dht.begin();
  bmp.begin();
}

void loop() {
  if(millis()-millist>=30000)
     {
     // получение данных
     int h = dht.readHumidity();
     int t = bmp.readTemperature();
     int p = bmp.readPressure();
     // отправка в serial
     // отправка в serial
     str="P="+String(p)+" Pa<br>";
     str+="T="+String(t)+" *C<br>";
     str+="H="+String(h)+" %$";
     Serial.print(str);     Serial.print(str);
     // отправка в sowtwareserial
     PCSerial.println(str);
     // новый отсчет 30 сек
     millist=millis();
     }
  // получение ESP8266 -- Arduino
  if (Serial.available()) {
    PCSerial.write(Serial.read());
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке

https://arduino-kit.ru/scetches exp_67_01.zip.

Загружаем скетч на плату Arduino, открываем последовательные порты Serial (монитор последовательного порта Arduino IDE) и softwareSerial (Termite). Видим вывод показаний атмосферного давления, температуры и влажности (рис. 67.2).

После проверки можно закомментировать строку

```
//PCSerial.println(str);
```

Теперь создаем скетч для ESP8266. Данные, поступающие из Arduino, будем сохранять в переменных p, t, h. При обращении клиента к серверу будем выводить страницу с показаниями переменных p, t, h.

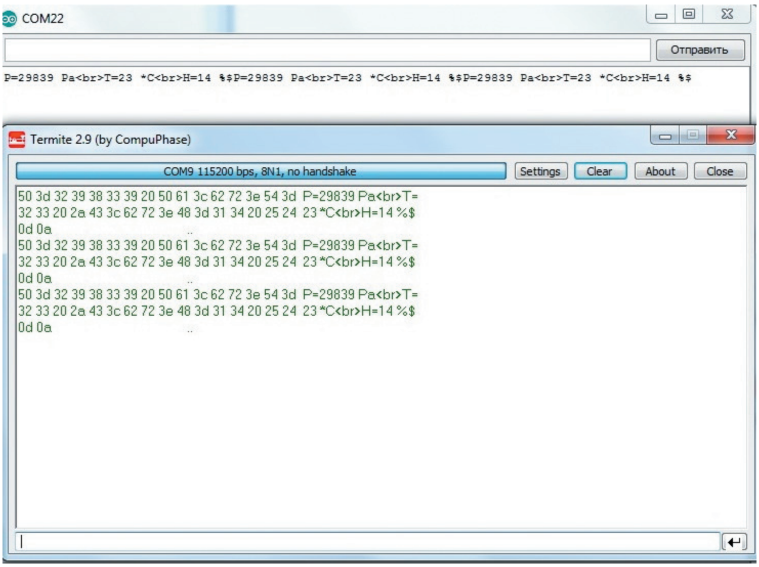


Рис. 67.2. Вывод показание в serial и softwareSerial

Содержимое скетча показано в листинге 67.2.

Для загрузки скетча в ESP 8266 переключатели на плате Arduino+WiFi устанавливаем следующим образом:

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

Листинг 67.2.

```
// подключение библиотек
#include <ESP8266WiFi.h>
// данные SSID и пароль точки доступа
const char* ssid     = "Kassa1";
const char* password = "12345678";
// создание объекта сервера
WiFiServer server(80);
 // данные, пришедшие из последовательного порта
String inputString = "";
String inputString1 = "";
// строка пришла
boolean stringComplete = false;
// пин подключения светодиода
int pinLed=4;     // GPIO4
```

```
void setup(void){
  // конфигурация пина светодиода
  pinMode(pinLed,OUTPUT);
  // и выключить
  digitalWrite(pinLed,LOW);
  delay(5000);
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Подсоединение к точке доступа
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  digitalWrite(pinLed,HIGH);

  // запуск сервера
  server.begin();
  Serial.println("HTTP server started");
  // резервирование 50 bytes для inputString:
  //inputString.reserve(50);
}

void loop(void){
    // проверка прихода строки из последовательного порта
    serialEvent1();
    if (stringComplete) {
       inputString1=inputString;
       Serial.println(inputString1);
       // очистить строку
       inputString = "";
       stringComplete = false;
    }
    // подключение клиента
    WiFiClient client = server.available();
    if (client) {
      Serial.println("new client");
      // конец запроса
      boolean currentLineIsBlank = true;
      while (client.connected()) {
        if (client.available()) {
```

```
            char c = client.read();
            Serial.write(c);
            if (c == '\n' && currentLineIsBlank) {
              // отправить заголовки и страницу
              client.println("HTTP/1.1 200 OK");
              client.println("Content-Type: text/html");
              client.println("Connection: close");
              client.println("Refresh: 5");
              client.println();
              client.println("<!DOCTYPE HTML>");
              client.println("<html>");
              client.println(inputString1);
              client.println("</html>");
              break;
            }
            if (c == '\n') {
              currentLineIsBlank = true;
            } else if (c != '\r') {
              currentLineIsBlank = false;
            }
          }
        }
        delay(1);
        // закрыть соединение
        client.stop();
        Serial.println("client disconnected");
      }

}
// получение строки по Serial
void serialEvent1() {
  boolean flag1=false;

  while (Serial.available() && flag1==false) {
    // получить байт:
    char inChar = (char)Serial.read();
    Serial.write(inChar);
    if (inChar == '$') {
      stringComplete = true;
      flag1=true;
    }
    else   // добавление в строку
      inputString += inChar;
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_67_02.zip.**

После загрузки скетча на ESP8266, необходимо перевести плату в режим ATmega328 ← →ESP8266. Переключатели необходимо установить следующим образом:

1	2	3	4	5	6	7
ON	ON	OFF	OFF	OFF	OFF	OFF

Теперь Arduino и ESP8266 соединены по последовательному порту.

Заходим из браузера по адресу сервера и видим страницу с показаниями датчиков (рис. 67.3).

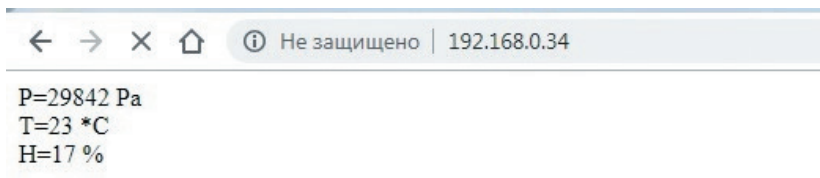


Рис. 67.3. Страница с показаниями датчиков, выдаваемая сервером

Данные обновляются каждые 5 секунд

Эксперимент 68.
Web-сервер на ESP8266 для управления светодиодами

В этом эксперименте создадим web-сервер и страницу, с которой будем управлять светодиодами, подключенными к модулю ESP8266

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Светодиод – 2;
- ❑ Резистор 220 Ом – 2;
- ❑ Провода MF – 3.

В этом эксперименте мы через web-страницу сервера будем управлять (включать/выключать) светодиодами, подключенными к контактам ESP8266.

Переключатели на плате Arduino+WiFi необходимо установить следующим образом:

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

Схема соединений показана на рис. 68.1.

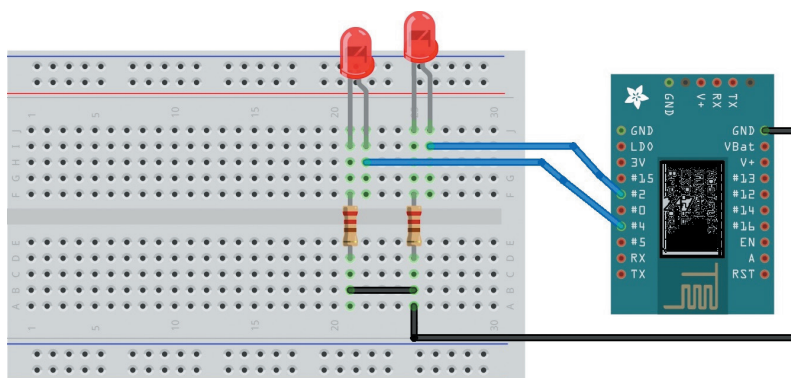


Рис. 68.1. Схема соединений подключения светодиодов к модулю ESP8266

Подключение проводим согласно карте контактов ESP8266, которая показана на рис. 68.2.

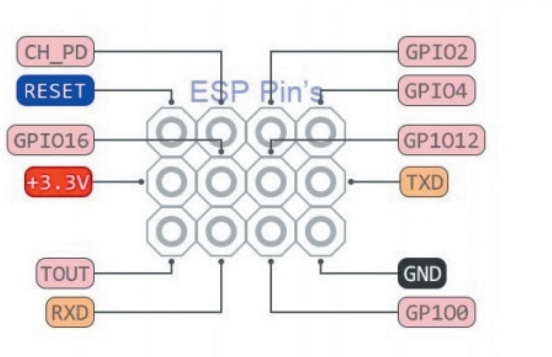


Рис. 68.2. Карта контактов ESP8266 платы Arduino+WiFi

Приступаем к написанию скетча. Необходимо подключить ESP8266 к точке доступа и запустить на нем web-сервер. При обращении к серверу формируем страницу, на которой находится форма, два элемента checkbox для выбора состояния светодиодов и скрытая кнопка отправки формы. Код формы на языке html показан в листинге 68.1.

Листинг 68.1.

```
<!DOCTYPE HTML>
<html>
<form name='form1' id='form1' action='' method='GET'>
<br> Led 1 <input type='checkbox' name='led1'
    onchange='document.getElementById("send1").click();'>
<br> Led 2 <input type='checkbox' name='led2'
   onchange='document.getElementById("send1").click();'>
<br> <input type='submit' name='send1' id='send1'
   value='send1' style="visibility : hidden" >
</form>
</html>
```

При изменении состояния checkbox-ов по событию onchange вызываем нажатие кнопки и отправку данных формы на сервер методом GET. На сервере делаем анализ пришедших данных и изменение состояния светодиодов. Далее идет формирование страницы с измененными состояниями элементов checkbox.

Содержимое скетча для Arduino IDE показано в листинге 68.2.

Листинг 68.2.

```
// подключение библиотек
#include <ESP8266WiFi.h>
// данные SSID и пароль точки доступа
const char* ssid     = "******";
const char* password = "********";
// создание объекта сервера
WiFiServer server(80);
 // данные, пришедшие из последовательного порта
String inputString = "";
String inputString1 = "";
// строка пришла
boolean stringComplete = false;
// состояние checkbox для led
String checked1="";
String checked2="";
// пин подключения светодиодов
int pinLed1=4;    // GPIO4
int pinLed2=2;    // GPIO2

void setup(void){
  // конфигурация пина светодиодов
  pinMode(pinLed1,OUTPUT);
  pinMode(pinLed2,OUTPUT);
  // и выключить
  digitalWrite(pinLed1,LOW);
  digitalWrite(pinLed2,LOW);
  delay(5000);
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Подсоединение к точке доступа
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());


  // запуск сервера
  server.begin();
  Serial.println("HTTP server started");
```

```
}

void loop(void){

    // подключение клиента
    WiFiClient client = server.available();
    if (client) {
      Serial.println("new client");
      // конец запроса
      boolean currentLineIsBlank = true;
      // собираем данные
      String data="";
      while (client.connected()) {
        if (client.available()) {
          char c = client.read();
          data+=c;
          Serial.write(c);
          if (c == '\n' && currentLineIsBlank) {
            // анализ данных запроса и установка светодиодов
            if(data.indexOf("led1=on")!=-1) {
               digitalWrite(pinLed1,HIGH);checked1="checked";
            }
            else {
               digitalWrite(pinLed1,LOW);checked1="";
            }
            if(data.indexOf("led2=on")!=-1) {
               digitalWrite(pinLed2,HIGH);checked2="checked";
            }
            else {
               digitalWrite(pinLed2,LOW);checked2="";
            }
            // отправить заголовки
            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println("Connection: close");
            client.println();
            // отправить страницу
            client.println("<!DOCTYPE HTML>");
            client.println("<html>");
            client.println("<form name='form1' id='form1'
                           action='' method='GET'>");
            client.println("<br> Led 1 <input type='checkbox'
                            name='led1' ");
            client.println(checked1);
            client.println("' onchange='document.getElementById
                           (\"send1\").click();'>");
            client.println("<br> Led 2 <input type='checkbox'
                            name='led2' ");
```

```
            client.println(checked2);
            client.println("' onchange='document.getElementById
                           (\"send1\").click();'>");
            client.println("<br> <input type='submit' name='send1'
                            id='send1' value='send1'
                            style=\"visibility : hidden\" >");
            client.println("</form>");
            client.println("</html>");
            break;
          }
          if (c == '\n') {
            currentLineIsBlank = true;
          } else if (c != '\r') {
            currentLineIsBlank = false;
          }
        }
      }
      delay(1);
      // закрыть соединение
      client.stop();
      Serial.println("client disconnected");
    }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_68_02.zip.**

Загружаем скетч в модуль ESP8266, открываем монитор последовательного порта, ждем, пока модуль ESP8266 подключиться к точке доступа (рис. 68.3) и с браузера обращаемся к серверу. На странице управляем состоянием светодиодов (рис. 68.4).

