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

Лабораторный практикум по робототехнике

Нижневартовск 2025

Содержание

1. Запуск первого скетча

Традиционно программа, которая выполняется платой Arduino, носит название «скетч», с английского переводится как «набросок». Все скетчи создаются и загружаются в плату через среду программирования Arduino IDE. В самой Arduino IDE уже есть скетчи-примеры для основных функциональных задач Arduino.

После запуска среды программирования нужно проверить, чтобы она нашла нашу плату. Смотрим пункт меню *Инструменты–>Порт*, там должен быть выбран порт, на который был установлен драйвер для платы (см. рис.1). После этого в пункте *Инструменты->Плата* нужно убедиться, что выбрана именно нужная Arduino. После этого можно приступать к программированию платы.

Самый простой пример – мигание светодиодом. На самой плате Arduino уже есть светодиод, который можно использовать для этих целей (см. рис. 4). Этот светодиод соединен с пином номер 13, т.е., изменяя напряжение на этом пине, можно управлять светимостью светодиода.

Рис. 4. Внешний вид платы Arduino Uno (встроенный управляемый светодиод выделен красным, возле буквы L).

Можно для проверки работоспособности платы использовать стандартный скетч (см. рис. 5). При выборе этого скетча в среде Arduino IDE в отдельном окне появится примерно следующий код программы (без учета комментариев):

```
void setup()
{
  pinMode(13, OUTPUT); //назначаем пин номер 13 на вывод (т.е. будем
  //управлять подачей на него питания или земли)
}
void loop() {
  digitalWrite(13, HIGH);  // включаем светодиод (HIGH означает подачу
                           //напряжения или питания)
  delay(1000);              // ждем секунду
  digitalWrite(13, LOW);  // выключаем светодиод (LOW означает
                          //отсутствие напряжения, т.е. земля)
  delay(1000);            // ждем секунду
```

```
}
```

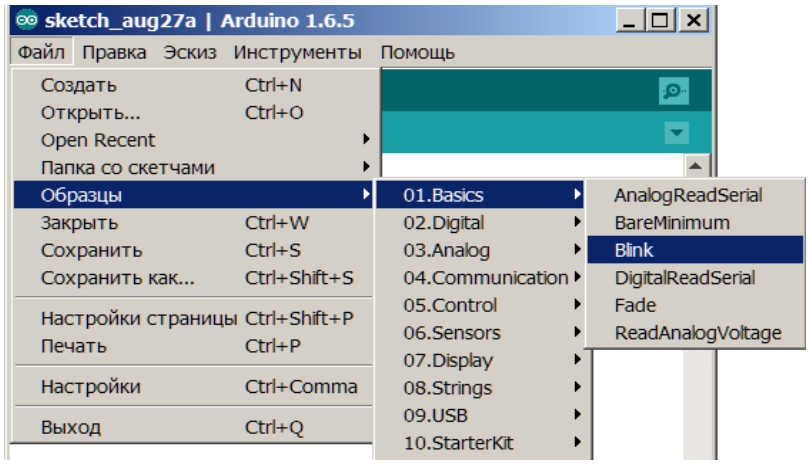


Рис. 5. Выбор скетча Blink (мигание светодиодом).

Первая функция setup выполняется один раз и используется для инициализации переменных и выполнения одноразовых действий, в нашем случае в ней на пин с номером 13 назначается вывод напряжения (а любой пин может быть назначен как на вывод напряжения, так и на считывание значений). А так как пин 13 соединен со встроенным светодиодом, то подача напряжения на этот пин приведет к включению светодиода.

Вторая функция loop выполняется постоянно (в бесконечном цикле) и, соответственно, все, что в ней будет написано, так же будет выполняться бесконечно. В нашем случае эта функция содержит 4 строчки:

`digitalWrite(13, HIGH)` – эта функция меняет напряжение на пине с указанным номером (в данном случае на пин с номером 13 подается напряжение, т.е. светодиод загорится).

`delay(1000)` – функция задержки выполнения программы на указанное количество миллисекунд (в данном случае на 1000 мс, т.е. на 1 секунду).

`digitalWrite(13, LOW)` – эта функция меняет напряжение на пине с указанным номером (в данном случае на пине с номером 13 убирается напряжение, т.е. светодиод гаснет).

`delay(1000)` – функция задержки выполнения программы.

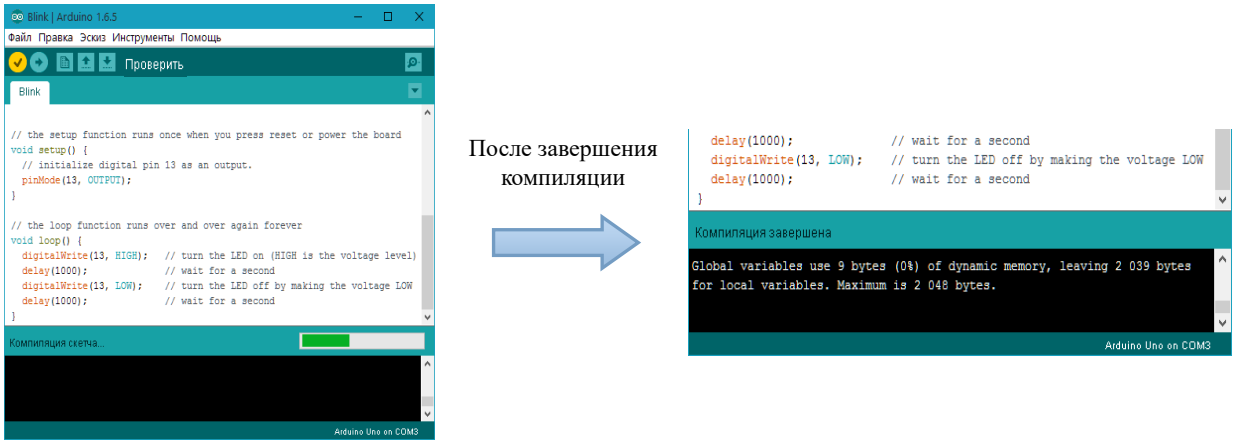


Рис. 6. Запуск проверки скетча, компиляция и результат.

То есть светодиод сначала горит секунду, потом не горит секунду и так в бесконечном цикле. Теперь, когда понятно, что должно произойти, можно запускать скетч на выполнение. Сначала проверим его. Для этого жмем кнопку «*Проверить*» (см. рис. 6) и ждем окончания компиляции скетча. Если все нормально, то внизу окна в черной области появится белый текст с указанием занятой скетчем памяти. Если что-то пошло не так, то появится оранжевый текст с описанием возникшей проблемы.

Если проверка завершилась успешно, то можно загружать скетч на плату Arduino. Жмем кнопку со стрелкой «*Вгрузить*» или «*Загрузить*», в зависимости от версии среды программирования (рядом с кнопкой *Проверить*). Скетч опять будет откомпилирован и загружен на плату Arduino. Обозначенный светодиод (см. рис. 4) начнет мигать.

