Лабораторная работа № 1

Управление яркостью трехцветного светодиода с клавиатуры

Теоретическая часть:

Уровни согласования

Прежде чем мы предоставим устройствам возможность общаться друг с другом, нам нужно определиться с основными вопросами такого общения. Эти вопросы можно разбить на пять уровней, каждый из которых основывается на предыдущем.

Физический. Каким образом вводы и выводы каждого устройства соединяются с другими? Сколько необходимо соединений между двумя устройствами для возможности обмена сообщениями?

Электрический. Какие уровни напряжения использовать для представления битов данных? 5 В? 3,3 В? Какое-либо другое напряжение?

Логический. Какую схему логики использовать — положительную или отрицательную? Схема, при которой высокий уровень напряжения представляет логическую 1, а низкий — логический 0, называется *положительной*, а когда значения уровней инвертированы: высокий уровень напряжения представляет логический 0, а низкий — логическую 1, — *отрицательной*.

Уровень данных. Как осуществляет синхронизация битов? Сколько битов считывается за раз: 8, 9, 10 или больше? Обозначаются ли группы битов в начале и в конце специальными битами?

Уровень приложений. Как организовываются группы битов для создания сообщений? Каков порядок обмена сообщениями для того, чтобы что-то сделать?

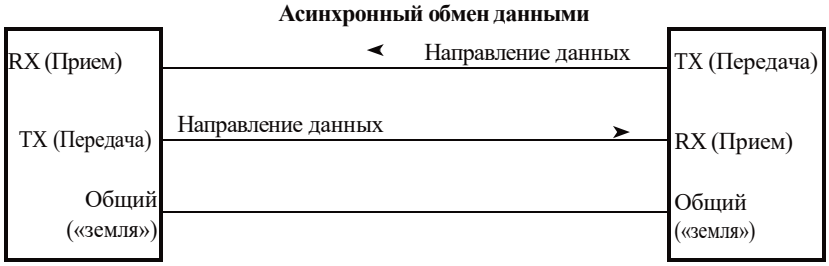
Это упрощенная версия модели организации сетевого обмена, называющейся моделью OSI. В действительности сетевые задачи никогда нельзя так четко разделить, но четкое представление различий между концептуальными уровнями значительно облегчит диагностирование проблем со связью. Подобное мышление в терминах уровней дает нам исходную точку для поиска причины проблемы и способ исключить из подозреваемых те части системы, которые не могут быть этой причиной.

Какой бы сложной ни была бы сеть, никогда не забывайте о том, что связь между электронными устройствами — это всего лишь электрические импульсы. Устройства, осуществляющие обмен данными по последовательному каналу, соединены между собой электрически. Посылающее устройство изменяет уровень напряжения на соединяющей линии через согласованные интервалы времени, и каждый такой интервал представляет собой один бит данных. Чтобы отправить значение 0 или 1 очередного бита, отправитель меняет уровень напряжения, а получатель определяет уровень полученного сигнала: высокий или низкий.

Отправитель и получатель могут согласовывать скорость отправки битов двумя способами (рис. 2.3). Скорость асинхронного последовательного обмена данными согласовывается между обеими сторонами обмена и синхронизируется (иногда говорят: тактируется) отправителем и получателем независимо друг от друга. А скорость синхронного последовательного обмена данными управляется отправителем, который подает постоянный сигнал синхронизации (тактирования) на отдельную линию (на рис. 2.3 показаны два интерфейса синхронного последовательного обмена: интерфейс SPI и интерфейс I2C). Синхронный последовательный обмен данными в основном применяется для связи между интегрированными схемами

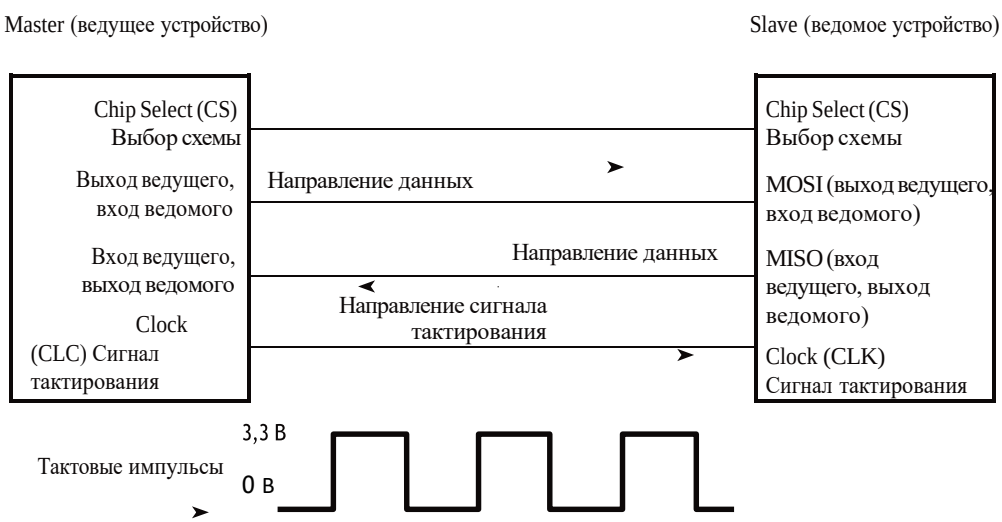
(например, между процессором компьютера и его микросхемами памяти). В этой главе рассматривается только асинхронный последовательный обмен данными, так как именно этот тип последовательной связи лежит в основе сетей: от Ethernet-соединений по проводам до беспроводной радиосвязи.