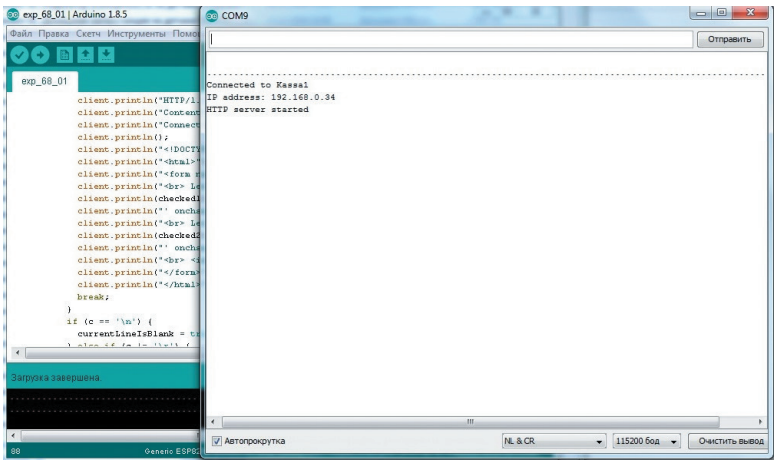


Рис. 68.3. Подключение ESP8266 к WiFi и запуск сервера

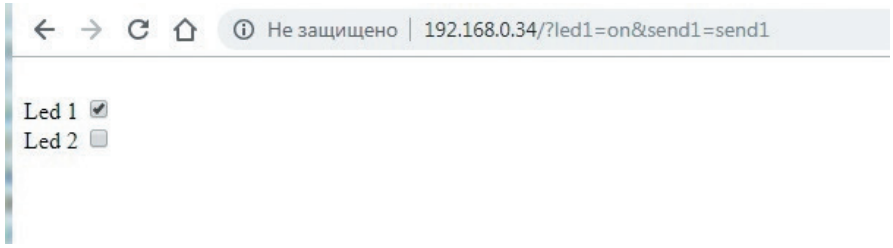


Рис. 68.4. Страница управления светодиодами

Рис. 68.5. Код страницы управления светодиодами

Эксперимент 69.
Web-сервер для управления реле через Arduino

В этом эксперименте создадим на web-сервере страницу, с которой будем управлять реле с помощью Arduino, подключенной к модулю ESP8266 по последовательному порту

В эксперименте мы будем использовать следующие компоненты:

- Плата Arduino UNO – 1;
- Кабель USB – 1;
- Плата прототипирования – 1;
- Релейный модуль – 1;
- Провода MF – 3.

В этом эксперименте мы с web-страницы сервера будем управлять (включать/выключать) реле, подключенным к Arduino, которая обменивается данными с модулем ESP8266 по последовательному порту.

В эксперименте 68 мы создали страницу на сервере для управления светодиодами. В скетч 68_02.ino мы внесем самые минимальные изменения. Необходимо формировать страницу, код которой показан в листинге 69.1.

Листинг 69.1.

```
<!DOCTYPE HTML>
<html>
<form name='form1' id='form1' action='' method='GET'>
<br> Relay 1 <input type='checkbox' name='relay1'
    onchange='document.getElementById("send1").click();'>
<br> Relay 2 <input type='checkbox' name='relay2'
   onchange='document.getElementById("send1").click();'>
<br> <input type='submit' name='send1' id='send1'
   value='send1' style="visibility : hidden" >
</form>
</html>
```

И добавляем код обработки данных, приходящих из браузера и формирования строки для отправки в Arduino:

```
// анализ данных запроса и отправка данных на Arduino
String toArduino="*";
if(data.indexOf("relay1=on")!=-1) {
```

```
    toArduino+="relay1=on;";checked1="checked";
}
else {
    toArduino+="relay1=off;";checked1="";
}
if(data.indexOf("relay2=on")!=-1) {
   toArduino+="relay2=on";checked2="checked";
}
else {
   toArduino+="relay2=off";checked2="";
}
toArduino+="$";
Serial.print(toArduino);
```

Полностью посмотреть код и скачать скетч запуска сервера, выдачи страницы и отправки данных на Arduino можно на сайте Arduino-kit по ссылке

https://arduino-kit.ru/scetches/exp_69_02.zip.

Переключатели на плате Arduino+WiFi необходимо установить следующим образом:

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

И загрузить скетч в модуль ESP8266.

Теперь подключим релейный модуль к Arduino. Схема соединений показана на рис. 69.1.

На Arduino загружаем скетч получения данных по последовательному порту из модуля ESP8266 и изменения состояния реле.

Содержимое скетча DE показано в листинге 69.3.

Листинг 69.3.

```
// пины подключения реле
int pinRelay1=7;
int pinRelay2=6;
// данные, пришедшие из последовательного порта
String inputString = "";
// строка пришла
boolean stringComplete = false;

void setup(){
  // конфигурация пина реле
  pinMode(pinRelay1,OUTPUT);
  pinMode(pinRelay2,OUTPUT);
  // и выключить
  digitalWrite(pinRelay1,HIGH);
  digitalWrite(pinRelay2,HIGH);
  // запуск последовательного порта
```

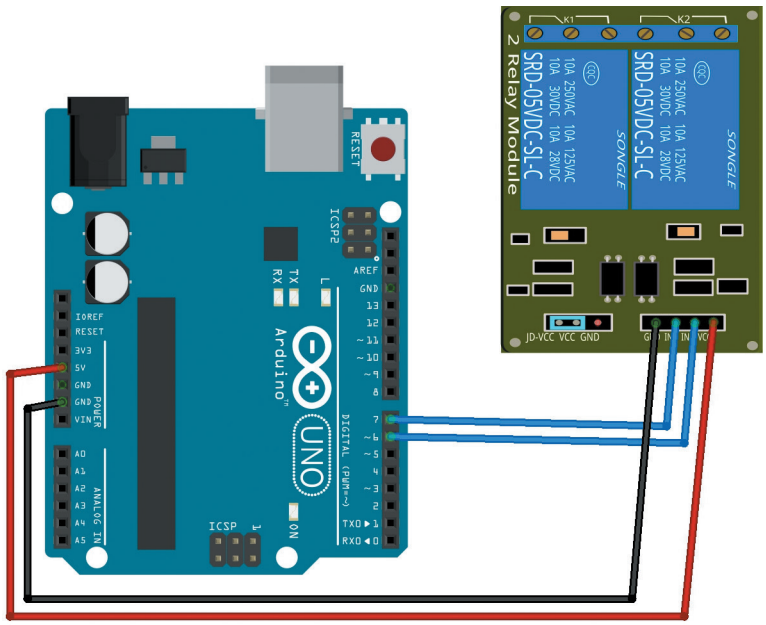


Рис. 69.1. Схема соединений подключения реле к Arduino

```
  Serial.begin(115200);
}

void loop(){
    // проверка прихода строки из последовательного порта
    if (stringComplete) {
       // проверка пришедших данных
       if(inputString.indexOf("relay1=on")!=-1) {
          digitalWrite(pinRelay1,LOW);  // включить
       }
       else {
          digitalWrite(pinRelay1,HIGH);  // выключить
       }
       if(inputString.indexOf("relay2=on")!=-1) {
          digitalWrite(pinRelay2,LOW);  // включить
       }
       else {
          digitalWrite(pinRelay2,HIGH);  // выключить
       }

       // очистить строку
       inputString = "";
       stringComplete = false;
    }
}
// получение строки по Serial
```

```
void serialEvent1() {
  boolean flag1=false;

  while (Serial.available() && flag1==false) {
    // получить байт:
    char inChar = (char)Serial.read();
    Serial.write(inChar);
    if (inChar == '$') {
      stringComplete = true;
      flag1=true;
    }
    else   // добавление в строку
      inputString += inChar;
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_69_03.zip.**

Для загрузки этого скетча на плату Arduino переключатели на плате Arduino+WiFi необходимо установить следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

После загрузки скетча на Arduino, необходимо перевести плату в режим ATmega328 ← →ESP8266. Переключатели необходимо установить следующим образом:

1	2	3	4	5	6	7
ON	ON	OFF	OFF	OFF	OFF	OFF

Теперь Arduino и ESP8266 соединены по последовательному порту.

Заходим из браузера по адресу сервера и видим страницу для изменения состояния реле (рис. 69.2).

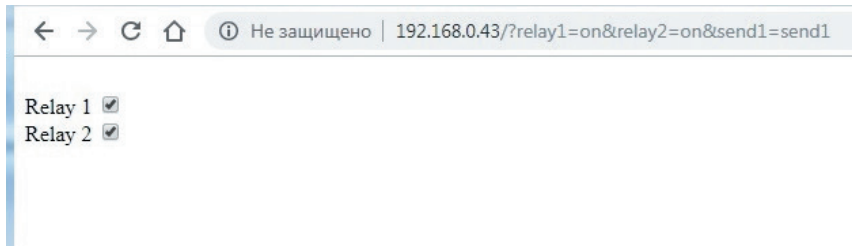


Рис. 69.2. Страница управления реле

Проверяем, изменяя состояние checkbox-ов на странице

Эксперимент 70.
Web-сервер управления текстом для бегущей строки на 4-х разрядной светодиодной матрице

В этом эксперименте создадим web-сервер и создадим страницу, с которой будем управлять текстом и скоростью движения бегущей строки на 4-х разрядной светодиодной матрице

В эксперименте мы будем использовать следующие компоненты:
- Плата Arduino UNO – 1;
- Кабель USB – 1;
- Плата прототипирования – 1;
- 4-х разрадная светодиодная матрица – 1;
- Потенциометр – 1;
- Провода MM – 2;
- Провода MF – 4.

В этом эксперименте будем изменять содержимое "бегущей строки" на 4-х разрядной светодиодной матрице, отправляя данные на Arduino по последовательному порту со страницы web-сервера на ESP8266.

На сервере необходимо формировать страницу, код которой показан в листинге 70.1.

Листинг 69.1.

```
<!DOCTYPE HTML>
<html>
<form name='form1' id='form1' action='' method='GET'>
<br> Text for the ticker <input  name='text1' >
<br> <input type='submit' name='send1' value='Send' >
</form>
</html>
```

В скетче 69_02 изменяем код выдачи страницы и код обработки данных, приходящих из браузера и формирования строки для отправки в Arduino:

```
// анализ данных запроса и отправка данных на Arduino
String toArduino="*";
int n1=data.indexOf("?text1=");
int n2=data.indexOf("&send1");
toArduino+=data.substring(n1+7,n2)+"$";
Serial.print(toArduino+String(n1)+" "+String(n2));
// отправить заголовки и страницу
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
```

```
client.println("Connection: close");
client.println();
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<form name='form1' id='form1' action=''
                method='GET'>");
client.println("<br> Text for the ticker <input  name='text1' >");
client.println("<br> <input type='submit' name='send1'
               value='Send' >");
client.println("</form>");
client.println("</html>");
break;
}
```

Полностью посмотреть код и скачать скетч запуска сервера, выдачи страницы и отправки данных на Arduino можно на сайте Arduino-kit по ссылке

https://arduino-kit.ru/scetches/exp_70_02.zip.

Переключатели на плате Arduino+WiFi необходимо установить следующим образом и загрузить скетч на модуль ESP8266:

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

Теперь подключаем к Arduino 4-х разрядную светодиодную матрицу. Схема соединений показана на рис. 70.1.

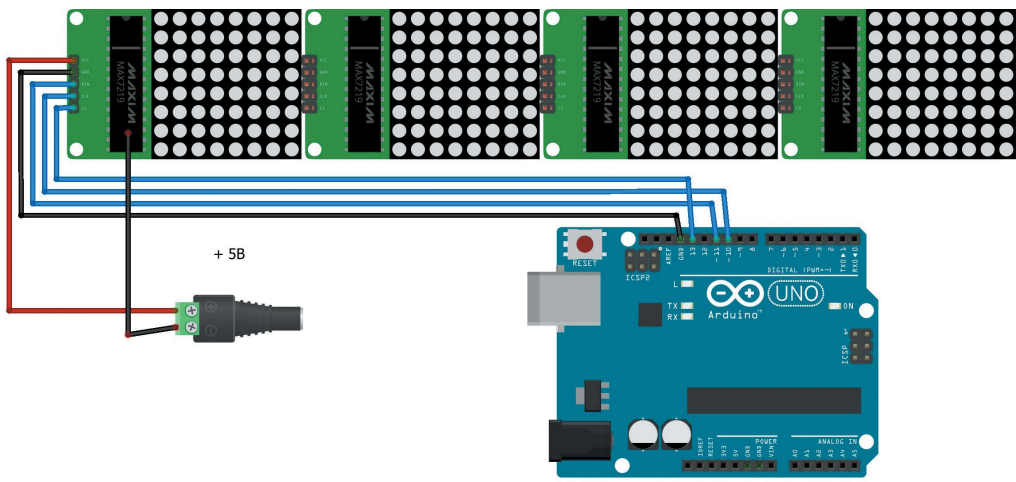


Рис. 70.1. Схема соединения для "бегущей строки" на 4-х разрядной светодиодной матрице

Содержимое скетча генерации бегущей строки и изменение текста по последовательному порту представлено в листинге 70.2.

Листинг 70.3.