Создание своего скетча

Теперь можно на основе примера мигания светодиодом создать и сохранить свой скетч. Для этого изменим стандартный скетч мигания светодиодом так, чтобы светодиод 2 раза мигал быстро, потом секунду не горел, потом опять мигал 2 раза быстро и так далее. Вот примерный код для этого:

```
void setup() {
  pinMode(13, OUTPUT);
}
void loop() {
  digitalWrite(13, HIGH);   // включаем светодиод
  delay(200);               // ждем 200 мс
  digitalWrite(13, LOW);    // выключаем светодиод
  delay(200);               // ждем 200 мс
  digitalWrite(13, HIGH);   // включаем светодиод
  delay(200);               // ждем 200 мс
  digitalWrite(13, LOW);    // выключаем светодиод
  delay(1000);              // ждем секунду
}
```

Жмем пункт меню Файл -> Сохранить как... и в диалоговом окне выбираем каталог для сохранения нового скетча и имя скетча (например, blink_new). Если

каталог не менялся, то в папке C:\Users*Ваша_учетка*\Documents\Arduino появится папка с введенным именем скетча, а в ней - сам файл скетча с расширением ino. После этого можно проверять, загружать скетч на плату и наслаждаться результатом.

Задания

1. Изменить скетч для мигания светодиодом, чтоб он мигал в формате SOS (3 раза быстро, 3 раза медленно, 3 раза быстро).
2. Изменить скетч для мигания светодиодом, чтоб он миганием показывал ваше имя или фамилию в формате азбуки Морзе.

2. Работа с несколькими светодиодами и основы работы с беспаечной макетной платой. Работа с кнопкой

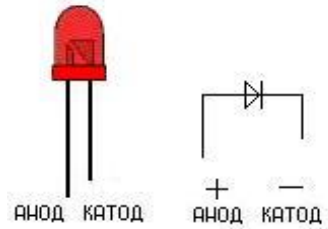


Рис. 7. Светодиод.

Теперь можно приступать к подключению к плате Arduino различной периферии. Начнем со светодиода. Его внешний вид и принцип подключения к источнику тока представлен на рис. 7. Чтобы подключить светодиод к Arduino, нужно использовать резистор. Подойдет любой с номиналом от 220 Ом до 1 кОм (от номинала резистора будет зависеть яркость светодиода, подробнее можно выяснить с помощью закона Ома и любого интернет-поисковика). Для примера возьмем 270 Ом.

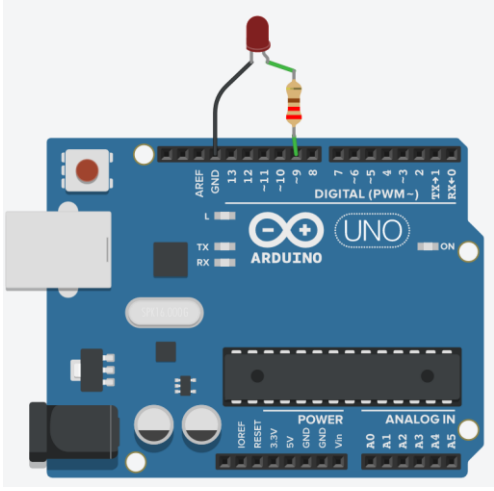


Рис. 8. Схема подключения светодиода к Arduino (изогнутая нога с резистором – анод, т.е. +).

Схема подключения светодиода к Arduino изображена на рис. 8. Можно использовать проводки папа-мама и мама-мама: провод папа-мама втыкаем в разъем GND (земля или "–") Ардуино, в провод втыкаем катод светодиода, в анод вставляем провод мама-мама, в другой конец провода – резистор на 270 Ом (можно 220 Ом), с другой стороны резистора вставляем провод папа-мама, и этот провод уже в Ардуино в порт номер 9.

Код программы можно использовать аналогичный коду в предыдущей лабораторной, заменив строку

```
pinMode(13, OUTPUT);
```

на

pinMode(9, OUTPUT);

то есть поменяв используемый порт на нужный нам (и в digitalWrite поменять номер пина). В результате светодиод будет работать согласно использованному коду.

Теперь попробуем увеличить количество светодиодов. По схеме, аналогичной предыдущей на рис. 8, подключим к пину Gnd внизу платы и пину с номером 8 еще один светодиод (с резистором). Теперь у нас 2 светодиода подключено к плате и можно написать код, который позволит мигать уже двумя светодиодами (сначала секунду горит один, гаснет, секунду горит другой, гаснет и т.д.):

```
void setup() {
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
}
void loop() {
  digitalWrite(8, LOW);     // выключаем второй светодиод
  digitalWrite(9, HIGH);    // включаем первый светодиод
  delay(1000);               // ждем секунду
  digitalWrite(9, LOW);     // выключаем первый светодиод
  digitalWrite(8, HIGH);    // включаем второй светодиод
  delay(1000);               // ждем секунду
}
```

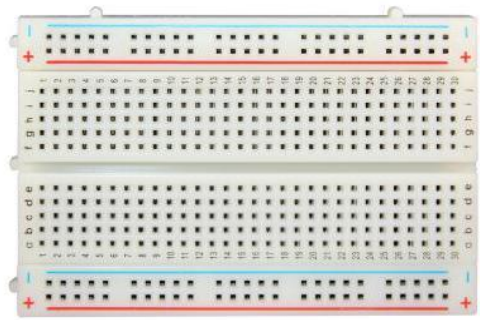


Рис. 9. Беспаечная макетная плата.

Но в данном случае у нас нерациональное использование пинов платы - мы используем 2 выхода Gnd (земля), хотя по законам электротехники можно все светодиоды подключить к одной земле. Поэтому воспользуемся макетной беспаечной платой (см. рис. 9). Применение макетной платы позволяет проверить, наладить и протестировать схему ещё до того, как устройство будет собрано на готовой печатной плате. Это позволяет избежать ошибок при конструировании, а также быстро внести изменения в разрабатываемую схему и тут же проверить результат.

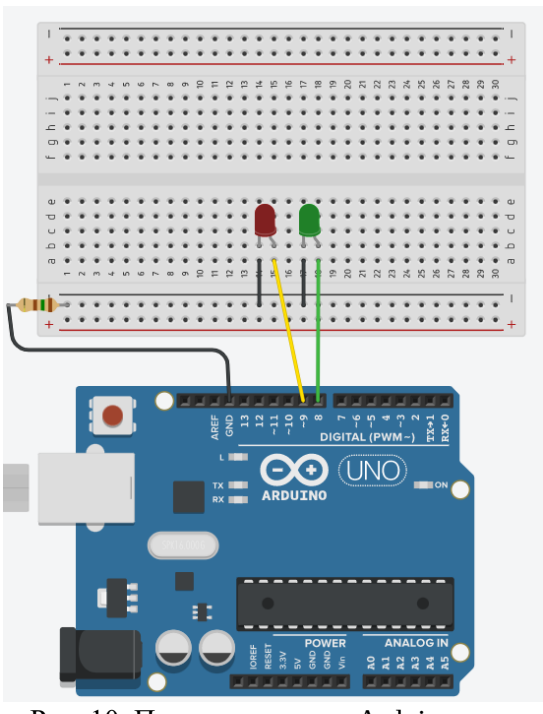


Рис. 10. Присоединение к Arduino двух светодиодов с помощью беспаечной макетной платы.