Асинхронный обмен данными

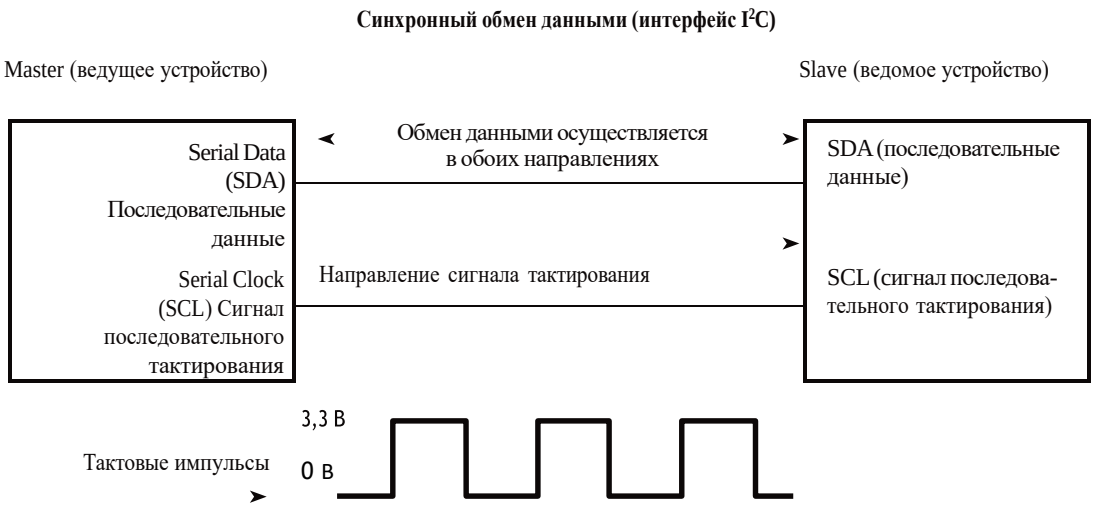


Асинхронный обмен данными: каждое устройство использует свой собственный генератор тактовых (синхронизирующих) сигналов, обмен данными осуществляется с заранее согласованной скоростью.

Синхронный обмен данными (интерфейс SPI)



Синхронный обмен данными (интерфейс SPI, Serial Peripheral Interface): ведущее устройство подает сигнал тактирования на ведомое устройство и инициирует обмен, подавая сигнал выбора схемы. Обмен данными происходит по смене уровня напряжения сигнала тактирования на обратное.



Синхронный обмен данными (интерфейс I²C): ведущее устройство подает сигнал тактирования на ведомое устройство, выбирая ведомое по его адресу. Обмен данными происходит по смене уровня напряжения сигнала тактирования на обратное.

Рис. 2.3. Типы последовательной связи

Устанавливаем соединение: нижние уровни

Сейчас самым активно использующимся примером является средство асинхронного последовательного обмена данными с использованием двух протоколов последовательной связи: TTL Serial и USB.

Первый из них — это протокол, понимаемый микроконтроллером, и называется он *последовательный TT8* (TTL² Serial). Этот протокол можно разделить на следующие концептуальные уровни:

Физический. Определяет контакты, используемые контроллером для обмена данными. В модуле Arduino данные принимаются на контакт, обозначенный RX (от Receive, прием), а передаются с контакта, обозначенного TX (от Transmit, передача).

- **Электрический.** Определяет напряжения для представления битов данных. В некоторых микроконтроллерах применяется напряжение 3,3 В, в других — 5 В.
- **Логический.** Высокий уровень напряжения (3,3 или 5 В) представляет логическое значение 0
- **Уровень данных.** Обмен данными обычно осуществляется со скоростью 9600 битов в секунду. Для представления одного символа требуется один байт, который содержит 8 битов данных, а также стартовый и стоповый биты (которые мы никогда не будем использовать).

  Уровень приложений. На этом уровне мы отправляем один байт от ПК на микроконтроллер, который обрабатывает его и возвращает один байт на ПК.

Но это еще не все. Импульсы напряжения не идут на ПК напрямую. Сначала они поступают на TTL/USB-микросхему на плате, которая преобразовывает последовательные TTL-сигналы в последовательные USB-сигналы. На некоторых микроконтроллерных платах — например, на Arduino Uno, преобразователь USB сигналов в сигналы ТТЛ реализован в виде отдельной микросхемы. На других же — например, на MKR1000 и Arduino 101, эта функциональность встроена в микросхему микроконтроллера.

Обработка преобразованных в формат USB сигналов осуществляется с использованием соответствующего протокола USB3, который во многом отличается от протокола последовательного обмена TTL. Он разбивается на следующие концептуальные уровни:

Физический. Шина USB состоит из двух проводов для передачи данных: Data+ и Data– и двух проводов питания (+5 В и общего).