```
// подключение библиотек
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>
// пин CS
int pinCS = 10;
// количество матриц по-горизонтали
int numberOfHorizontal = 15;
// количество матриц по-вертикали
int numberOfVertical = 1;
// создание объекта
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontal, number-
OfVertical);
// строка для вывода
String text = "Arduino-KIT 2019";
// текущее смещение от 0
int offset=32;
// значение скорости
int speed1=500;
 // данные, пришедшие из последовательного порта
String inputString = "";
// строка пришла
boolean stringComplete = false;

void setup() {
    // яркость от 0 до 15
    matrix.setIntensity(7);
    // запуск последовательного порта
    Serial.begin(9600);
   // резервирование 30 bytes для inputString:
   inputString.reserve(30);
}

void loop() {
    // проверка прихода строки из последовательного порта
    if (stringComplete) {
       text=inputString;
       offset=32;
       // очистить строку
       inputString = "";
       stringComplete = false;
    }
```

```
    // очистка экрана
    matrix.fillScreen(LOW);
    // вывод строки с позиции offset
    for ( int i = 0 ; i < text.length(); i++ ) {
        matrix.setRotation( i, 1 );
        if(i*6+offset>(-6) && i*6+offset<32) {
            matrix.drawChar(i*6+offset, 0, text[i], HIGH, LOW, 1);
        }
    }
    matrix.write();
    // задержка (скорость)
    delay(speed1);
    // изменение смещения
    offset=offset-1;
    // в начало - позиция 32
    if(offset+text.length()*6==0)
       offset=32;
}
//
void serialEvent() {
  boolean flag1=false;

  while (Serial.available() && flag1==false) {
    // получить байт:
    char inChar = (char)Serial.read();
    if (inChar == '#') {
      stringComplete = true;
      flag1=true;
    }
    else   // добавление в строку
      inputString += inChar;
  }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/ exp_70_03.zip.**

Для загрузки этого скетча на плату Arduino переключатели на плате Arduino+WiFi необходимо установить следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Загружаем скетч на плату Arduino и наблюдаем "бегущую строку". Пробуем менять содержимое отправкой команды из монитора последовательного порта.

После загрузки скетча на Arduino, необходимо перевести плату в режим ATme-

ga328 ← →ESP8266. Переключатели необходимо установить следующим образом

1	2	3	4	5	6	7
ON	ON	OFF	OFF	OFF	OFF	OFF

Теперь Arduino и ESP8266 соединены по последовательному порту.

Заходим из браузера по адресу сервера и видим страницу для ввода текста для "бегущей строки" (рис. 70.2).

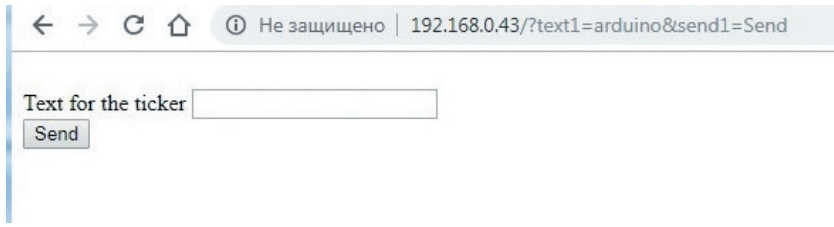


Рис. 70.2. Страница ввода текста для "бегущей строки".

Проверяем, изменяя состояние checkbox-ов на странице.

Эксперимент 71.
Домашняя метеостанция для сервиса Народный мониторинг

В этом эксперименте мы приступим к созданию метеостанции для сервиса Народный мониторинг

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino+WiFi – 1;
- ❑ Кабель USB – 1.

В экспериментах 67-70 мы рассматривали использование нашей платы в качестве WiFi web-сервера. В этом эксперименте будем использовать плату Arduino+WiFi в качестве web-клиента. Рассмотрим отправку данных в сервис Народный мониторинг.

На карте мира сайта Народный мониторинг собираются и отображаются практически в реальном времени показания от различных датчиков среды, установленных как на улице, так и в помещении.

Чтобы стать участником проекта, необходимо зарегистрироваться. Заходим на сайт http://www.narodmon.ru и выбираем пункт меню Вход → Стать участником проекта. В форме вводим адрес электронной почты, куда будут отправлены логин и пароль для входа в профиль (см. рис. 71.1).

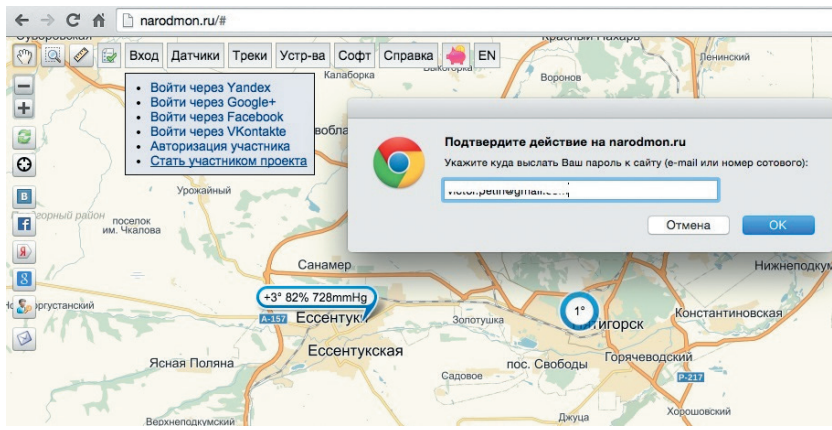


Рис. 71.1. Регистрация в сервисе Народный мониторинг

После регистрации можно добавлять свое устройство на карту. В качестве устройства будем использовать домашнюю метеостанцию на датчиках BMP280 и DHT11 из эксперимента 41.

Сначала необходимо настроить передачу и отображение показаний всех датчиков устройства. Для добавления домашней метеостанции на карту необходимо подключить ее к сети Интернет и настроить передачу показаний датчиков на сайт narodmon.ru с интервалом от 5 до 15 минут.

Для передачи данных на сайт будем использовать самый простой протокол передачи данных – HTTP GET. Для этого необходимо подключить модуль ESP8266 по WiFi к сети Интернет обратиться на сайт Народного мониторинга по следующему адресу:

http://narodmon.ru/get?ID=MAC&mac1=value1&...&macN=valueN

где

- ID – уникальный адрес своего устройства;
- mac1, … macnN – название датчиков устройства;
- value1, … valueN – показания датчиков.

В качестве ID будем использовать число, отражающие географические координаты нашей метеостанции, например:

ID= 440365430747 (широта – 44.0365, долгота – 43.0747)

Для названия датчиков будем использовать следующие метрики:

H1 – для относительной влажности (датчик DHT11);

T1 – для температуры (датчик BMP280);

P1 – для атмосферного давления (датчик BMP280).

Тогда адрес отправки данных для нашей метеостанции будет следующий:

http://narodmon.ru/get?ID=440365430747&H1=valueh&T1=valuet&P1=valuep

Строку

/get?ID=440365430747&H1=valueh&T1=valuet&P1=valuep

модуль ESP8266 будет получать из Arduino по последовательному порту. Скетч получения модулем данных из последовательного порта и отправка данных на сайт Народного мониторинга показан в листинге 71.1.