Ряды, обозначенные «+» и «-», можно подключить к питанию и земле соответственно, и все разъемы в подключенном ряду тоже будут иметь «+» или «-» соответственно. Для примера подсоединим наши 2 светодиода к Arduino при помощи макетной платы.

Черным проводом соединяем пин Gnd платы Arduino с нижним рядом с обозначением «-» на макетной плате (ряд с синей полосой). Остальные провода оставляем как было (см. рис 10). И теперь мы используем на 1 пин платы Arduino меньше, что позволяет подключить больше устройств в более сложных схемах.

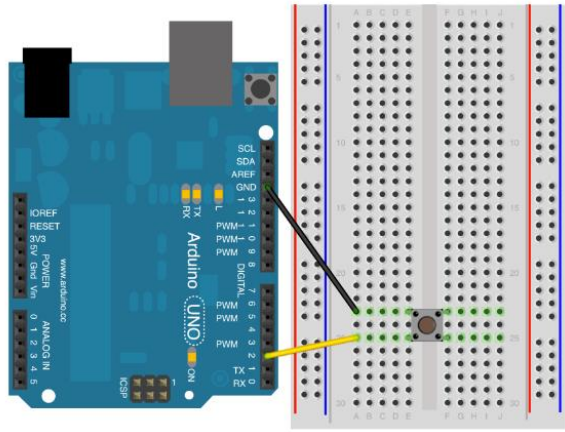


Рис. 11. Подключение кнопки без резистора.

Подключение к нашей схеме со светодиодами кнопки позволит добавить интерактивности. Добавим кнопку к нашей схеме (см. рис. 11). Код для обработки нажатия кнопки может быть примерно следующим:

```
void setup() {
  // задаем 2-ой контакт как входной,
  pinMode(2, INPUT_PULLUP); //а также включаем встроенный подтягивающий резистор
  pinMode(13, OUTPUT);
}

void loop() {
  // сохраняем информацию, полученную от кнопки, в переменную:
  int sensorVal = digitalRead(2);
  // Когда кнопка не нажата, Arduino считывает это как HIGH, а если нажата, то как LOW.
  if (sensorVal == LOW) {  // Если кнопка нажата,
    digitalWrite(13, HIGH); // то светодиод будет гореть,
  }
  else {
    digitalWrite(13, LOW); //а если нет - то не будет

  }
}
```

Для некоторых случаев, например, для вкл/выкл светодиода каждым очередным нажатием кнопки или подсчета кол-ва нажатий на кнопку, удобнее использовать следующий вариант кода:

```
//переменная state будет отвечать за состояние кнопки, ledState - за состояние светодиода
bool state = 0, ledState = 0;
void setup() {
  pinMode(2, INPUT_PULLUP);
  pinMode(13, OUTPUT);
}

void loop() {
  bool but = digitalRead(2); //считываем состояние кнопки
  if (but == LOW) state = 1; //если нажали, задаем переменной state 1 (true)
  if (but == HIGH && state == 1){ //если кнопку отпустили
    ledState = !ledState; //то переключаем переменную для состояния светодиода
    digitalWrite(13, ledState); //и передаем это состояние на светодиод
//здесь можно сделать подсчет кол-ва нажатий на кнопку
    state = 0; //сбрасываем состояние кнопки
  }
  delay(15); //небольшая задержка для правильной работы всей схемы
}
```

Задания

I. Подключить 3 светодиода через макетную плату и сделать скетч для их работы по следующей схеме (по вариантам):

1. поочередное зажигание и затухание каждого.
2. зажигаются все вместе, гаснут каждый по очереди.

3. зажигаются по очереди, гаснут все вместе.
4. зажигаются 2, гаснут, зажигается третий и гаснет.
5. зажигается 1, гаснет, зажигаются 2 других и гаснут.
6. зажигается 1, гаснет, зажигаются 1 и 2, гаснут, зажигаются 1, 2, 3, гаснут.
7. 1 раз мигает первый, гаснет, 2 раза мигает второй, гаснет, 3 раза мигает третий, гаснет.

II. Схема состоит из 3-х светодиодов и кнопки, расположенных на беспаечной макетной плате. Реализовать по вариантам:

1. при нажатии на кнопку загорается первый светодиод, гаснет, при втором нажатии загорается второй, гаснет, при третьем нажатии загорается третий и гаснет.
2. при нажатии на кнопку загораются первый и второй светодиоды, гаснут, при втором нажатии загораются второй и третий светодиоды и гаснут.
3. при первом нажатии кнопки начинает мигать первый светодиод, при втором – первый и второй светодиод, при третьем – все три.
4. при первом нажатии кнопки начинает мигать первый светодиод, при втором нажатии первый светодиод гаснет и начинает мигать второй светодиод, при третьем нажатии второй светодиод гаснет и начинает мигать третий светодиод.
5. при первом нажатии загораются 3 светодиода, гаснут, при втором нажатии загораются 2, гаснут, при третьем нажатии загорается 1 светодиод.
6. при первом нажатии кнопки начинает мигать первый светодиод, при втором – первый и третий светодиод, при третьем – второй.
7. при первом нажатии кнопки начинает мигать первый светодиод, при втором нажатии первый светодиод гаснет и начинает мигать третий и второй светодиод, при третьем нажатии второй светодиод гаснет и начинает мигать первый.

3. Работа с DC-мотором через драйвер мотора. Функция millis

С помощью Arduino можно легко управлять DC-мотором - направлением вращения и скоростью вращения. Но напрямую подключать мотор к Arduino нельзя – можно повредить плату, нужно использовать специальную микросхему – драйвер мотора. Это специальная плата, к которой подключается мотор, а сам драйвер мотора подключается к Arduino. Сама плата драйвера мотора может выглядеть по-разному (см. рис. 12).

Схематически плату драйвера мотора можно изобразить как на рис. 13, где цифрами обозначены:

1, 2 - выходы для подключения DC-мотора (можно подключить 2 мотора к плате);

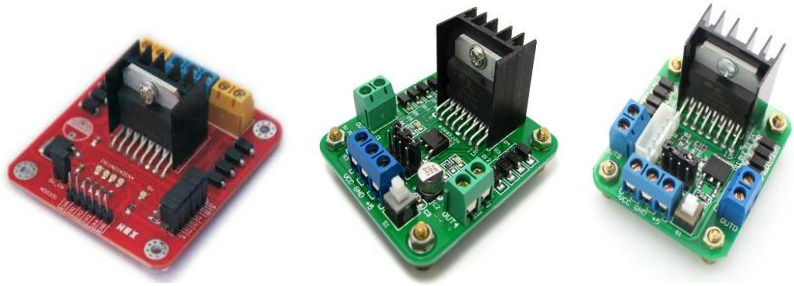


Рис. 12. Разные виды платы драйвера мотора.