Электрический. Сигнал на линии Data– всегда противоположен сигналу на линии Data+, в результате чего сумма напряжений этих сигналов всегда равна нулю. Эта особенность используется приемником для проверки на наличие ошибок электрического сигнала — для этого он складывает напряжения обеих этих линий. Если полученная сумма не равна нулю, приемник отбрасывает такой сигнал

Логический. Логическое значение 1 представляется сигналом напряжения +5 В (на линии Data+) или –5 В (на линии Data–), а логическое значение 0 — сигналом напряжения 0 В

Уровень данных. Уровень данных модели дели протокола USB более сложный, чем соответствующий уровень протокола последовательного обмена ТТЛ. В USB скорость обмена данными может достигать 480 мегабит в секунду. Один символ представляется одним байтом, который содержит 8 битов данных, а также стартовый и стоповый биты. Несколько устройств USB под управлением ПК могут использовать для обмена данными одну и ту же пару проводов (набор проводов для обмена сигналами называется *шиной*). Поскольку к одной шине может быть подсоединено несколько устройств, операционная система присваивает каждому из этих устройств однозначный адрес и обеспечивает обмен данными между каждым подсоединенным к шине устройством и его соответствующим приложением на компьютере.

Уровень приложений. На уровне приложения преобразователь USB/TTL-Serial на плате Arduino отправляет операционной системе компьютера несколько байтов, чтобы идентифицировать себя. Операционная система использует эти байты для того, чтобы сопоставить плату с программой ее драйвера, который другие программы может использовать для обмена данными с этим устройством.

Все это управление прозрачно для пользователя, так как контроллер USB компьютера передает ему только те байты, которые предназначены для него. Микросхема преобразователя USB/ TTL-Serial на плате Arduino представляет себя операционной системе компьютера в качестве последовательного порта и отправляет данные через разъем USB с выбранной пользователем скоростью (9600 битов в секунду — как в примере).

Рис. 2.4. Кабель-переходник USB/TTL-Serial комп

Одна из примечательных особенностей микроконтроллеров — их низкая стоимость, что позволяет не особенно ограничивать себя в их числе. Например, для проекта с большим числом датчиков можно или разработать сложную программу, чтобы один микроконтроллер мог выполнять опрос всех датчиков, или же дать каждому датчику свой собственный микроконтроллер. А если информацию с этих датчиков нужно передать в персональный компьютер, поначалу может показаться, что будет легче использовать для этого один микроконтроллер — по причине ограниченного количества последовательных портов. Однако благодаря шине USB количеством доступных последовательных портов можно не озабочиваться. Если микроконтроллер оснащен портом USB, его можно подсоединить к любому разъему USB компьютера, и он будет распознаваться операционной системой компьютера как еще один последовательный порт. А количество разъемов USB компьютера можно увеличить, подключив к одному из них USB-концентратор (хаб).

Например, если к компьютеру подсоединить три модуля Arduino через USB-хаб, в операционной системе появится три новых последовательных порта. На машинах под Mac OS они будут отображаться примерно так:

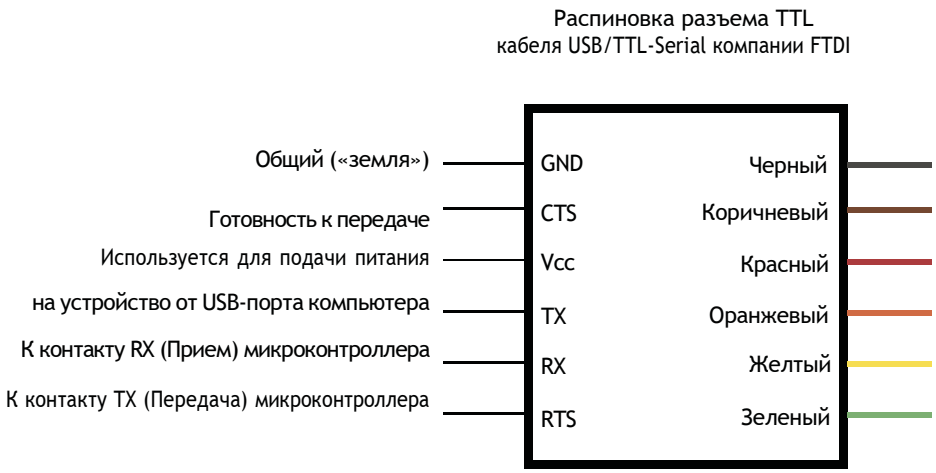
```
/dev/cu.usbmodem1441

/dev/cu.usbmodem1461
```

В системах POSIX, включая macOS, порты часто указываются дважды: один раз как `/dev/tty.usb-modemXX` и еще раз как `/dev/cu.usbmodemXX`. Эта особенность является пережитком эпохи телефонных модемов: порты TTY[4] служили для коммутации входящих звонков, а CU[5] — для исходящих. Для устройств USB/TTL-Serial, которые применяются в этой книге, тип используемого порта не имеет существенной важности.

На машинах под Windows эти порты будут отображаться, например, так: COM8, COM9, COM10.