Листинг 71.1.

```
// подключение библиотек
#include <ESP8266WiFi.h>
// данные SSID и пароль точки доступа
const char* ssid     = "Kassa1";
const char* password = "12345678";
// создание объекта клиента
WiFiClient client;
 // данные, пришедшие из последовательного порта
```

```
String inputString = "";
String inputString1 = "";
// строка пришла
boolean stringComplete = false;
// сервер Народного мониторинга
char server[] = "www.narodmon.ru";
// массив для получения данных от сервера
char response[40];

void setup(void){
  delay(5000);
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  Serial.print("*");

  // Подсоединение к точке доступа
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop(void){
    serialEvent1();
    if (stringComplete) {
       // отправить данные на сайт
       sendDataNarodmon(inputString);
       // очистить строку
       inputString = "";
       stringComplete = false;
    }
}
// получение строки по Serial
void serialEvent1() {
  boolean flag1=false;

  while (Serial.available() && flag1==false) {
    // получить байт:
    char inChar = (char)Serial.read();
    Serial.write(inChar);
    if (inChar == '$') {
      stringComplete = true;
      flag1=true;
```

```
      }
      else   // добавление в строку
        inputString += inChar;
    }
  }

  // отправка данных на сайт Народного мониторинга
  void sendDataNarodmon(String str) {
     if (client.connect(server, 80)) {
        //// отправка данных на сервер narodmon.ru
        Serial.print(str);
        client.println("GET "+str+" HTTP/1.1");
        client.println("Host: www.narodmon.ru");
        client.println("Connection: close");
        client.println();
        // получение ответа
        unsigned long previos=millis();
        for(int i=0;i<40;i++)
           response[i]=0;
        int x=0;int f=0;
        do{
           if(client.available() > 0) {
              // получать данные из ESP8266
              char s = client.read();
              if(s=='#')
                 f=1;
              if(f==1) {
                 response[x]=s;
                 Serial.print(response[x]);
                 x++;
              }
           Serial.write(s);
           }
        }
        while((millis() - previos) < 5000);
        Serial.println(response);
        client.stop();
     }
     else {
       // нет соединения
       Serial.println("connection failed");
       client.stop();
     }

  }
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_71_01.zip.

Переключатели на плате Arduino+WiFi необходимо установить следующим образом:

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

И загрузить скетч на модуль ESP8266.

Открываем монитор последовательного порта и пробуем отправлять данные (см. рис. 71.2).

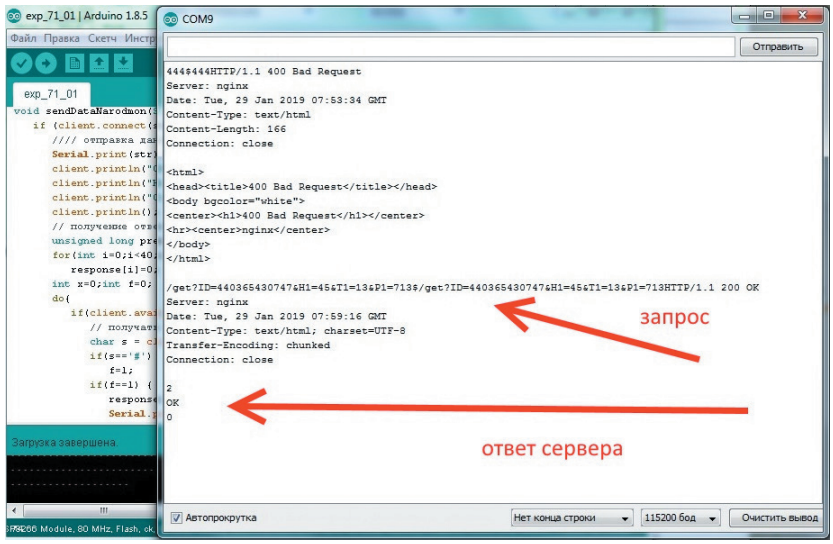


Рис. 71.2. Получение данных по последовательному порту и отправка на сайт Народного мониторинга

При отправке строки

/get?ID=440365430747&H1=45&T1=13&P1=713$/get?ID=440365430747&H1=45&T1=13&P1=713

Приходит ответ OK. Значит, данные успешно отправлены. Переходим к следующему этапу – добавлению устройства в свой профиль для отображения на карте.

Переходим на сайт http://narodmon.ru, авторизуемся и открываем пункт меню Датчики → Мои Датчики → Добавить устройство, где необходимо ввести ID твоего (нашего?) устройства (рис. 71.3).

В появившемся окне выбераем тип данных и название для каждого из датчика, установливаем доступ к показаниям для каждого датчика (публичный или приватный), выполняем привязку к карте, указав полный адрес его размещения или геокоординаты (рис. 71.4).

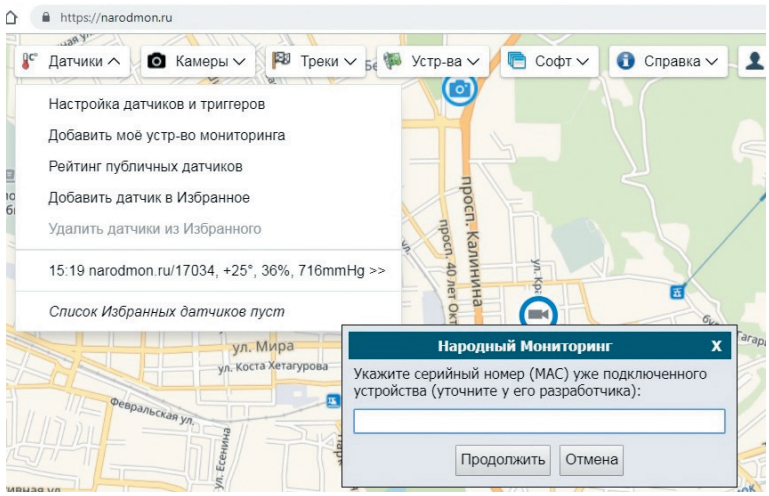


Рис. 71.3. Ввод ID устройства

Внимание !!! Установка публичного доступа возможна только для уличных датчиков.

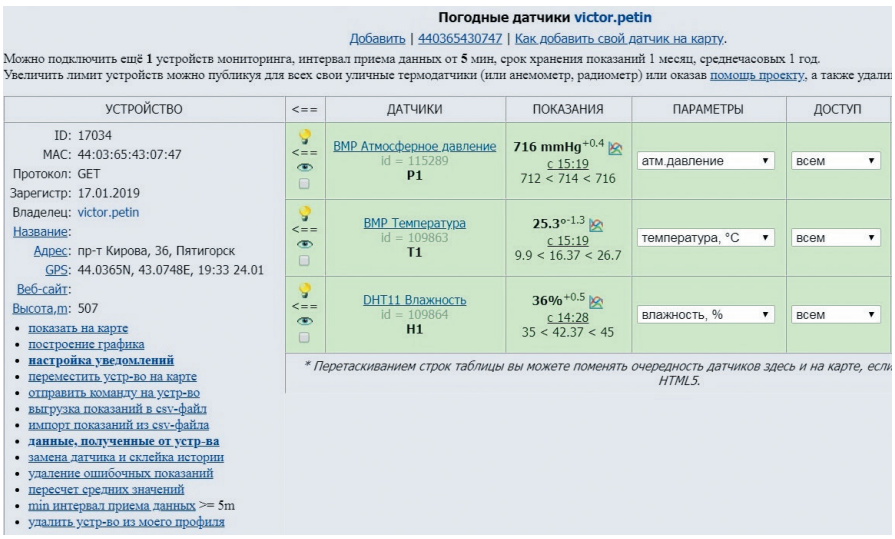


Рис. 71.4. Редактирование данных устройства

После этого устройство появится на карте (рис. 71.5).

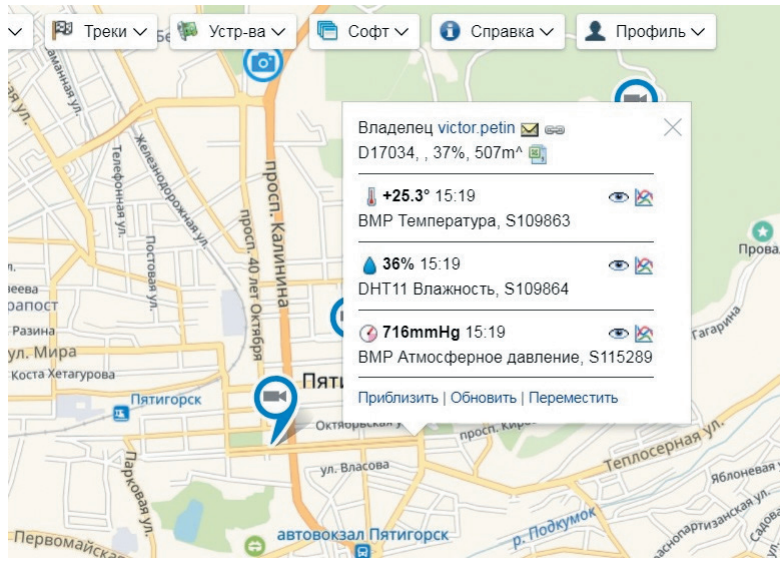


Рис. 71.5. Устройство на карте Народного мониторинга.

В следующем эксперименте организуем отправку реальных данных домашней метеостанции.

Эксперимент 72.
Отправка данных датчиков домашней метеостанции на сайт Народного мониторинга

В этом эксперименте соберем домашнюю метеостанцию, обработаем данные с датчиков и отправим их на модуль ESP8266 для сайта Народного мониторинга

В эксперименте мы будем использовать следующие компоненты:
- Плата Arduino UNO – 1;
- Кабель USB – 1;
- Плата прототипирования – 1;
- Датчик BMP280 – 1;
- Датчик DHT11 – 1;
- Провода MM – 8.

В эксперименте 71 мы настроили отправку данных, получаемых с последовательного порта, на сайт Народного мониторинга. Строку с данными, которые будем получать с датчиков, будем формировать на Arduio, а затем отправлять по последовательному порту на модуль ESP8266.

Схема соединений показана на рис. 72.1.

Нам необходимо каждые 5 минут получать показания с датчиков BMP280 и DHT11 и формировать строку вида

/get?ID=440365430747&H1=valueh&T1=valuet&P1=valuep*

Затем отправлять эту строку по последовательному порту в модуль ESP8266. С датчиками BMP280 и DHT11 мы работали в экспериментах 39, 41.

Содержимое скетча показано в листинге 72.1.

Листинг 72.1.

```
// подключение библиотек
#include "DHT.h"
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
```

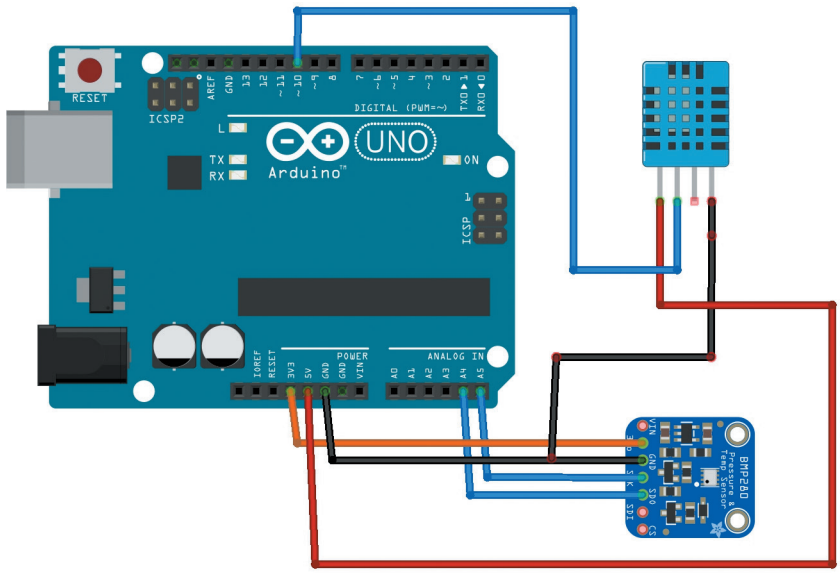


Рис. 72.1. Схема соединений для метеостанции на датчиках BMP180 и DHT11

```
// пин для подключения датчика DHT
#define DHTPIN 10
// тип датчика DHT
#define DHTTYPE DHT11   // DHT 11
// создание экземпляров
Adafruit_BMP280 bmp;
DHT dht(DHTPIN, DHTTYPE);
// для опроса
unsigned long millissend=0;
unsigned long pauseseconds=10;
// ID - устройства в Народном мониторинге
String IDstr="441369430175";

void setup()
{
  // подключение последовательного порта
  Serial.begin(115200);
  // запуск датчиков DHT,BMP20
  dht.begin();
  bmp.begin();
}

void loop() {
  // ждем время паузы
  if(millis()-millissend>=1000) {
     pauseseconds--;
```

```
        //Serial.print("time to send - ");Serial.println(pauseseconds);
        if(pauseseconds==0) {
           // получение данных
           int h = dht.readHumidity();
           int t = bmp.readTemperature();
           float p = bmp.readPressure()/133.32;
           // формирование строки
           String str="/get?ID="+IDstr;
           str=str+"&H1="+String(h);
           str=str+="&T1="+String(t);
           str=str+="&P1="+String(p);
           str=str+="*";
           // отправка в последовательный порт
           Serial.print(str);
           //
           pauseseconds=300;
           }

        millissend=millis();
    }
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/ exp_72_01.zip.**

Переключатели на плате Arduino+WiFi необходимо установить следующим образом и загрузить скетч на Arduino

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Открываем монитор последовательного порта и видим отправку в последовательный порт данных каждые 5 минут (первая отправка через 10 секунд) (см. рис. 72.2).

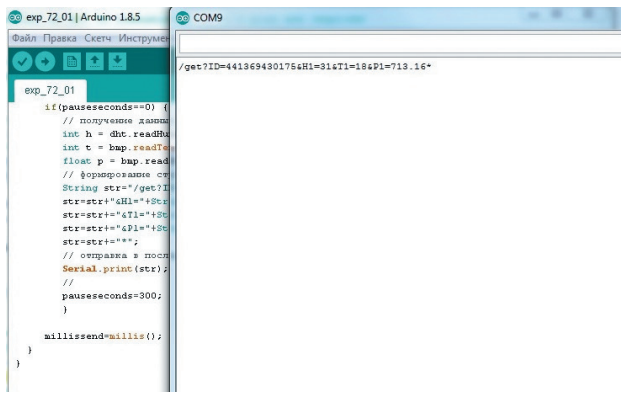


Рис. 72.2. Отправка данных по последовательному порту.

Далее необходимо перевести плату в режим ATmega328 ←→ESP8266. Переключатели необходимо установить следующим образом:

1	2	3	4	5	6	7
ON	ON	OFF	OFF	OFF	OFF	OFF

Теперь Arduino и ESP8266 соединены по последовательному порту.

Через некоторое время проверяем отправку данных на сайт Народного мониторинга.

Заходим на сайт Народного мониторинга в меню Профиль → Мои Датчики и в открывшемся окне переходим по ссылке данные полученные от устройства (см. рис. 72.3).

Полученные показания датчиков / Latest Sensors Readings, UTC+3, find=''

```
2019-01-30 11:26:35  185.75.6.102  GET  ID=440365430747&H1=45&T1=12.50&P1=716.65
2019-01-30 11:21:30  185.75.6.102  GET  ID=440365430747&H1=45&T1=12.00&P1=716.59
2019-01-30 11:16:24  185.75.6.102  GET  ID=440365430747&H1=45&T1=12.10&P1=716.65
2019-01-30 11:11:19  185.75.6.102  GET  ID=440365430747&H1=45&T1=12.30&P1=716.68
2019-01-30 11:06:13  185.75.6.102  GET  ID=440365430747&H1=45&T1=11.90&P1=716.71
2019-01-30 11:01:08  185.75.6.102  GET  ID=440365430747&H1=45&T1=11.70&P1=716.80
```

[Вывести данные за предыдущий час >>]

Рис. 72.3. Данные полученные от устройства

Данные отображаются на карте (рис. 72.4), можно посмотреть и графики по каждому датчику (72.5).