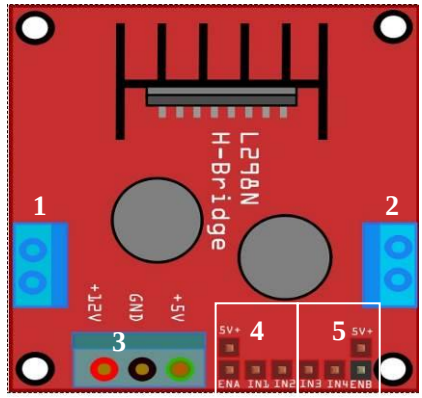


Рис. 13. Схематическое изображение платы драйвера мотора.

3 - входы для подключения питания (от 5 вольт и выше - вход с обозначением +12V, от него питаются моторы), земли (GND) и дополнительного питания самой платы, если на вход +12V будет подаваться большое напряжение (выше 12 вольт);

4 - группа пинов для управления первым двигателем (ENA, IN1, IN2);

5 - группа пинов для управления вторым двигателем (ENB, IN3, IN4).

Схема подключения мотора через плату драйвера изображена на рис. 14. Обязательно должна присутствовать перемычка, соединяющая 2 крайних нижних пина справа (возле отверстия, обозначены ENB и 5V+). Она определяет - подается питание на мотор или нет.

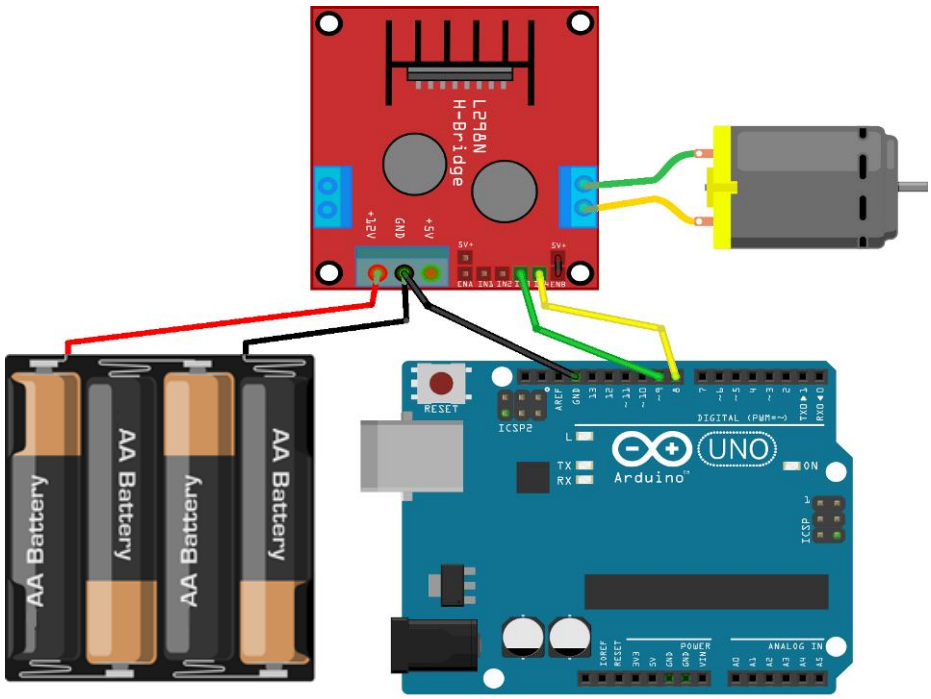


Рис. 14. Схема подключения мотора через плату драйвера.

Пример скетча - 2 секунды мотор вращается в одну сторону, 2 - в другую.

```
int IN1=8;
int IN2=9;
void setup()
{
```

```
 pinMode(IN1,OUTPUT);
 pinMode(IN2,OUTPUT);
}
void loop()
{
 digitalWrite(IN1,LOW);// вращение в одну сторону
 digitalWrite(IN2,HIGH);
 delay(2000);
 digitalWrite(IN1,HIGH);// вращение в другую сторону
 digitalWrite(IN2,LOW);
 delay(2000);
}
```

Если мотор не крутится, то нужно проверить батарейки – есть ли в них заряд.

Еще один вид драйвера мотора - Motor Shield от российской компании Amperka (рис. 14б). Он устанавливается сверху прямо на Arduino и позволяет управлять двумя моторами (рис. 14в). Для коммуникации с микроконтроллером используются цифровые контакты Arduino:

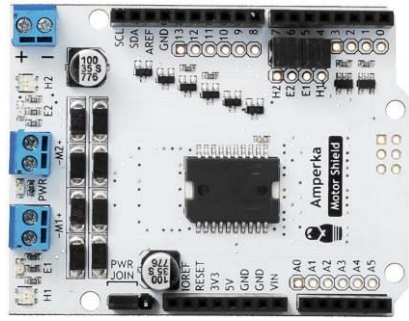


Рис. 14б. Amperka Motor Shield.

4 — направление, мотор M1;

5 — скорость (ШИМ – подробнее в лаб.раб. 4), мотор M1;

6 — скорость (ШИМ), мотор M2;

7 — направление, мотор M2.

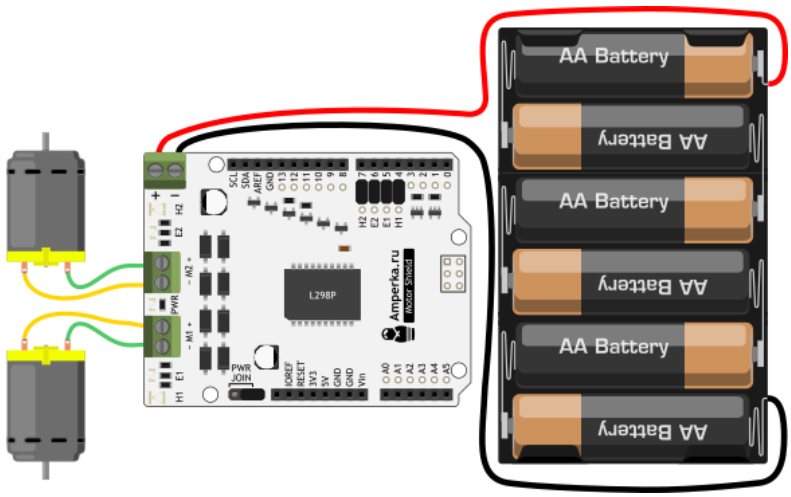


Рис. 14в. Подключение двух моторов и питания к Motor Shield.

Код для работы одного мотора может быть примерно следующим:

```
#define SPEED_1 5 //константы для пинов
#define DIR_1 4

void setup() {
  pinMode(4, OUTPUT); // настраиваем выводы платы 4 и 5 на вывод сигналов
  pinMode(5, OUTPUT);
}

void loop() {
  digitalWrite(DIR_1, LOW);   // устанавливаем направление мотора «M1» в одну сторону
  analogWrite(SPEED_1, 255);   //включаем мотор на максимальной скорости
```

```
  delay(1000);   // ждём одну секунду
  digitalWrite(DIR_1, HIGH);   //устанавливаем направление мотора «M1» в другую сторону
  delay(1000);   // ждём одну секунду
  analogWrite(SPEED_1, 0); // выключаем первый мотор
  delay(1000);
}
```

Разберем еще одну полезную функцию – millis(). Она возвращает время, прошедшее с момента запуска Arduino в миллисекундах. Поэтому данную функцию можно использовать для замены delay, которая просто останавливает работу микроконтроллера.