Большинство современных микроконтроллерных плат оснащены встроенным модулем преобразователя USB/TTL-Serial. Для тех плат, которые не имеют встроенного преобразователя, на рынке предлагаются отдельные модули ценой порядка $15–20. Один из наиболее популярных таких преобразователей с разъемом для макетных плат, который удобно использовать для подсоединения устройств, оснащенных последовательным интерфейсом технологии TTL, выпускает компания FTDI (Future Technology Devices International), сайт которой находится по адресу **www.ftdichip.com**. Этот преобразователь можно приобрести в разделе макетных плат и кабелей магазинов фирм Maker SHED, SparkFun, Adafruit и многих других. Доступен он в двух версиях: 5 В и 3,3 В — любую из них можно использовать для всех проектов этой книги. Кабель-переходник USB/ TTL-Serial компании FTDI показан на рис. 2.4, а распиновка его разъема TTL — на рис. 2.5

Рис. 2.5. Распиновка разъема TTL кабеля USB/TTL-Serial компании FTDI. Кроме линий передачи, приема и питания он также имеет линии для аппаратного управления обменом данных: RTS (Request-to-send, запрос на передачу) и CTS (Clear-to-send, готовность к передаче). Некоторые устройства используют эти линии для управления потоком последовательных данных



Есть и еще один протокол — если вы используете микроконтроллерный модуль BASIC Stamp или другой микроконтроллерный модуль с интерфейсом иным, чем USB, он, скорее всего, оснащен 9-контактным разъемом для подключения к компьютеру или к адаптеру USB/RS-232 (распиновка разъемов USB и RS-232 показана на рис. 2.6). Этот разъем называется DB-9 или D-sub-9 и является стандартным разъемом для другого последовательного протокола: RS-232. Протокол RS-232 был стандартом для последовательного интерфейса компьютеров до протокола USB и все еще встречается на некоторых специализированных периферийных устройствах. Этот протокол состоит из следующих концептуальных уровней:

Физический. Данные в разъеме RS-232 компьютера принимаются на контакт 2, а передаются с контакта 3. Контакт 5 — «земля».
Электрический. Данные в стандарте RS - 232 передаются двумя уровнями напряжения: от +3 В до +25 В и от –3 В до –25 В.
Логический. Высокий уровень напряжения (от +3 В до +25 В) представляет логическое значение 0, а низкий (от –3 В до –25 В) — логическое значение 1. Как можно видеть, эта схема построена на основе *отрицательной* (или инвертированной) логики.

Уровень данных. Такой же, как и в протоколе TTL, — 8-битовыйбайт данных с дополнительными стартовым и стоповым бита

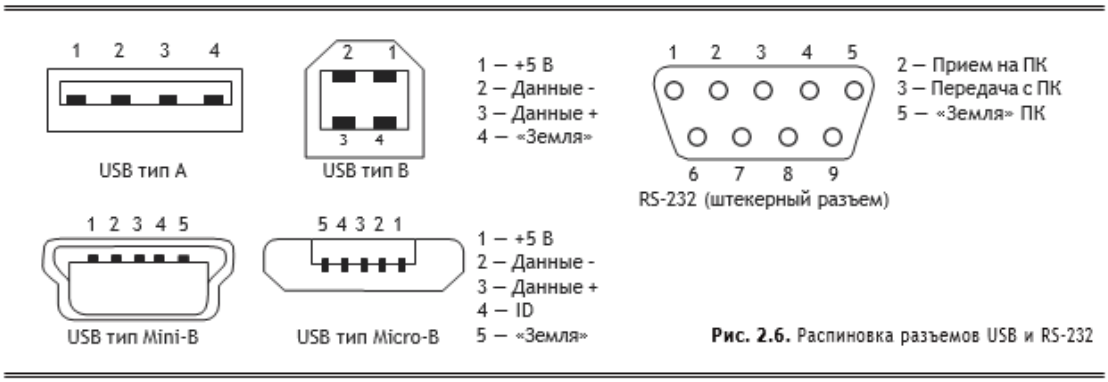


Рис. 2.6. Распиновка разъемов USB и RS-232

Спрашивается, как же подключать некоторые микроконтроллеры — например, BASIC Stamp, непосредственно к последовательному порту RS-232? Поскольку микроконтроллеры обычно не могут создавать отрицательные напряжения, для преобразования выходных сигналов ТТЛ микроконтроллера в уровни RS-232 обычно используется отдельная специальная микросхема. Протокол RS-232 намного проще, чем протокол USB, но, к сожалению, он, по большому счету, сильно устарел. Большинство современных микроконтроллеров оснащены встроенным преобразователем USB/TTL-Serial.

В большинстве случаев вам никогда не придется думать о таком смешении протоколов, и вы сможете просто использовать для выполнения этой задачи соответствующие преобразователи. Тем не менее, будет полезно немного знать о внутреннем механизме этих протоколов на случай, если придется искать причину какой-либо проблемы со связью.

Преобразователи USB/Serial

Большинство современных электронных модулей, с которыми вам придется иметь дело, будут, скорее всего, оснащены каким-либо последовательным интерфейсом для взаимодействия с микроконтроллерами. Самым распространенным из таких интерфейсов является последовательный TTL-интерфейс — TTL-Serial. Например, большинство модулей системы GPS.

оснащены интерфейсом TTL-Serial. Для любого, кто интересуется созданием современных электронных устройств, преобразователь USB/ TTL-Serial станет одним из самых необходимых инструментов.
На рынке доступен ряд микросхем преобразователя USB/Serial. Одной из наиболее популярных является микросхема FT232RL, изготавливаемая уже упомянутой ранее компанией FTDI. На основе этой микросхемы компания делает кабель-переходник, другие компании также используют эту микросхему для изготовления плат-преобразователей. Кабель- переходник компании FTDI (см. рис. 2.4) приобрел такую популярность, что его распиновка (см. рис. 2.5) была принята в качестве де-факто стандарта на рынке любительской электроники и используется во многих устройствах.
Кроме кабеля-преобразователя компании FTDI, преобразовательные платы изготавливают такие компании, как Adafruit, SparkFun, Parallax и ряд других. Микросхемой FT232RL также оснащаются многие устройства, включая адаптеры XBee компании Digi и платы-клоны Arduino RedBoard компании SparkFun.
Эта микросхема удобна тем, что она может обрабатывать последовательные TTL-сигналы разных напряжений: 5 и 3,3 В. Компания SparkFun изготавливает отдельные версии плат для каждого напряжения, а у платы FTDI Friend компании Adafruit на тыльной стороне предусмотрен ряд перемычек, перепаивая которые можно менять рабочее напряжение контактов для приема и передачи последовательных сигналов. Кроме того, эта плата также способна работать с уровнями напряжения сигналов RS-232. Для работы с RS-232 предназначена и адаптерная плата компании Parallax, оснащенная к тому же разъемом DB-9.