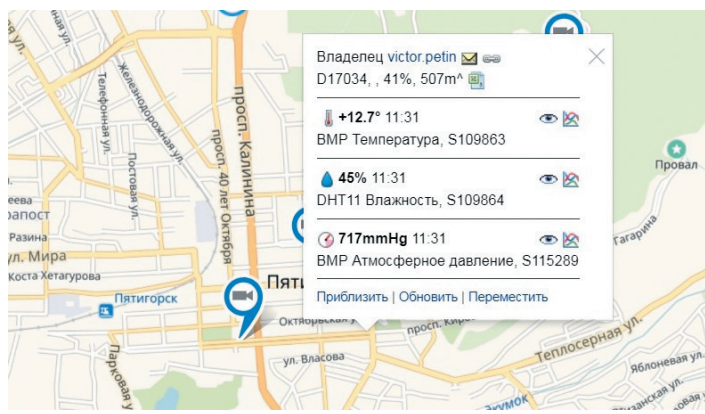


Рис. 72.4. Данные полученные от устройства

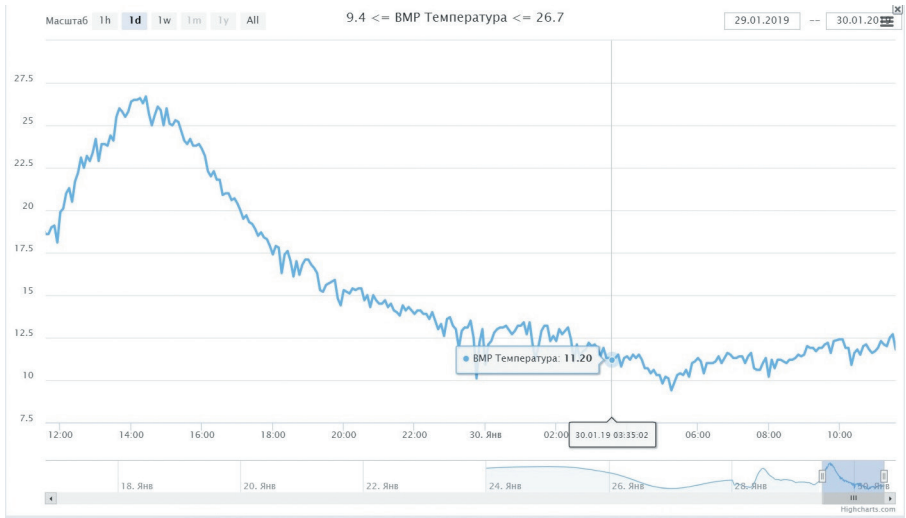


Рис. 72.5. График данных датчика

Эксперимент 73.
Прием на устройстве команд, отправленных с сайта Народного мониторинга

В этом эксперименте мы рассмотрим отправку команд из сервиса Народный мониторинг на устройство

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino+WiFi – 1;
- ❑ Кабель USB – 1.

Сервис Народный мониторинг предоставляет возможность отправлять данные с сайта на устройство, что позволяет подключить к домашней метеостанции через реле исполнительные устройства и управлять ими через Интернет. Сначала определим список команд, которые будем отправлять с сайта Народный мониторинг, например:

- **relay1=1 – включить реле1;**
- **relay1=0 – выключить реле1;**
- **relay2=1 – включить реле2;**
- **relay2=0 – выключить реле2.**

Набрать данные команды можно из своего профиля (пункт меню Профиль → Мои датчики), кликнув на ссылку Отправить команду на устр-во. В появившемся окне вводим команду (рис. 73.1)

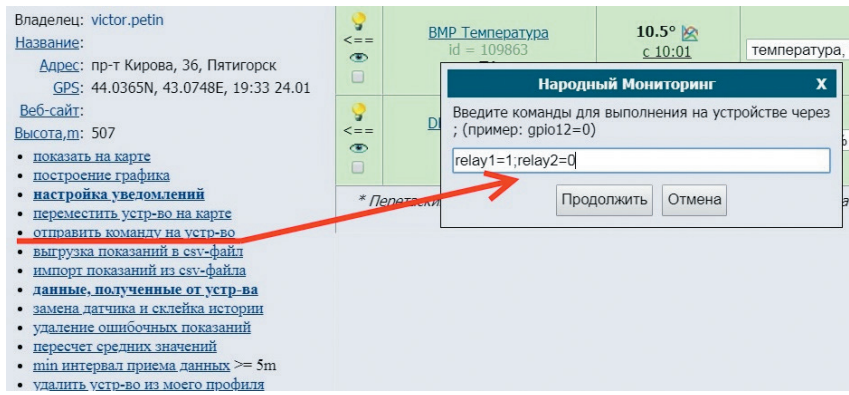


Рис. 73.1. Ввод списка команд для отправки на устройство

Команда появится в очереди на исполнение (рис. 73.2) и при очередном получении данных от наших датчиков, будет отправлена как ответ сервера

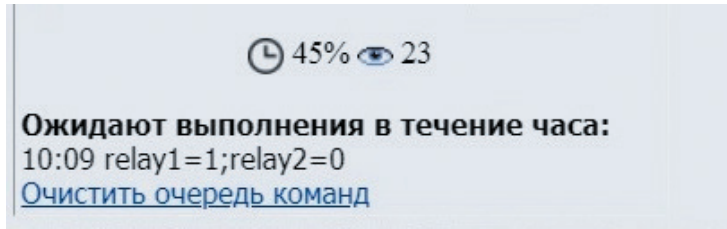


Рис. 73.2. Очередь команд

Проверим получение ответа от сервера при отправке данных.

Переключатели на плате Arduino+WiFi необходимо установить следующим образом

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

И загрузить на модуль ESP8266 скетч из листинга 71.1.

На сайте Народный мониторинг вводим список команд для отправки на устройство (рис. 73.1).

Открываем монитор последовательного порта и пробуем отправлять данные. Набираем строку:

/get?ID=440365430747&H1=45&T1=13&P1=713$/get?ID=440365430747&H1=45&T1=13&P1=713

При этом на сервер отправляются введенные нами данные. После чего смотрим ответ сервера (см. рис. 73.3).

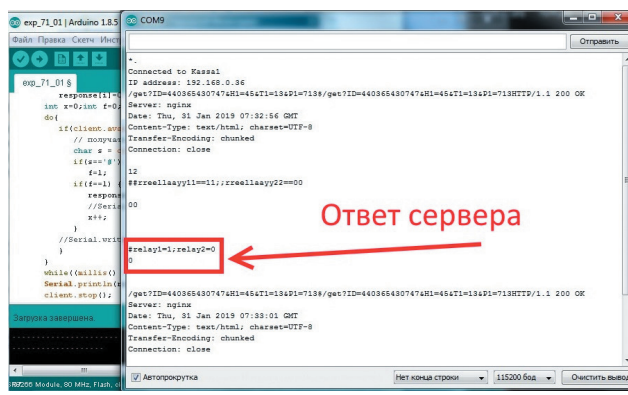


Рис. 73.3. Получение данных по последовательному порту, отправка на сайт Народного мониторинга и получение ответа

Данный ответ от сервера, добавив символ '$', будем отправлять по последовательному порту на Arduino. Из скетча 71_01 ноебходимо убрать вывод в последовательный порт отладочной информации. Скачать необходимый скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/ exp_73_01.zip.**

Управлять устройствами отправкой команд с сайта не совсем удобно. К счастью есть возможность упростить управление с помощью телефона или планшета с операционной системой Android. Для этого необходить установить на свой телефон (или планшет) приложение Народный мониторинг 2018 из Play Market (рис. 73.4).

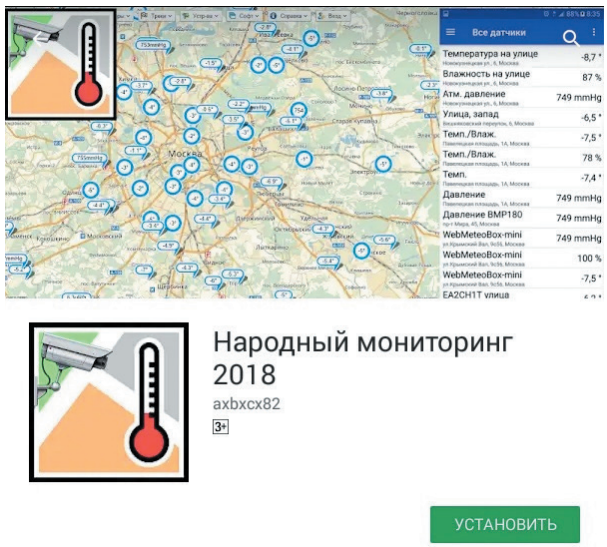


Рис. 73.4. Приложение Народный мониторинг 2018 в Play Market

После установки приложения, запускаем его, авторизуемся с данными своего профиля в Народном мониторинге и через некоторое время получаем доступ к данным датчиков своей домашней метеостанции (рис. 73.5, 73.6)

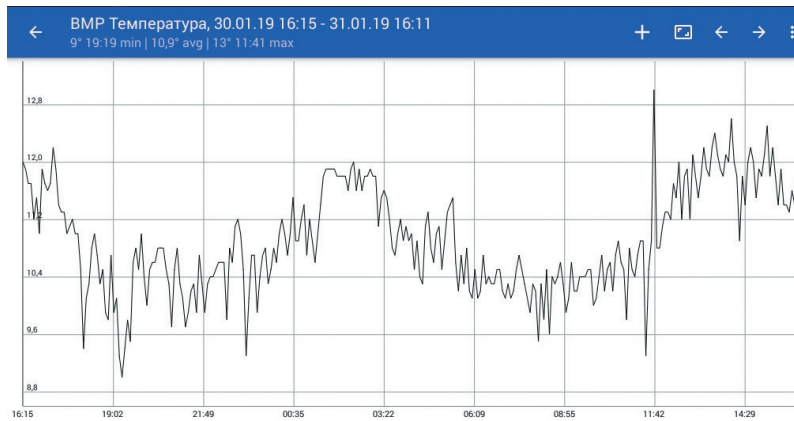


Рис. 73.5, 73.6. Данные домашней метеостанции в мобильном приложении Народный мониторинг 2018

Рассмотрим, как организовать в приложении отправку команд управления. Выбираем пункт меню Управление и создаем кнопки для отправки команд (рис. 73.7, 73.8)

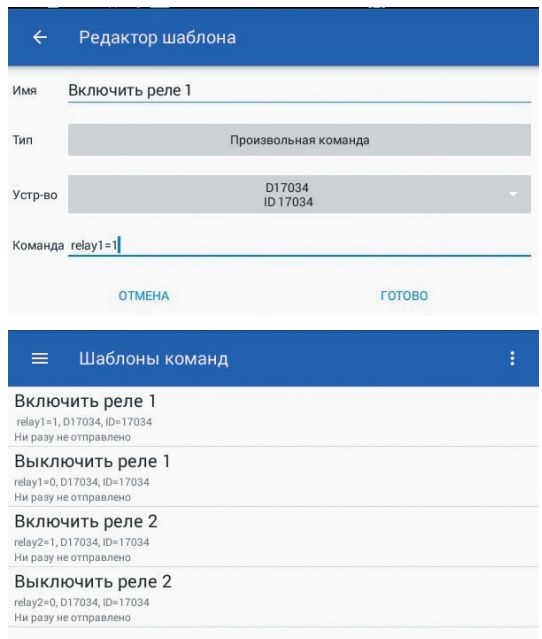


Рис. 73.7, 73.8. Создание кнопок интерфейса для отправки команд

Теперь можно управлять контактами реле домашней метеостанции с телефона.

Эксперимент 74.
Обработка и исполнение команд, полученных с сайта Народный мониторинг

В этом эксперименте рассмотрим прием и исполнение на домашней метеостанции команд управления отправляемых с сайта Народного мониторинга

В эксперименте мы будем использовать следующие компоненты:
- ❑ Плата Arduino+WiFi – 1;
- ❑ Кабель USB – 1;
- ❑ Плата прототипирования – 1;
- ❑ Датчик BMP280 – 1;
- ❑ Датчик DHT11 – 1;
- ❑ Релейный модуль– 1;
- ❑ Провода MM – 8.

В эксперименте 73 мы настроили отправку с сайта Народного мониторинга и получение на модуле ESP8266 команд управления. Теперь рассмотрим перенаправление по последовательному порту данных команд на Arduino, также их исполнение. В домашнюю метеостанцию из эксперимента 72 добавим релейный модуль, к которому можно потом подключить исполнительные устройства, например вентилятор или лампу освещения.

Схема соединений показана на рис. 74.1.

На Arduino необходимо организовать прием и анализ команд, приходящих по последовательному порту. В скетч из эксперимента 72 добавляем сбор в строковую переменную inputString. По получению символа '$' строка скомплектована и вызывается функция command() для поиска ожидаемых команд:
- relay1=1 – включить реле1;
- relay1=0 – выключить реле1;
- relay2=1 – включить реле2;
- relay2=0 – выключить реле2.

Содержимое скетча показано в листинге 74.1.