Например, мигание светодиодом с использованием этой функции:

```
bool led = HIGH; //переменная для хранения состояния светодиода
long time = 0; //переменная для хранения времени

void setup() {
  pinMode(13, OUTPUT);
  time = millis(); //сохраняем текущее значение времени в переменную
}

void loop() {
  digitalWrite(13, led); //передаем в светодиод текущее значение его состояния
//если разница между текущим временем и сохраненным стало равным или больше 1000
  if ( (millis() - time)>=1000 ) {
    led = !led;  //то меняем значении для светодиода
    time = millis(); //и сохраняем новое значение времени (т.е. сбрасываем таймер)
  }
}
```

Другой пример – вращение мотора 2 секунды в одну сторону, потом 2 секунды в другую и т.д.

```
int IN1=8;
int IN2=9;
long time = 0; // переменная для хранения времени
boolean m1 = LOW; // переменные для хранения значений для управления мотором
boolean m2 = HIGH;
void setup()
{
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  time = millis(); // сохраняем начальное значение времени
  digitalWrite(IN1,m1); // включаем мотор с текущими параметрами
  digitalWrite(IN2,m2);
}
void loop()
{
  if(millis()-time >= 2000){ // если прошло 2 секунды с прошлого замера
    m1 = !m1;  // меняем значения для управления мотором,
    m2 = !m2; // чтобы поменять направление вращения
    digitalWrite(IN1,m1);  // применяем новые параметры для мотора
```

```
    digitalWrite(IN2,m2);
    time = millis(); // новое значение времени
  }
}
```

Задания

Собрать схему из 3-х светодиодов, кнопки и мотора с драйвером мотора. Запрограммировать следующие действия (по вариантам):

1. при нажатии на кнопку мотор начинает крутиться и по очереди загораются все светодиоды, второй раз нажимаем на кнопку – мотор крутится в другую сторону и светодиоды гаснут.
2. добавить вторую кнопку. При нажатии на первую кнопку мотор работает и светодиоды горят, при нажатии на вторую кнопку мотор останавливается и светодиоды гаснут.
3. пока нажата кнопка – мотор крутится, и все светодиоды горят, как только кнопка не нажата – мотор останавливается и светодиоды гаснут.
4. пока нажата кнопка – мотор крутится в одну сторону, и горят светодиоды 1 и 3, как только кнопка не нажата – мотор крутится в другую сторону и светодиод 2 горит.
5. добавить вторую кнопку. При нажатии на первую кнопку мотор начинает крутиться и зажигаются светодиоды 1 и 2, при нажатии на вторую кнопку мотор крутится в другую сторону, гаснут 1 и 2 светодиоды и зажигается 3-й светодиод.
6. при нажатии на кнопку мотор начинает крутиться, светодиоды загораются все сразу, второй раз нажимаем – светодиоды по одному гаснут и мотор прекращает крутиться.
7. добавить вторую кнопку. При нажатии на первую кнопку мотор начинает крутиться в одну сторону и светодиоды горят, при отпускании мотор останавливается. При нажатии на вторую кнопку мотор крутится в другую сторону, светодиоды гаснут, при отпускании кнопки мотор останавливается.

4. Широтно-импульсная модуляция. Мониторинг последовательного порта в Arduino IDE

ШИМ (широтно-импульсная модуляция) или по-английски PWM (pulse-width modulation) – это способ задания аналогового сигнала цифровым методом, то есть из

цифрового выхода, дающего только нули и единицы, получить какие-то плавно меняющиеся величины.

Его применяют для задания мощности выходного сигнала, т.е. чтобы, например, регулировать яркость светодиода или скорость вращения мотора.

Для использования ШИМ с Arduino нужно знать, какие порты поддерживают его. Обычно такие цифровые порты имеют знак ~ возле своего номера. Для Arduino Uno и Nano это порты 3, 5, 6, 9, 10, 11. Для Arduino Mega – 2-13. Для задания параметров ШИМ используется функция `analogWrite(ledPin, fadeValue)`, где `ledPin` – пин для сигнала в формате ШИМ, `fadeValue` – значение ШИМ от 0 до 255.

Пример использования ШИМ для постепенного усиления яркости и затем затухания светодиода:

```
int ledPin=11;
void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
    //в цикле будем менять значение для ШИМ от 0 до 255
  for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) {
    analogWrite(ledPin, fadeValue); //выдаем сигнал в формате ШИМ на нужный пин
    delay(30); //ждем 30 миллисекунд
  }
  // тоже самое, но уже для постепенного затухания светодиода
  for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= 5) {
    analogWrite(ledPin, fadeValue);
    delay(30);
  }
}
```

Для отладочного вывода в Arduino IDE можно использовать вывод информации в последовательный порт с использованием функций Serial.print и Serial.println. Для использования последовательного порта необходимо открыть к нему доступ, вызвав в функции setup() функцию `Serial.begin(speed);` где `speed` – скорость последовательного порта.

Чтобы увидеть, что выводится в последовательный порт, в среде программирования Arduino IDE нажимаем пункт меню Инструменты -> Монитор последовательного порта. Откроется дополнительное окно, в котором будет видно все, что мы выводим в последовательный порт. Необходимо, чтобы скорость, заданная для последовательного порта в Serial.begin и скорость, указываемая в нижнем правом углу открывшегося окна (см. рис. 15), совпадали.

Например, добавим в предыдущий пример с ШИМ отладочный вывод о яркости

светодиода:

```
int ledPin=11;
void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600); //открываем доступ к последовательному порту
}

void loop() { //в цикле for будем менять значение для ШИМ от 0 до 255
  for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) {
    analogWrite(ledPin, fadeValue); //выдаем сигнал в формате ШИМ на нужный пин
    if (fadeValue % 50 == 0) { //если значение яркости кратно 50,
      Serial.print("led light = "); // то выводим в порт надпись "led light = "
      Serial.println(fadeValue); //и значение текущей яркости светодиода
    }
    delay(30); //ждем 30 миллисекунд
  }
  // тоже самое, но уже для постепенного затухания светодиода
  for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= 5) {
    analogWrite(ledPin, fadeValue);
    if (fadeValue % 50 == 0) {
      Serial.print("led light = ");
      Serial.println(fadeValue);
    }
    delay(30);
  }
}
```

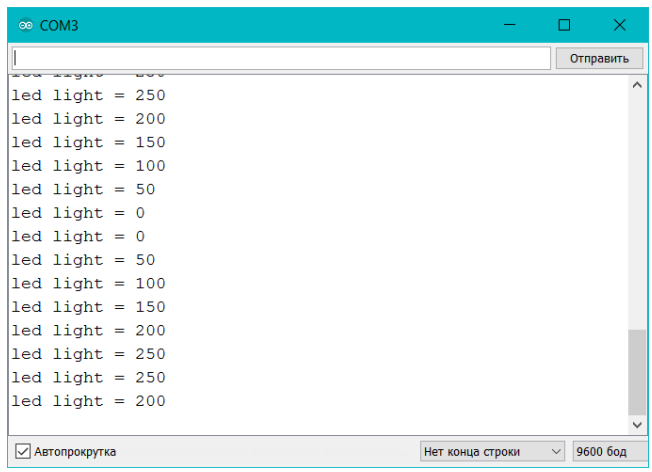


Рис. 15. Вывод текста в последовательный порт.