Использование адаптера USB/TTL-Serial не представляет собой ничего сложного. Контакт передачи (TX) этого адаптера подсоединяется к контакту приема (RX) устройства и наоборот. Общий («земляной») контакт платы подключается к соответствующему контакту устройства, а при питании устройства от адаптерной платы контакт питания V_{CC} платы подключается к контакту питания устройства. Прежде чем запитывать устройство от адаптера, необходимо убедиться в том, что устройство может работать на уровне напряжения, подаваемого адаптером. Напряжение питания большинства адаптеров USB/TTL-Serial составляет 5 В, что может повредить устройства с питанием 3,3 В. В *главе 1* мы узнали, как с помощью адаптера USB/TTL- Serial компании FTDI подключиться через монитор порта к плате Raspberry Pi. Но в том случае плата микроконтроллера не запитывалась от адаптерной платы.

Чтобы использовать с компьютером любой из адаптеров USB/Serial, на компьютер нужно установить драйверы микросхемы этого адаптера. Качество работы адаптера USB/TTL-Serial во многом зависит от качества его драйвера. Драйверы более дешевых адаптеров обычно совместимы с более узким диапазоном платформ. Поэтому имеет смысл немного переплатить, но приобрести устройство с доступными, качественными и много платформенными драйверами. В этом отношении особенно выделяется компания FTDI — драйверы от этой компании всегда вовремя обновляются для поддержки текущих версий Windows, macOS и различных дистрибутивов Linux.
Но кроме компании FTDI, адаптеры USB/TTL- Serial изготавливаются и другими компаниями — например, компаниями Prolific (адаптер PL2303), Silicon Labs (адаптер CP2102), Jiangsu Heng Qin (адаптер CP340) и другими. Драйверы для упомянутых адаптерных плат этих компаний можно загрузить со следующих вебсайтов:

- **www.ftdichip.com/FTDrivers.htm** (FTDI);
- **www.silabs.com/products/mcu/Pages/ USBtoUARTBridgeVCPDrivers.aspx** (Silicon Labs);
- **www.prolific.com.tw/US/Show-Product. aspx?pcid=41** (Prolific);
- **www.wch.cn/download/CH341SER_ZIP. html** (Jiangsu Heng Qin).

В плате Arduino Uno в качестве адаптера USB/ TTL-Serial используется микроконтроллер общего

назначения Atmel 16U2, запрограммированный под эту задачу. Для этого адаптера не требуются драйверы под macOS и Linux, а драйверы USB для Windows устанавливаются автоматически установщиком Windows. Исходный код адаптера USB/TTL-Serial для Arduino Uno можно загрузить из каталога **hardware/ardu- ino/avr/firmwares/ATmegaxxu2** репозитория **https://github.com/arduino**, а дополнительную информацию о конструкциях устройств USB вы найдете на веб-сайте **www.usb.org/develop- ers/usbfaq**.

При использовании адаптера USB/TTL-Serial для программирования микроконтроллера вам, вероятно, нужно будет также задействовать выводы CTS (clear-to-send, готовность к передаче) и RTS (request-to-send, запрос на передачу). Большинство микроконтроллеров, полагающихся на адаптеры USB/TTL-Serial, — такие, как адаптерная плата Huzzah! на микросхеме ESP8266 компании Adafruit или плата ESP8266 Thing компании SparkFun, оснащены контактами для этих выводов, так что вам не нужно беспокоиться об этом. Дополнительную информацию по использованию адаптеров USB/TTL- Serial для программирования микроконтроллеров ищите в руководстве по программированию вашей платы.

Использование платы **Arduino** в качестве адаптера **USB/TTL-Serial**

Если у вас нет под рукой отдельного адаптера USB/ TTL-Serial, вместо него можно воспользоваться совместимой с Arduino платой, запрограммировав ее должным образом.

В частности, для этой цели подойдут платы MKR1000 или Arduino 101. Обе платы используют одинаковые процессоры, оснащенные встроенными средствами работы с USB, и последовательные порты этих плат, отображаемые в мониторе порта, подключены непосредственно к процессору. Для обращения к выводам TX (передача) и RX (прием) платы обычно используется объект `Serial1`. Далее приводится скетч для реализации этой задачи. В нем вывод передачи USB перенаправляется на вывод приема (RX) платы, а вывод приема USB — на вывод передачи (TX) платы. Таким образом, любая микроконтроллерная плата на микроконтроллере со встроенными средствами работы с USB превращается в адаптер USB/TTL-Serial.

```
void setup() {
   // инициализируем оба последовательные
   // подключения: Serial.begin(9600); // USB Serial1.begin(9600); // TTL
}
void loop() {
      // считываем RX TTL, отправляем на USB: if (Serial1.available()) {
      char c = Serial1.read(); Serial.write(c);
   }
// считываем USB, отправляем на TX TTL: if
(Serial.available()) {
      char c = Serial.read();
      Serial1.write(c);
   }
}
```