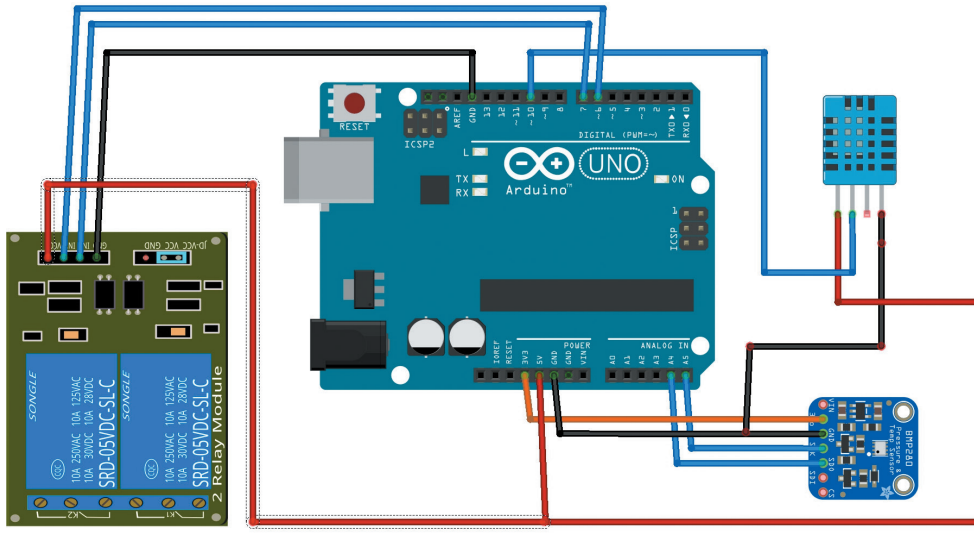


Рис. 74.1. Схема соединений для метеостанции на датчиках BMP280 и DHT11

Листинг 74.1.

```
// подключение библиотек
#include "DHT.h"
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
// пин для подключения датчика DHT
#define DHTPIN 10
// тип датчика DHT
#define DHTTYPE DHT11   // DHT 11
// создание экземпляров
Adafruit_BME280 bmp;
DHT dht(DHTPIN, DHTTYPE);
// для опроса
unsigned long millissend=0;
unsigned long pauseseconds=10;
// ID - устройства в Народном мониторинге
String IDstr="441369430175";
// контакты подключения реле
const int pinRelay1=7;
const int pinRelay2=6;
// данные, пришедшие из последовательного порта
String inputString = "";
// строка пришла
boolean stringComplete = false;
```

```
void setup()
{
  // подключение последовательного порта
  Serial.begin(115200);
   // конфигурация пинов
   pinMode(pinRelay1,OUTPUT);
   pinMode(pinRelay2,OUTPUT);
   // начальная установка
   digitalWrite(pinRelay1,HIGH);
   digitalWrite(pinRelay2,HIGH);

  // запуск датчиков DHT,BMP20
  dht.begin();
  bmp.begin();
}

void loop() {
    // проверка прихода строки из последовательного порта
    if (stringComplete) {
       command();
       // очистить строку
       inputString = "";
       stringComplete = false;
    }

  // ждем время паузы
  if(millis()-millissend>=1000) {
     pauseseconds--;
     if(pauseseconds==0) {
        // получение данных
        int h = dht.readHumidity();
        int t = bmp.readTemperature();
        float p = bmp.readPressure()/133.32;
        // формирование строки
        String str="/get?ID="+IDstr;
        str=str+"&H1="+String(h);
        str=str+="&T1="+String(t);
        str=str+="&P1="+String(p);
        str=str+="*";
        // отправка в последовательный порт
        Serial.print(str);
        //
        pauseseconds=300;
        }

     millissend=millis();
```

```
    }
  }
  // получение данных по последовательному порту
  void serialEvent() {
    boolean flag1=false;

    while (Serial.available() && flag1==false) {
      // получить байт:
      char inChar = (char)Serial.read();
      if (inChar == '$') {
        stringComplete = true;
        flag1=true;
      }
      else   // добавление в строку
        inputString += inChar;
    }
  }
  // проверка пришедших с сервера команд
  void command() {
       //
      if(inputString.indexOf("relay1=1")!=-1) {
         digitalWrite(pinRelay1,LOW);
      }
      if(inputString.indexOf("relay1=0")!=-1) {
         digitalWrite(pinRelay1,HIGH);
      }
      if(inputString.indexOf("relay2=1")!=-1) {
         digitalWrite(pinRelay2,LOW);
      }
      if(inputString.indexOf("relay2=0")!=-1) {
         digitalWrite(pinRelay2,HIGH);
      }
  }
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_74_01.zip.**

Переключатели на плате Arduino+WiFi необходимо установить следующим образом и загрузить скетч на Arduino.

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

Открываем монитор последовательного порта, где видим отправку в последовательный порт данных каждые 5 минут, и пробуем отправлять команды, которвые приводят к установке реле (см. рис. 74.2).

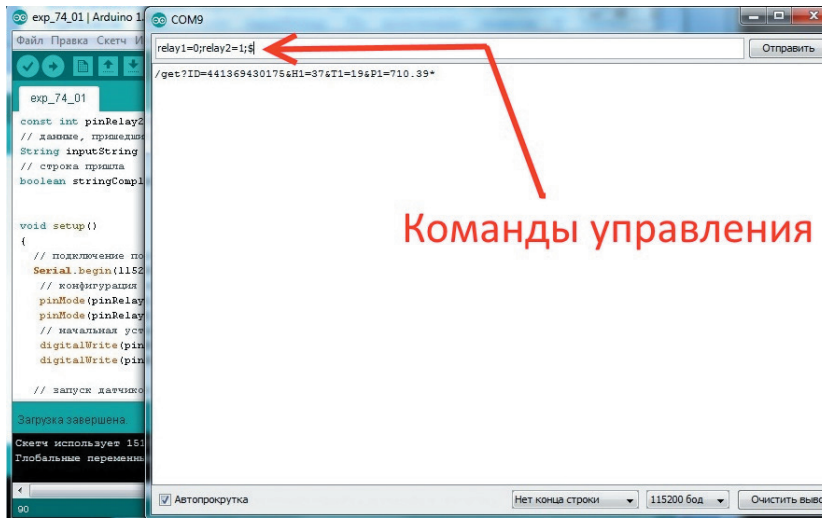


Рис. 74.2. Получение команд управления по последовательному порту

Далее необходимо перевести плату в режим ATmega328 ← →ESP8266. Переключатели необходимо установить следующим образом:

1	2	3	4	5	6	7
ON	ON	OFF	OFF	OFF	OFF	OFF

Теперь Arduino и ESP8266 соединены по последовательному порту.

Пробуем отправлять команды управления с сайта и мобильного приложения Народный мониторинг 2018.

Эксперимент 75.
Протокол MQTT. Отправка данных по протоколу MQTT

В этом эксперименте познакомимся с протоколом MQTT для взаимодействия устройств "Интернета вещей" и рассмотрим отправку данных на сервер MQTT

В эксперименте мы будем использовать следующие компоненты:

- Плата Arduino UNO – 1;
- Кабель USB – 1;
- Плата прототипирования – 1;
- Датчик BMP280 – 1;
- Датчик DHT11 – 1;
- Relay shield – 1;
- Провода MM – 8.

Наша плата Arduino-WiFi, имея возможность подключения к сети Интернет, может являться устройством IoT ("Интернет вещей"). Рассмотрим самый распространенный протокол взаимодействия устройств IoT – MQTT (Message Queue Telemetry Transport).

Он обладает рядом преимуществ по отношению к другим протоколам:

- низкое потребление трафика;
- соединение между клиентом и сервером всегда открыто;
- не нагружает интернет канал;
- отсутствие задержек в передаче данных;
- удобная система подписок на топики;

Все это дает возможность мониторинга и управления в режиме реального времени.

Обмен сообщениями в протоколе MQTT осуществляется между клиентом (client), который может быть издателем или подписчиком (publisher/subscriber) сообщений, и брокером (broker) сообщений (например, Mosquitto MQTT). Издатель и подписчик не передают друг другу сообщения напрямую, не устанавливают прямой контакт, могут не знать о существовании друг друга. Издатель отправляет данные на MQTT брокер, указывая в сообщении определенную тему, топик (topic).

Подписчики могут получать разные данные от множества издателей в зависимости от подписки на соответствующие топики.

Топики представляют собой символы с кодировкой UTF-8. Иерархическая структура топиков имеет формат "дерева", что упрощает их организацию и доступ к данным. Топики состоят из одного или нескольких уровней, которые разделены между собой символом "/".

```
/home/living /living-room1/temperature
```

Устройства MQTT используют определенные типы сообщений для взаимодействия с брокером, ниже представлены основные:

- Connect – установить соединение с брокером;
- Disconnect – разорвать соединение с брокером;
- Publish – опубликовать данные в топик на брокере;
- Subscribe – подписаться на топик на брокере;
- Unsubscribe – отписаться от топика.

Схема взаимодействия между подписчиком, издателем и брокером показана на рис. 75.1.

Рис. 75.1. Схема взаимодействия по протоколу MQTT

Однако MQTT требует наличие своего собственного сервера брокера. Тут есть два выхода: либо создавать свой сервер, либо использовать сторонние сервисы. Например, удобный сервис www.cloudmqtt.com, у которого есть бесплатный тарифный план (Cute Cat).

После регистрации, выбора тарифного плана, местоположения сервера (EU или USA) создается устройство, в котором можно посмотреть Ваши данные: адрес сервера, имя и пароль пользователя, порты подключения и ключ API key (hbc/ 75.2).

Настроим отправку данных брокеру с нашей домашней метеостанции (эксперимент 74). Мы оставим без изменений схему соединений и скетч для Arduino (можно только значительно уменьшить интервал отправки данных с датчиков из Arduino в последовательный порт). Изменяем скетч для модуля ESP8266. При написании скетча используем библиотеку PubSubClient. Необходимо из строки, получаемой из Arduino по последовательному порту выделить показания температуры, влажности и атмосферного давления:

```
int h1=inputString.indexOf("H1=");
int h2=inputString.indexOf("&T1=");
```

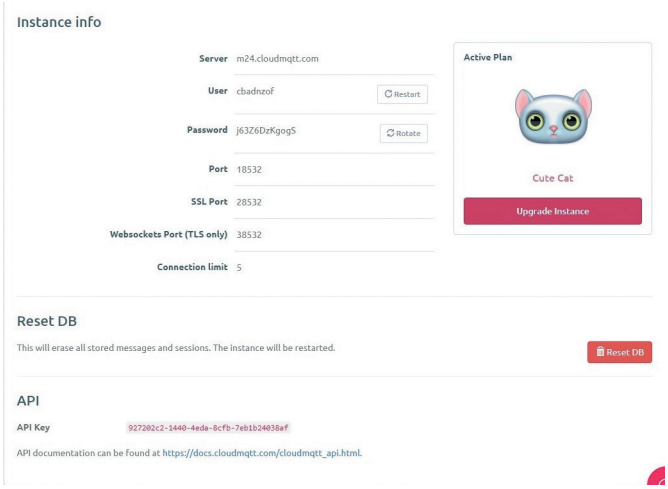


Рис. 75.2. Данные для взаимодействия с брокером MQTT

```
int t1=inputString.indexOf("T1=");
int t2=inputString.indexOf("&P1=");
int p1=inputString.indexOf("P1=");
int p2=inputString.length();
int h=inputString.substring(h1+3,h2).toInt();
int t=inputString.substring(t1+3,t2).toInt();
int p=inputString.substring(p1+3,p2).toInt();
```

И затем данные отправить в темы meteo/humidity, /meteo/temperature, /meteo/pressure/.

Содержимое скетча показано в листинге 75.1.

Листинг 75.1.