И теперь в монитор порта будет выводиться значение яркости светодиода (рис. 15).

Можно также и вводить данные с помощью монитора порта, в строке с кнопкой Отправить (рис. 15). Получать оттуда данные можно либо с помощью функции Serial.read() в случае символьных данных, либо с помощью функции Serial.parseInt() для целых чисел (или Serial.parseFloat() для вещественных). Для получения всей введенной строки удобно использовать функцию Serial.readStringUntil('\n'), которая возвращает значение типа String. А если нужно введенное значение из строки преобразовать в число, то можно использовать функцию toInt() класса String:

```
String s = Serial.readStringUntil('\n');
int x = s.toInt();
```

Например, сделаем ввод с монитора последовательного порта значения для задания скорости вращения мотора:

```
int pinA = 10;  // пины 10 и 11 назначаем на управление мотором
int pinB = 11;
int speed=0;

void setup() {
  pinMode(pinA, OUTPUT);
  pinMode(pinB, OUTPUT);
  Serial.begin(9600);
  digitalWrite(pinA,LOW);
```

```
  digitalWrite(pinB,LOW);
}

void loop() {
 if (Serial.available()>0) { //если есть данные из последовательного порта
    speed = Serial.parseInt(); //записываем в переменную speed данные из порта
    analogWrite(pinA, speed); //устанавливаем значение для одного из портов,
                              //к которым подключен мотор
    Serial.print("motor speed = ");
    Serial.println(speed);
  }
}
```

И теперь мотор будет крутиться с той скоростью, которую мы зададим через монитор последовательного порта (рис. 16). Просто вводим нужное число в строку и нажимаем кнопку Отправить.

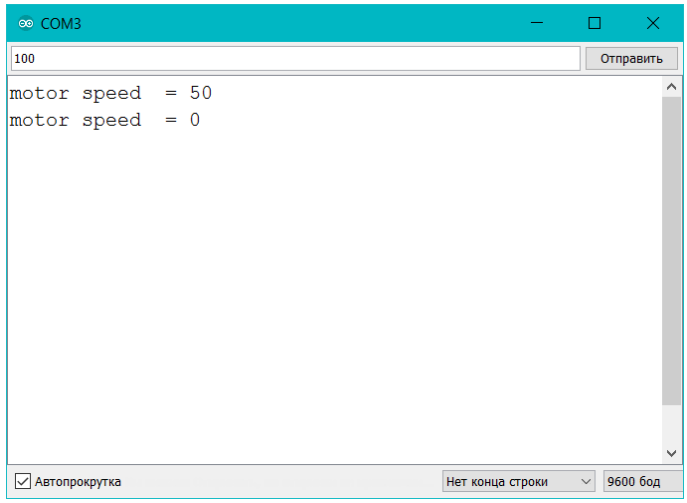


Рис. 16. Ввод данных из монитора последовательного порта.

Задания

(обязательно использовать монитор последовательного порта для отладочного вывода или ввода данных)

1. По нажатию кнопки светодиод должен постепенно разгораться. При втором нажатии – постепенно затухать.
2. Пока держим кнопку – светодиод постепенно разгорается, когда зажегся на полную – постепенно затухает, отпускаем кнопку – светодиод останавливается на текущем значении.
3. При первом нажатии кнопки светодиод загорается на 1/3 от полной яркости, при втором нажатии – на 2/3, при третьем – светодиод загорается на полную.
4. При первом нажатии кнопки мотор начинает работать на ¼ от полной мощности, при втором нажатии – на ½, при третьем – на ¾, при четвертом – мотор крутится на полную.
5. В наличии светодиод и 2 кнопки. Постепенные нажатия первой кнопки увеличивают яркость светодиода (можно на 10%), постепенные нажатия второй кнопки – уменьшают яркость (тоже на 10%).
6. По нажатию кнопки первый светодиод должен постепенно затухать, а второй– постепенно разгораться. При втором нажатии – наоборот.

7. С монитора порта вводится значение в процентах, и светодиод загорается с соответствующей яркостью.

5. Работа с сервомотором. Потенциометр

Сервомотор представляет из себя мотор, который можно поворачивать на определенное количество градусов (как правило, от 0 до 180 или от 0 до 60). К Arduino сервомотор можно подключить по схеме на рис. 17.

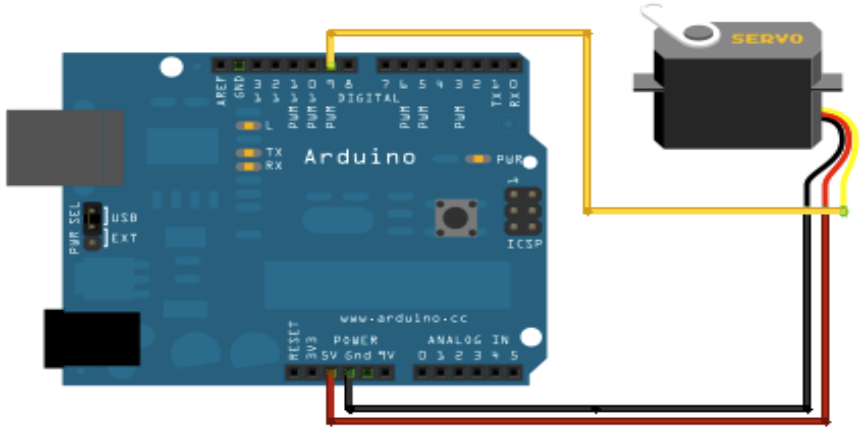


Рис. 17. Схема подключения сервомотора.

У сервомоторов обычно 3 провода: черный или коричневый - земля (GND), подключаем к пину GND; красный – питание, подключаем к пину +5V; желтый, оранжевый или белый – управление, подключаем к любому пину с PWM (см. предыдущую лабораторную).

Вот примерный код для управления сервомотором:

```
#include <Servo.h> //подключаем библиотеку для работы с сервомотором
Servo myservo;  //создаем переменную-объект для управления сервомотором
int pos = 0;    //переменная для хранения угла поворота сервомотора

void setup()
{
  myservo.attach(9);  // назначаем управляющий пин для сервомотора
}                        //(номер 9)

void loop()
{
  for(pos=0; pos<=180; pos += 1) //делаем цикл от 0 до 180 с шагом 1
  {
    myservo.write(pos); // поворачиваем сервомотор на позицию pos
    delay(15);       //ждем 15 мс, чтобы сервомотор достиг позиции
  }
  for(pos = 180; pos>=0; pos-=1) //делаем цикл от 180 до 0 с шагом -1
  {
    myservo.write(pos); // поворачиваем сервомотор на позицию pos
    delay(15);         //ждем 15 мс, чтобы сервомотор достиг позиции
  }
}
```

Запустив этот скетч, можно увидеть, что сервомотор сначала поворачивается от 0 до 180 градусов, потом в обратную сторону.