Микроконтроллеры более старых плат — например, Arduino Uno, не имеют встроенных средств для работы с USB. Поэтому, чтобы позволить таким платам взаимодействовать с компьютером через порт USB, они оснащаются отдельной микросхемой адаптера USB/TTL-Serial. Выход передачи (TX) такого адаптера подсоединен к контакту приема (RX) микроконтроллера и наоборот. Это означает, что здесь можно обойти микроконтроллер и напрямую использовать плату Arduino в качестве адаптера USB/TTL-Serial. Для этого в микроконтроллер нужно загрузить скетч, который ничего не делает:

```
void setup() {}

void loop() {}
```

Затем подсоединить требуемое последовательное устройство следующим образом (обратите внимание: на первый взгляд может казаться, что подключение должно выполняться наоборот):

- контакт приема внешнего устройства — к контакту RX (0) платы Arduino;
- контакт передачи внешнего устройства — к контакту TX (1) платы Arduino.

Теперь наше устройство может обмениваться данными напрямую с адаптером USB/TTL-Serial платы Arduino, обходя микроконтроллер. Когда же снова нужно будет использовать микроконтроллер, просто отсоедините устройство и загрузите требуемый скетч.

Практическая часть

Проект 1.1

Управление яркостью трехцветного светодиода с клавиатуры

В этом примере мы будем управлять микроконтроллером нажатием клавиш клавиатуры компьютера. Это очень простой проект — в нем используется минимальное количество деталей, что даст нам возможность сосредоточиться непосредственно на обмене данными.

Для любого приложения, каким бы простым оно ни было, требуется коммуникационный протокол. Даже такая несложная задача, как просьба включить свет, требует, чтобы как отправитель, так и получатель команды знали, как сказать «включить свет».

Этот проект будет работать на любой совместимой с Arduino платой — при условии, что она оснащена тремя контактами цифрового вывода, с которыми можно использовать команду `analogWrite()`. Такие контакты называются *выводами ШИМ*, поскольку длительность подаваемых на них сигнальных импульсов можно модулировать (включать сигнал и быстро выключать), создавая, таким образом, на них некое подобие изменяющегося напряжения.

Требуемые компоненты

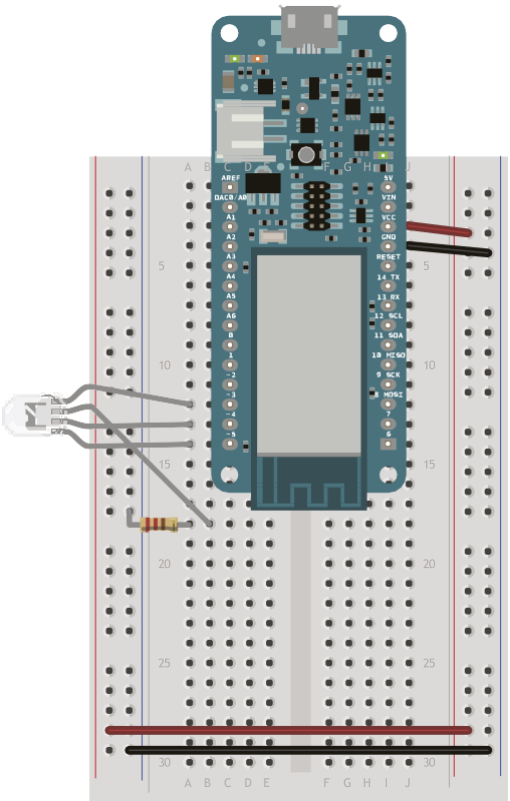
- Трехцветный (RGB) светодиод с общим катодом, 1 шт.
- Резистор на 220 Ом, 1 шт.
- Совместимая с Arduino плата (на рис. 2.8, *вверху*, показана плата MKR1000, а *внизу* — плата Arduino 101), 1 шт.

  Используемые возможности платы: УАПП (UART), вывод ШИМ (PWM).
- Беспаечная макетная плата, 1 шт.
- Персональный компьютер.

Для этого проекта нам потребуется маленькая лампочка, способная светиться разными цветами. Яркостью и цветом свечения этой лампочки мы и будем управлять командами с компьютера. Такой лампочкой нам послужит трехцветный светодиод, который в действительности представляет собой три светодиода: красный, зеленый и синий, упакованные в общем корпусе. Каждый из этих цветных светодиодов имеет свой собственный вывод анода (положительный), однако вывод катода (отрицательный) у них общий. Подключите вывод катода светодиода (это будет самый длинный вывод из четырех имеющихся) через последовательный резистор номиналом 220 Ом к выводу «земли» микроконтроллерной платы, а остальные три вывода — к ее выводам ШИМ, как показано на рис. 2.7 (для платы MKR1000 — это контакты 3, 5 и 4, а для платы Arduino 101 — 3, 5 и 6).

Подсоединив светодиод к плате, просверлите небольшое отверстие в теннисном мячике — чуть большего диаметра, чем диаметр светодиода, и наденьте мячик на светодиод, как показано на рис. 2.8. Мячик на светодиоде будет играть роль светорассеивающего абажура. Если светодиоды высвечивают изнутри на мячике слишком яркую точку, ее можно смягчить, обработав верх корпуса светодиода наждачной бумагой.