```
// подключение библиотек
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
// данные SSID и пароль точки доступа
const char* ssid     = "Kassa1";
const char* password = "12345678";
// создание объекта клиента
WiFiClient wclient;
// данные mqtt
#define mqtt_server  "m24.cloudmqtt.com"
#define mqtt_port 18532
#define mqtt_user  "cbadnzof"
#define mqtt_pass  "**************"
// создание pubsub клиента
PubSubClient client;
// список тем отправки
```

```
#define topich "/meteo/humidity"
#define topict "/meteo/temperature"
#define topicp "/meteo/pressure"
// данные, пришедшие из последовательного порта
String inputString = "";
String inputString1 = "";
// строка пришла
boolean stringComplete = false;

void setup(void){
  delay(5000);
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  Serial.print("*");

  // Подсоединение к точке доступа
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.print("$");

  client.setClient(wclient);
  client.setServer(mqtt_server, mqtt_port);
}

void loop(void){
    if (!client.connected()) {
      reconnect();
    }
    client.loop();
    // проверка данных по последовательному порту
    serialEvent1();
    if (stringComplete) {
       // публикация данных
       publishData(inputString);
       // очистить строку
       inputString = "";
       stringComplete = false;
    }

}
// переподключение к mqtt
```

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...$");
    // Attempt to connect
    if (client.connect("Arduino+WiFi", mqtt_user, mqtt_pass)) {
       Serial.println("Connected to MQTT server$");
    }
    else {
       Serial.println("Could not connect to MQTT server");
       Serial.println(" try again in 5 seconds$");
       delay(5000);
    }
  }
}
// получение строки по Serial
void serialEvent1() {
  boolean flag1=false;
  while (Serial.available() && flag1==false) {
    // получить байт:
    char inChar = (char)Serial.read();
    //Serial.write(inChar);
    if (inChar == '$') {
      stringComplete = true;
      flag1=true;
    }
    else   // добавление в строку
      inputString += inChar;
  }
}

// отправка данных в темы брокера
void publishData(String str) {
    // выведение данных из строки
    // пришедшей из последовательного порта
    int h1=inputString.indexOf("H1=");
    int h2=inputString.indexOf("&T1=");
    int t1=inputString.indexOf("T1=");
    int t2=inputString.indexOf("&P1=");
    int p1=inputString.indexOf("P1=");
    int p2=inputString.length();
    int h=inputString.substring(h1+3,h2).toInt();
    int t=inputString.substring(t1+3,t2).toInt();
    int p=inputString.substring(p1+3,p2).toInt();
    Serial.print("h=");Serial.print(h);
    Serial.print("t=");Serial.print(t);
    Serial.print("p=");Serial.println(p);
    Serial.println("send$");
```

```
    client.publish(topich, String(h).c_str(), true);
    delay(500);
    client.publish(topict, String(t).c_str(), true);
    delay(500);
    client.publish(topicp, String(p).c_str(), true);
}
```

Скачать данный скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_75_01.zip.**

Переключатели на плате Arduino+WiFi необходимо установить следующим образом и загрузить скетч на ESP8266.

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

Открываем монитор последовательного порта, и пробуем отправлять строку вида:

```
/get?ID=440365430747&H1=45&T1=13&P1=713$
```

Идет выделение данных из строки и отправка данных на сервер в темы (рис. 75.3).

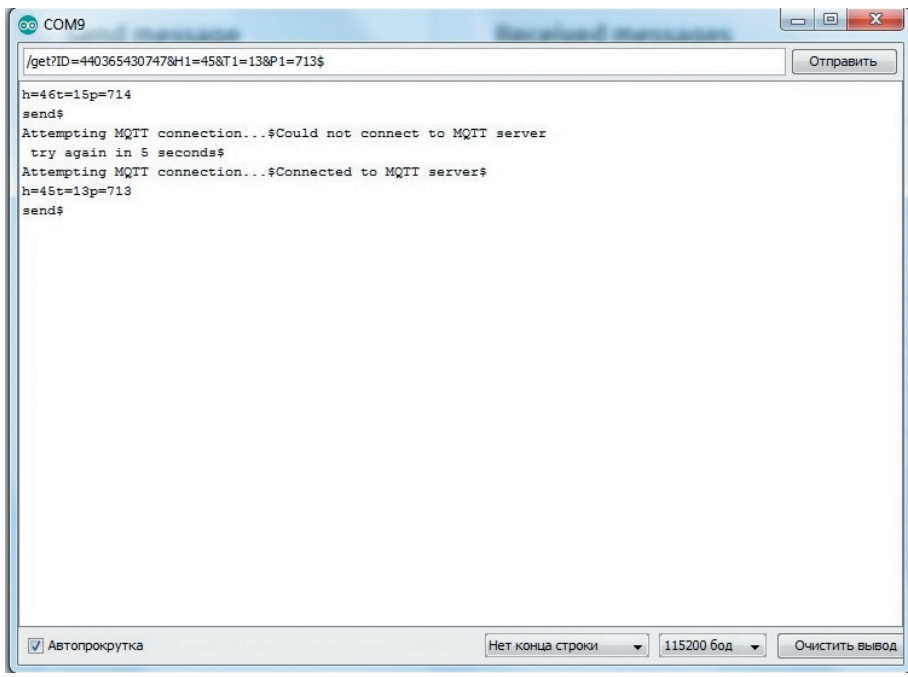


Рис. 75.3. Получение строки данных по последовательному порту

Далее необходимо перевести плату в режим ATmega328 ← →ESP8266. Переключатели необходимо установить следующим образом

1	2	3	4	5	6	7
ON	ON	OFF	OFF	OFF	OFF	OFF

Теперь Arduino и ESP8266 соединены по последовательному порту.

Заходим в профиль созданного устройства на сайте www.cloudmqtt.com и в пункте WEBSOCKET UI смотрим приходящие в темы данные (рис. 75.4).

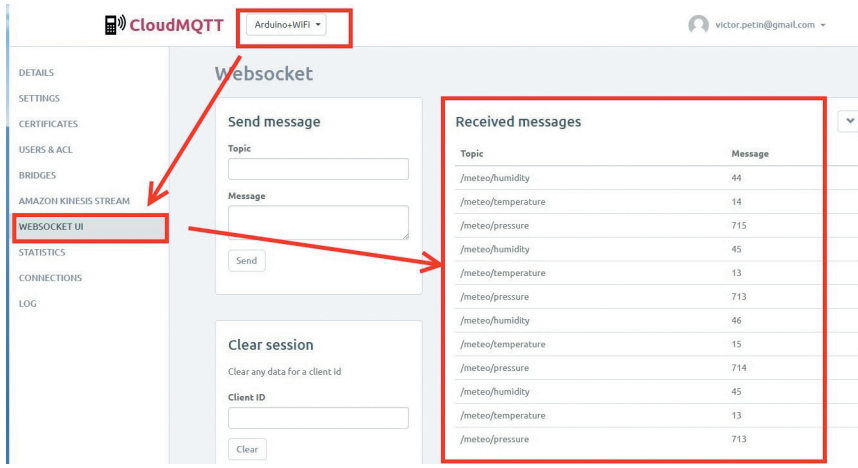


Рис. 75.4. Получение данных на брокере.

Эксперимент 76.
Получение данных по протоколу MQTT

В этом эксперименте рассмотрим получение данных по протоколу MQTT

В эксперименте мы будем использовать следующие компоненты из эксперимента 75. В эксперименте 75 мы рассмотрели отправку данных с нашей домашней метеостанции на брокер MQTT, расположенный по адресу www.cloudmqtt.com. В этом эксперименте рассмотрим получение данных по протоколу MQTT для управления исполнительными устройствами, подключенными к домашней метеостанции через Relay shield.

Отправить данные на метеостанцию можно из профиля на сайтewww.cloudmqtt.com – пункт меню WEBSOCKET UI (рис. 76.1).

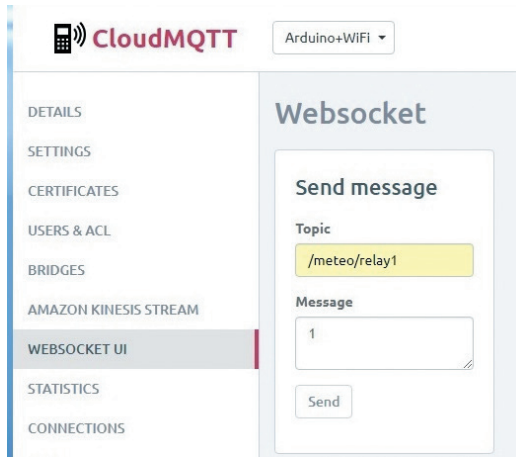


Рис. 76.1. Отправка команд управления из профиля cloudmqtt.com

Отправлять данные будем в темы /meteo/relay1 и /meteo/relay2. Чтобы получать данные команды от сервера (брокера), наше устройство при подключении к брокеру должно подписаться на эти темы:

```
client.subscribe("/meteo/relay1");
client.subscribe("/meteo/relay2");
```

Теперь необходимо назначить функцию обратного вызова callback для обработки сообщений от брокера, которая формирует строку команд управления для отправки по последовательному порту на Arduino:

```
client.setCallback(callback);
void callback(char* topic, byte* payload, unsigned int length) {
   Serial.print(topic);
   Serial.print("=");
   for (int i = 0; i < length; i++) {
      char receivedChar = (char)payload[i];
      if (receivedChar == '0')
         Serial.print("1");
      if (receivedChar == '1')
         Serial.print("0");
   }
   Serial.println("$");
}
```

При получении данных от брокера в последовательный порт отправляется строка вида:

/meteo/relay1=1$

Скачать полностью скетч можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_76_01.zip.**

Переключатели на плате Arduino+WiFi необходимо установить следующим образом и загрузить скетч на ESP8266.

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

Открываем монитор последовательного порта, и при отправке команд на сайте www.cloudmqtt.com в темы /meteo/relay1 и /meteo/relay2 видим получение сообщений от брокера и отправку строк для управления реле в последовательный порт (рис. 76.2).

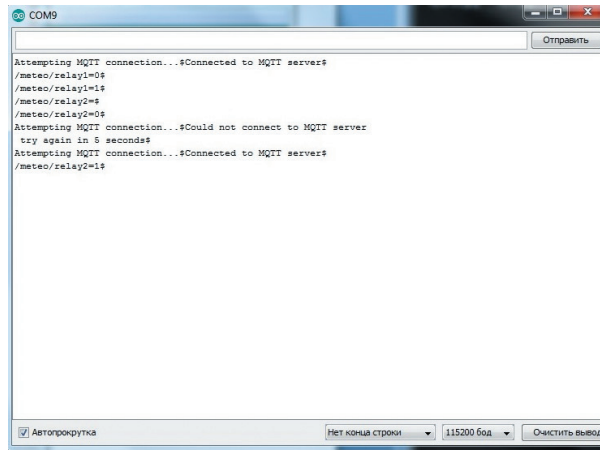


Рис. 76.2. Получение команд от брокера и отправка команд управления в последовательный порт

Более удобный вариант – использование Android-приложения IoT MQTT Dashboard, который можно использовать в качестве пульта, а так же для отображения информации с датчиков домашней метеостанции. Приложение представляет собой готовый mqtt клиент с небольшим количеством очень удобных виджетов.

Скачиваем приложение на телефон. Сначала создаем новое соединение и прописываем данные из нашего профиля cloudmqtt (рис. 76.3).

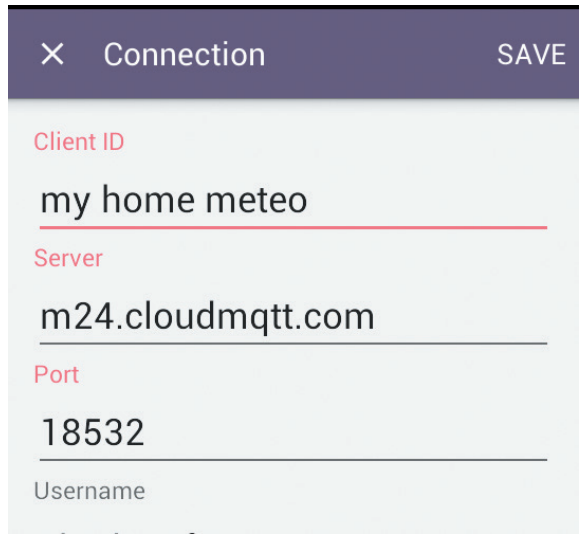
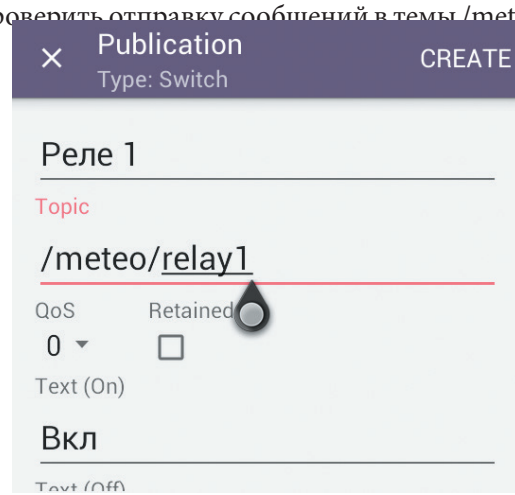


Рис. 76.3. Создание соединения

Выбираем созданное соединение и во вкладке PUBLISH создаем виджеты (switch) для отправки данных в темы /meteo/relay1 и /meteo/relay2 (рис. 76.4, 76.5, 76.6). Теперь можно проверить отправку сообщений в темы /meteo/relay1 и /meteo/relay2.