Потенциометры – это регулируемые делители напряжения, которые предназначены для регулирования напряжения при неизменной величине тока, и выполненные по типу переменного резистора (рис. 18).

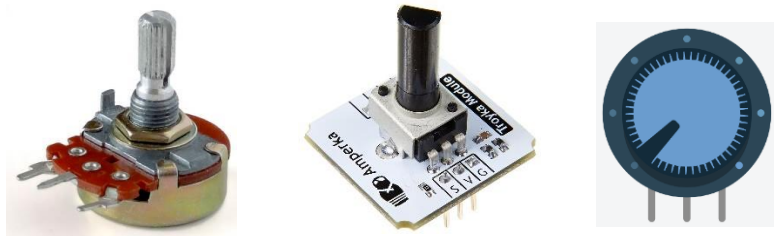


Рис. 18. Потенциометры.

Потенциометр очень удобно использовать для ручного задания параметров вроде времени таймера, яркости или контрастности дисплея, скорости вращения двигателя, высоты звука. Модуль с потенциометром по своей сути — это аналоговый сенсор, который сообщает микроконтроллеру положение ручки потенциометра через аналоговый вход Arduino (пины A0-A5).

Подключение потенциометра к Arduino происходит с помощью 3 проводов: питание, земля и сигнальный (обычно средний выход, соединяем его с A0) – рис. 19.

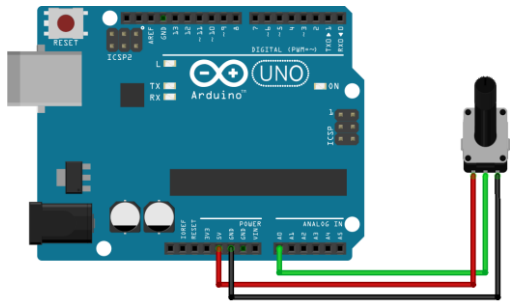


Рис. 19. Подключение к Arduino

Рассмотрим пример с использованием потенциометра для регулирования яркости светодиода:

```
int ledPin=11;  //пин светодиода
int dx = 0;  //переменные для значения с пина A0
int dx2 = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(A0, INPUT);
  Serial.begin(9600);
}

void loop() {
    dx = analogRead(A0); //считываем сигнал с нужного пина
    if (dx2!=dx){  //если значение поменялось
      Serial.print("dx = "); //выводим его
      Serial.print(dx);
      dx2 = dx;  //присваиваем новое значение
      Serial.print(", ledPin = ");
      Serial.println(dx/4); //выводим преобразованное значение (0..255)
      analogWrite(ledPin, dx/4);  //и передаем его на светодиод
    }
}
```

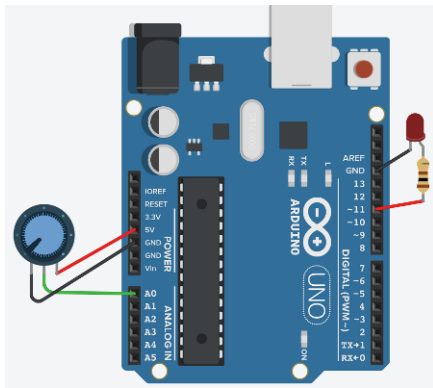


Рис. 20. Потенциометр и светодиод

Схема подключения представлена на рис. 20, вывод в монитор порта примерно следующий:

```
dx = 246, ledPin = 61
dx = 266, ledPin = 66
dx = 389, ledPin = 97
dx = 409, ledPin = 102
dx = 491, ledPin = 122
```

Задания

I. Для сервомотора:

1. С монитора последовательного порта вводится число и сервомотор должен повернуться на этот угол. Если угол неправильный, то нужно вывести сообщение об этом.
2. Пока нажата кнопка, сервомотор вращается.
3. С монитора последовательного порта вводятся 2 числа – значения двух углов. И сервомотор поворачивается между значениями этих углов в бесконечном цикле.
4. Нажимаем одну кнопку – сервомотор вращается в одну сторону 1 секунду. Нажимаем вторую кнопку – сервомотор вращается в другую сторону 1 секунду.
5. С монитора последовательного порта вводится число, означающее кол-во вращений сервомотора от начальной до конечной позиции. Например, если ввели 1, то сервомотор вращается 1 раз в одну сторону, а если ввели 2 – туда-обратно, 3 – туда, обратно и опять туда, и т.д.
6. Пока держим одну кнопку – сервомотор вращается в одну сторону. Держим вторую кнопку – сервомотор вращается в другую сторону. Если ни одна кнопка не зажата – мотор не двигается.
7. С монитора последовательного порта вводится число, означающее угол, на который нужно повернуть сервомотор. Сервомотор поворачивается и останавливается. При нажатии кнопки сервомотор возвращается на 0.

II. Для потенциометра:

1. Схема состоит из потенциометра и мотора. В исходном состоянии мотор не крутится. При перемещении потенциометра меняется скорость вращения мотора (увеличивается или уменьшается).
2. Схема состоит из потенциометра и 5 светодиодов. В зависимости от сигнала потенциометра светит соответствующее кол-во светодиодов (при 0 – 0 светодиодов, при 1023 – 5 светодиодов).
3. Схема состоит из потенциометра и сервомотора. В зависимости от сигнала потенциометра сервомотор поворачивается от 0 до 180.
4. Схема состоит из потенциометра и мотора. Скорость и направление мотора должны регулироваться потенциометром: крайнее положение потенциометра

слева – мотор крутится с максимальной скоростью влево, крутим вправо – скорость уменьшается, доходим до середины – мотор останавливается. После половины продолжаем крутить – мотор начинает вращаться вправо, увеличивая скорость в зависимости от поворота, максимальная скорость вращения мотора вправо – при крайнем правом положении потенциометра.

5. Схема состоит из потенциометра и 5 светодиодов. В зависимости от сигнала потенциометра горит соответствующий светодиод: при 0 – первый, потом при повороте потенциометра первый гаснет и загорается 2й, потом 2й гаснет и загорается 3й, и т.д., при крайнем правом положении – горит только 5й, и то же самое в обратную сторону. Т.е. горящий светодиод как бы перемещается в зависимости от положения потенциометра.
6. Схема состоит из потенциометра, 5 светодиодов и кнопки. Кнопка переключает текущий светодиод, а потенциометр настраивает его яркость.
7. Схема состоит из потенциометра, мотора и кнопки. Кнопка переключает направление вращения мотора, потенциометр – скорость вращения.