Макетная плата с модулем MKR1000



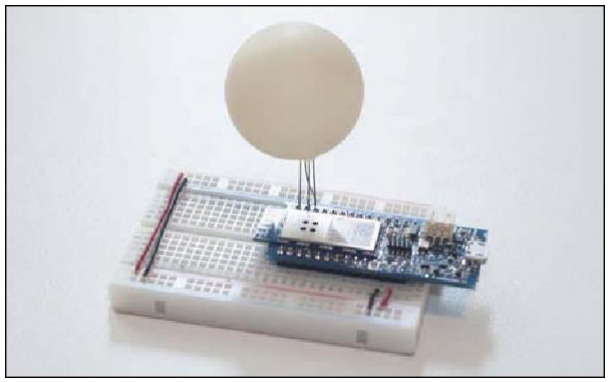
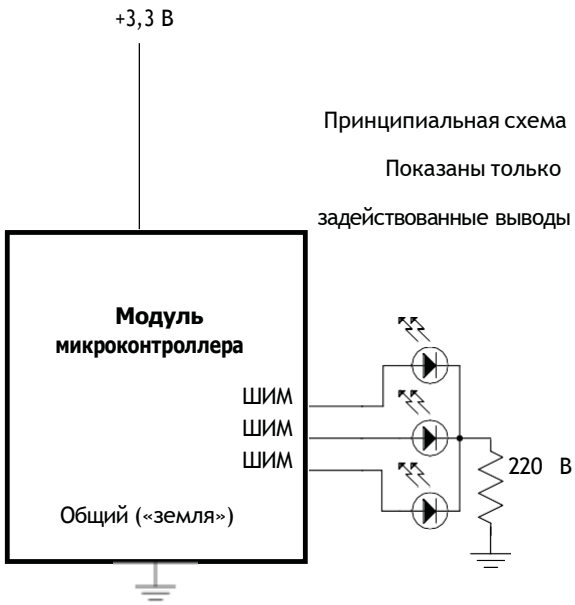


Рис. 2.8. Светодиод с надетым на него теннисным мячиком, играющим роль светорассеивающего абажура



Макетная плата с модулем Arduino 101/Uno

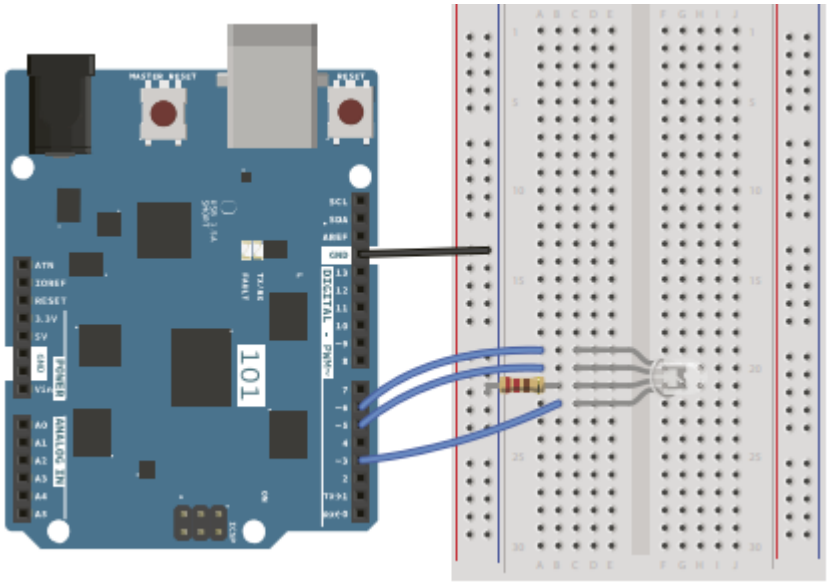


Рис. 2.7. Подключение анодов трехцветного светодиода к выводам ШИМ микроконтроллерной платы: для платы MKR1000 (*вверху*) — это контакты 3, 5 и 4, а для платы Arduino 101 (*внизу*) — 3, 5 и 6.

Создаем коммуникационный протокол

Итак, схему мы собрали, теперь нужно решить, как нам взаимодействовать с микроконтроллером, чтобы управлять светодиодами. Для этого нам требуется создать коммуникационный протокол, который в нашем случае будет очень простым:

- чтобы выбрать цвет светодиода, посылаем первую букву требуемого цвета в нижнем регистре: `r`, `g` или `b`;
- чтобы задать уровень яркости выбранного светодиода, посылаем цифру: от `0` до `9`.

Например, чтобы задать (по шкале от 0 до 9) для красного светодиода уровень яркости — 5, для зеленого — 3 и для синего — 7, посылаем следующий код:

```
r5g3b7
```

Вот и весь протокол — простой до предела. Но нам еще нужно создать программу для микроконтроллера, которая читает по одному байту за раз, а затем решает, что делать — в зависимости от значения считанного байта.