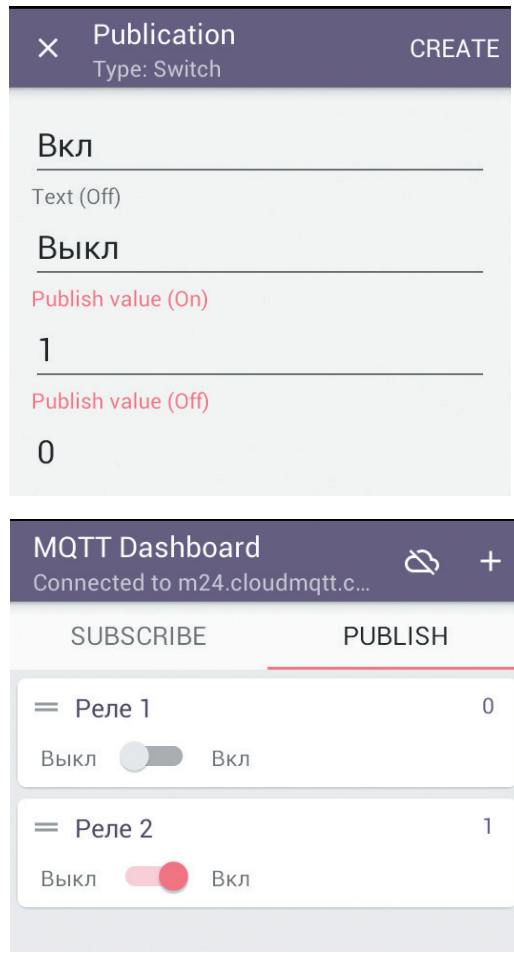


Рис. 76.4, 76.5, 76.6. Создание виджетов для отправки сообщений в темы

Приложение позволяет также настроить отображение данных метеостанции, котрые отправляются брокеру. Во вкладке SUBSCRIBE создаем виджеты для отображения температуры, влажности и давления, где небходимо подписаться на топики /meteo/humidity, /meteo/temperature, /meteo/pressure (рис. 76.7).

И получим на экране телефона табло с отображением данных нашей домашней метеостанции (рис. 76.8)

Далее необходимо перевести плату в режим ATmega328 ← →ESP8266. Переключатели необходимо установить следующим образом

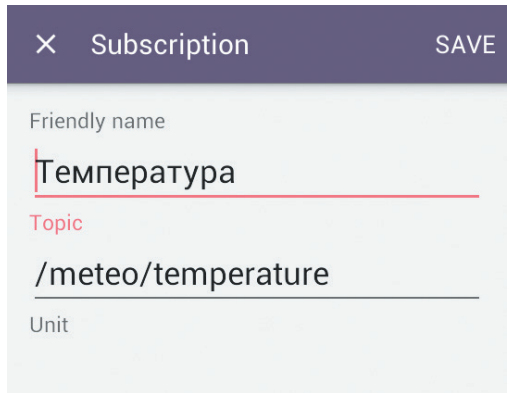


Рис. 76.7. Создание виджетов для получения данных из тем

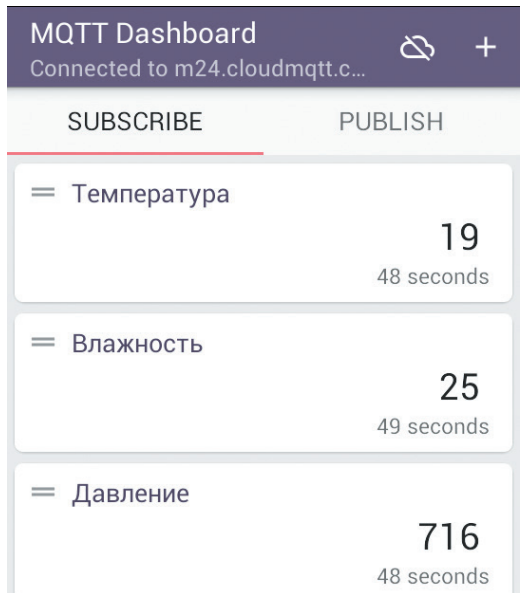


Рис. 76.8. Отображение данных метеостанции

1	2	3	4	5	6	7
ON	ON	OFF	OFF	OFF	OFF	OFF

Теперь Arduino и ESP8266 соединены по последовательному порту. Проверяем поступление данных и команды управления.

Эксперимент 77.
Отправляем с web-сервера в интернет-магазин Arduino-kit отзывы и пожелания о книге и наборе

В этом эксперименте создадим на web-сервере страницу, для отправки отзывов по этой книге и набору "Лаборатория электроники и программирования"

В эксперименте мы будем использовать следующие компоненты:
- Плата Arduino UNO – 1;
- Кабель USB – 1.

Эксперимент заключительный и пришло время оставить отзывы по книге и набору, с которым мы вместе провели 77 экспериментов, с которым Вы освоили на практике основы программирования, конструирования электронных устройств и робототехники на основе контроллеров – плат Arduino и WiFi модулей ESP8266. Но Вы сможете не просто оставить отзывы и предложения, но создадим очередной проект, где мы реализуем на нашей плате Arduino WiFi сервер, формирующий web-страницу, и отправляющий Ваши ответы на сайт производителя набора "Лаборатория электроники и программирования" – https://arduino-kit.ru.

Нам необходимо создать на модуле ESP8266 web-сервер, выдающий страницу, где можно оставить отзыв о наборе, ответив на несколько вопросов. HTML- код страницы показан в листинге 77.1.

Листинг 77.1.

```
<!DOCTYPE HTML>
<html>
  <head>
     <meta http-equiv=pragma content=no-cache>
     <meta http-equiv=Expires content='-1'>
     <meta http-equiv=Cache-Control content=no-cache>
     <script>
     function viewquestion46() {
       var inp = document.getElementsByName('question4');
       if (inp[4].type == "radio" && inp[4].checked)
         {document.getElementById('question46').style.
                     visibility='visible';}
```

```
      else
        {document.getElementById('question46').
              style.visibility='hidden';
        document.getElementById('question46').value='';}
 }
</script>
</head>
 <body>
    <form name="survey" method='post'
    action="http://victorpetin.ru/arduino-kit/survey_to_mail.php" >
     Оцените книгу "77 ПРОЕКТОВ НА ARDUINO" по 5-бальной шкале <br>
     <select name="question1">
        <option disabled>Выберите оценку</option>
        <option selected value="5">5</option>
        <option value="4">4</option>
        <option value="3">3</option>
        <option value="2">2</option>
        <option value="1">1</option>
     </select><br>
     Оцените набор-конструктор "ЛАБОРАТОРИЯ ЭЛЕКТРОНИКИ И
     ПРОГРАММИРОВАНИЯ НА ОСНОВЕ ARDUINO" по 5-бальной шкале <br>
     <select name="question2">
        <option disabled>Выберите оценку</option>
        <option selected value="5">5</option>
        <option value="4">4</option>
        <option value="3">3</option>
        <option value="2">2</option>
        <option value="1">1</option>
     </select><br>
     Вы посоветуете данный набор-конструктор своим друзьям
     и знакомым?<br>
        <label>Да</label>
        <input type="radio" name="question3" value="1" checked>
        <label>Нет</label>
        <input type="radio" name="question3" value="2">
     <br>
     Какую платформу Вы считаете наиболее перспективной для
     учебного процесса и DIY: <br>
        <label>Arduino</label>
        <input type="radio" name="question4" value="1"
              checked onchange="viewquestion46();">
        <label>Raspberry Pi</label>
        <input type="radio" name="question4" value="2"
            onchange="viewquestion46();">
        <label>BBC: microbit</label>
        <input type="radio" name="question4" value="3"
            onchange="viewquestion46();">
        <label>ESP32</label>
```

```
            <input type="radio" name="question4" value="4"
                onchange="viewquestion46();">
            <label>другую</label>
            <input type="radio" name="question4" value="5"
                onchange="viewquestion46();">
            <br>
            <input type="text" name="question46" id="question46"
             style="visibility:hidden">
          <br>
          Какие пожелания по элементной базе в составе набора? <br>
          <textarea rows="10" cols="45" name="question5"></textarea>
          <br>
          Какие эксперименты добавить? <br>
          <textarea rows="10" cols="45" name="question6"></textarea>
          <br>
          <input type=submit value="Отправить">
        </form>
    </body>
</html>
```

Создаем скетч создания web-сервера, выдающий данную страницу при обращении к серверу из браузера.

Скачать скетч можно на сайте Arduino-kit по ссылке

https://arduino-kit.ru/scetches/exp_77_02.zip.

Переключатели на плате Arduino+WiFi необходимо установить следующим образом и загрузить скетч на модуль ESP8266.

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

Открываем монитор последовательного порта, где можно посмотреть ip-адрес, присвоенный модулю ESP8266 в сети (рис. 77.1). Обращаемся из браузера по данному адресу и получаем страницу для отправки отзыва (рис. 77.2).

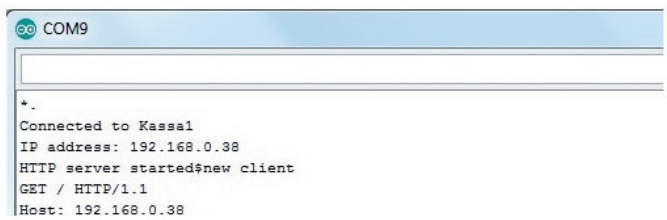


Рис. 77.1. Получение модуля ESP8266 к WiFi-сети

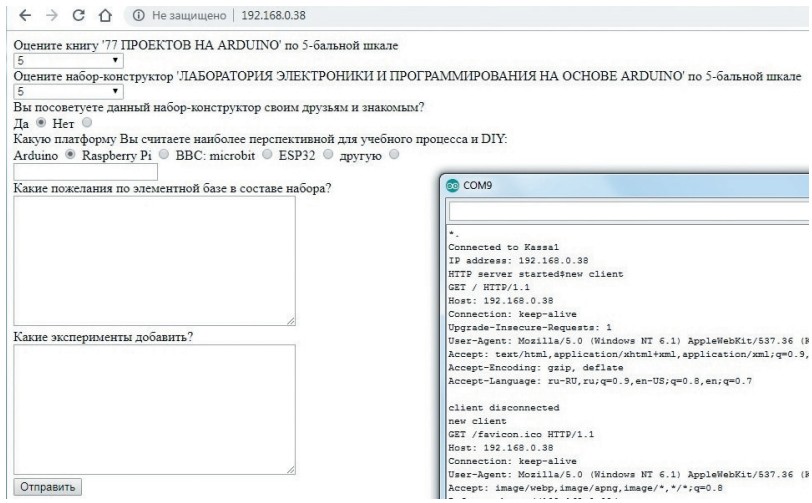


Рис. 77.2. Страница для отправки отзыва

Просьба оставить отзыв, который поможет нам создавать интересные электронные наборы. Мы учтем все Ваши пожелания и предложения.

Спасибо!

Книга издательства «ДМК Пресс» можно заказать
в торгово-издательском холдинге «Планета Альянс» наложенным платежом,
выслав открытку или письмо по почтовому адресу:
115487, г. Москва, 2-й Нагатинский пр-д, д. 6А
При оформлении заказа следует указать адрес (полностью),
по которому должны быть высланы книги;
фамилию, имя и отчество получателя.
Желательно также указать свой телефон и электронный адрес.
Эти книги вы можете заказать и в интернет-магазине: www.alians-kniga.ru.
Оптовые закупки: тел.: (499) 782-38-89.
Электронный адрес: books@alians-kniga.ru

Образовательные электронные конструкторы серии «СМАЙЛ» можно
заказать в компании ЭМБИТЕХ
Сайт: www.mbitech.ru
Оптовые закупки: тел.: (499) 502-84-00
Электронный адрес: info@mbitech.ru

Виктор Александрович Петин

77 проектов для Arduino

Главный редактор Мовчан Д.А.
dmkpress@gmail.com
Корректор Синяева Г.И.
Верстка и
дизайн обложки Махмутова Э.Ш.

Формат 70x100 1/16
Гарнитура «Pt Serif». Печать офсетная.
Усл. печ.л. 16,41. Тираж 200 экз.

Веб-сайт издательства: **www.dmkpress.com**