Пишем код

Прежде всего, надо задать *константы*, в которых будут храниться номера контактов, к которым подключены аноды светодиода (эти значения устанавливаются в зависимости от типа платы). Также нужно задать *переменные*: для хранения номера контакта текущего светодиода и для хранения значения яркости:

```
/*
Программа управляет трехцветным светодиодом, ubи выводы r
(красный), g (зеленый) и b (синий) подсоединены к контактам
3, 5 и 4 соответственно.
/*
// Константы для хранения номеров контактов вывода:
// По умолианию заданы номера контактов для платы MKR1000.
// Номера контактов для платы Arduino 101 указаны
// в комментариях.
const int redPin = 3; // для Arduino 101 используем контакт 3
const int greenPin = 5; // для Arduino 101 используем контакт 6
const int bluePin = 4; // для Arduino 101 используем контакт 5

int currentPin = 0; // текущий контакт для задания яркости
int brightness = 0; // текущий уровень яркости
```

Далее метод `setup()` запускает процедуру последовательной передачи данных и задает контакты вывода на выводы светодиода:

```
void setup() {
   // инициируем последовательную передаuу:
   Serial.begin(9600);
   // задаем контакты вывода:
   pinMode(redPin, OUTPUT);
   pinMode(greenPin, OUTPUT);
   pinMode(bluePin, OUTPUT);
}
```

В основном цикле все зависит от наличия байтов для чтения:

```
void loop() {
   // если буфер содержит последовательные данные,
   // сиитываем байт:
   if (Serial.available() > 0) {
      int inByte = Serial.read();
```

Если какие-либо входящие данные поступили, мы обрабатываем из них только несколько предопределенных значений. Так, получив одно из гребуемых нам значений, мы используем операторы `if`, чтобы задать номер контакта и значение яркости:

```
      // реагируем только на знаuения 'r', 'g', 'b' или
      // '0' по '9'.
      // игнорируем любые другие знаuения: if
      (inByte == 'r') {
         currentPin = redPin;
      }
      if (inByte == 'g') { currentPin
         = greenPin;
      }
      if (inByte == 'b') { currentPin
         = bluePin;
      }
```

И, наконец, задаем на текущем выбранном контакте (светодиоде) гребуемый уровень яркости с помощью метода `analogWrite()`:

```
      if (inByte >= '0' && inByte <= '9') {
         // сопоставляем знаuение входящего байта
         // диапазону команды analogRead(): brightness
         = map(inByte, '0', '9', 0, 255);
         // устанавливаем заданную яркость на выбранном
         // контакте:
         analogWrite (currentPin, brihtness);
      }
   }
}
```

Ссылка на гитхаб проектов: https://github.com/TheSuspect17/Internet_of_things_Base.git

Загрузите этот скетч в микроконтроллер, а затем откройте монитор порта, щелкнув по соответствующему значку в панели инструментов окна редактора (рис. 2.9).

В строке ввода окна монитора порта введите следующую команду и нажмите кнопку **Отправить**:

```
r9
```

Теннисный мячик должен подсветиться красным цветом. Теперь попробуйте команду:

```
r2g7
```

Красный цвет погаснет, и мячик подсветится зеленым. Далее введите:

```
g0r0b8
```

Мячик подсветится синим цветом. Вуаля! Мы создали последовательно управляемый разноцветный светильничек.

Если выводимые цвета не соответствуют командам, вводимым с клавиатуры, вы, вероятно, приобрели светодиод не той модели, что указана в списке деталей проекта, и у него другая распиновка. Эта проблема легко решается корректировкой постоянных, сопоставленных с номерами контактов в скетче.
Для управления светодиодом не обязательно требуется окно монитора порта среды Arduino IDE. Любое приложение, способное управлять последовательным портом, может отсылать в Arduino команды управления лампой. Так что, закончив следующий проект, попробуйте написать собственную программу управления светодиодом в среде Processing.

Несколько слов об ASCII

Кстати, вы не обратили внимание на то, что значения переменных в программе указываются в одинарных кавычках? Это потому, что мы для этих переменных используем значения ASCII. Протокол ASCII сопоставляет символам букв и цифр некоторые числовые значения. Например, значение ASCII для буквы 'r' — 114, а для цифры '0' — 48. Заключая символ в одинарные кавычки, мы даем указание микроконтроллеру использовать не сам символ, а его значение ASCII. Например, вот эту строку кода:

```
brightness = map(inByte, '0', '9', 0, 255);
```

можно заменить этой:

```
brightness = map(inByte, 48, 57, 0, 255);
```

поскольку в ASCII символ ‘0' представляется значением 48, а символ ‘9' — значением 57. Применение символов в одинарных кавычках вместо их значений ASCII не является обязательным для работоспособности программы, но де- лает код более удобочитаемым. В первой строке приведенного примера мы используем для представления значений ASCII буквенные символы в одинарных кавычках, а во второй — значения ASCII этих символов. В последствии в книге вам будут встречаться примеры, где используются оба рассмотренных подхода.

Усложняем задачу

В предыдущем проекте мы управляли микроконтроллером с компьютера, используя для этого очень простой протокол. Теперь давайте добавим возможность «новогодней гирлянды».

1) Реализуйте функцию автоматического переключения светодиода в режиме:

1 секунда – горение красного цвета

1 секунда – горение синего цвета

1 секунда – горение зеленого цвета

2) Реализуйте функцию включения и остановки гирлянды по заданной команде
3) Добавьте в схему еще 2 rgb светодиода
4) Запустите гирлянду на полученной схеме из 3 диодов

***

5) Создайте режим для гирлянды, в котором диоды будут переключаться не одновременно в одинаковом режиме, а последовательно переливаясь разными цветами.