6. Работа со сдвиговым регистром

В лабораторной 2 мы подключали 2 светодиода к Arduino, используя 2 пина для управления. Но если нам нужно подключить, к примеру, 8 светодиодов, то придется использовать 8 пинов для управления каждым, а если понадобится подключить 16 светодиодов или еще больше? Тратить 1 пин Arduino для каждого светодиода – расточительно. Для более удобной работы с управлением несколькими светодиодами (и не только) можно использовать специальную микросхему – сдвиговый регистр 74HC595.

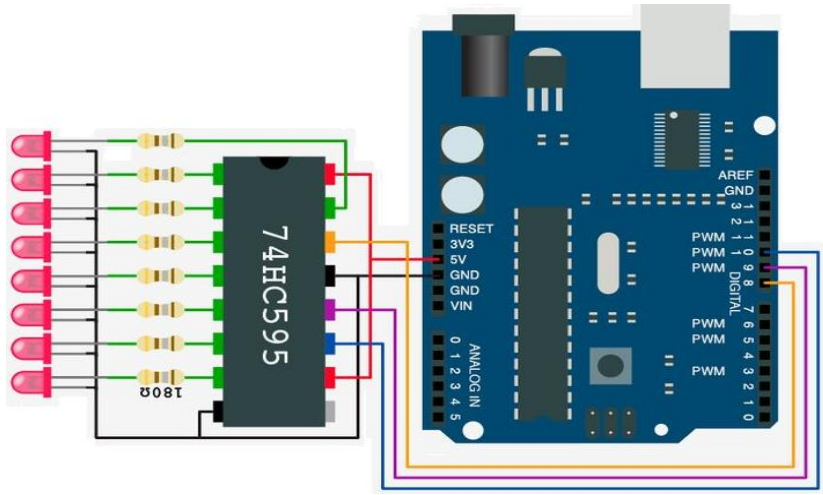


Рис. 21. Схема подключения микросхемы 74HC595.

Он подсоединяется к Arduino через 5 проводов (питание, земля и 3 управляющих) и позволяет управлять 8-ю выходами. Поддерживается подключение нескольких микросхем 74HC595 друг к другу, увеличивая число управляемых выходов. Схема подключения сдвигового регистра с 8-ю светодиодами к Arduino представлена на рис. 21.

Правильно соблюсти подключение поможет полукруглая выемка в верхней части микросхемы. Подключение к Arduino через макетную плату показано на рис. 22.

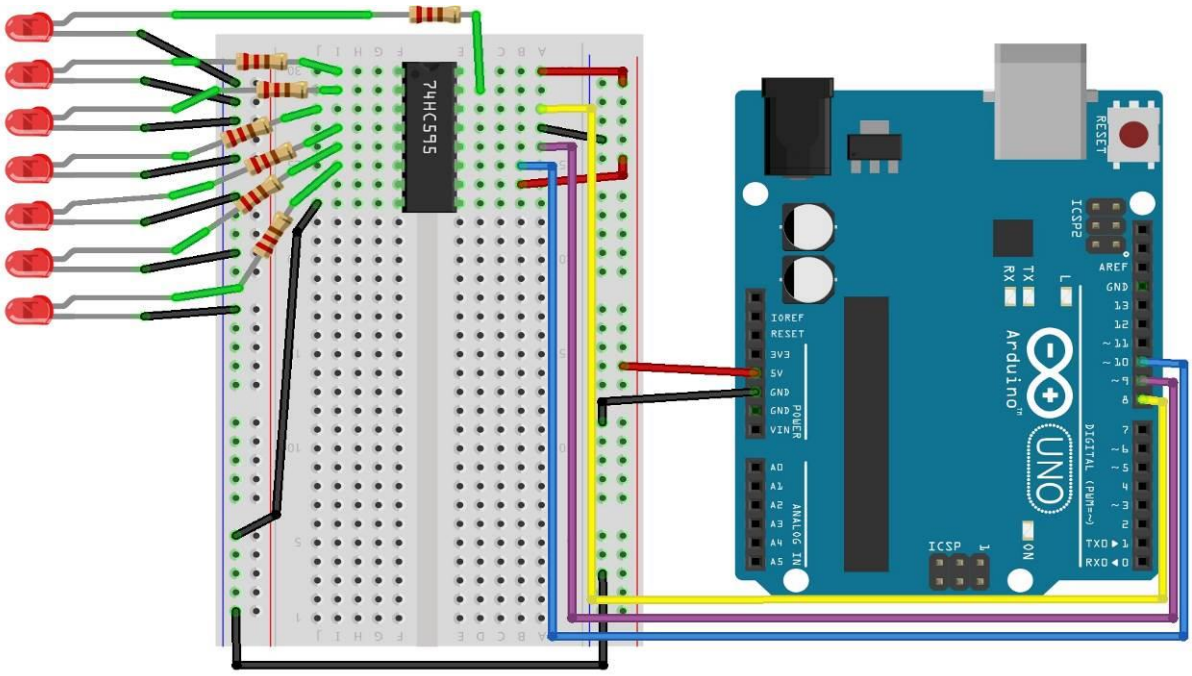


Рис. 22. Схема подключения сдвигового регистра со светодиодами через макетную плату.

Пример скетча с использованием сдвигового регистра:

```
int DS_pin = 8;     //назначаем пины 8,9,10 для управления
int STCP_pin = 9; //сдвиговым регистром
int SHCP_pin = 10;
boolean registers[8]; //массив логических значений для управления выходами
                      //сдвигового регистра

void writereg() //функция для передачи данных на сдвиговый регистр
{//она позволяет синхронизировать передачу данных на сдвиговый регистр
```

```
  digitalWrite(STCP_pin, LOW); //открываем сдвиговый регистр на прием данных
  for (int i = 7; i >= 0; i--)
  {
    digitalWrite(SHCP_pin, LOW); //открываем порт регистра для приема 1 значения
    digitalWrite(DS_pin, registers[i] ); //посылаем логическое значение в
                                         //сдвиговый регистр
    digitalWrite(SHCP_pin, HIGH); //закрываем порт регистра для приема
  }
  digitalWrite(STCP_pin, HIGH); //закрываем сдвиговый регистр
}

void setup()
{
  pinMode(DS_pin, OUTPUT); //назначаем пины для управления сдвиговым регистром
  pinMode(STCP_pin, OUTPUT);
  pinMode(SHCP_pin, OUTPUT);
  writereg();    //вызываем функцию для работы со сдвиговым регистром первый раз
}                //для сброса его портов

void loop()
{//в цикле будем устанавливать значение каждому элементу массива и передавать весь
  for (int i = 0; i < 8; i++)  //массив в сдвиговый регистр
  {      // и соответствующие светодиоды будут загораться
    registers[i] = HIGH;
    writereg();//вызов функции для работы со сдвиговым регистром
    delay(100);                    //задержка 100мс
  }
  for (int i = 7; i >= 0; i--)  //аналогично предыдущему циклу,
  {      //только теперь светодиоды будут гаснуть
    registers[i] = LOW;
    writereg();
    delay(100);
  }
}
```

В результате, запустив скетч, можно увидеть, как загораются по одному все светодиоды, а потом также по одному гаснут.

Задания

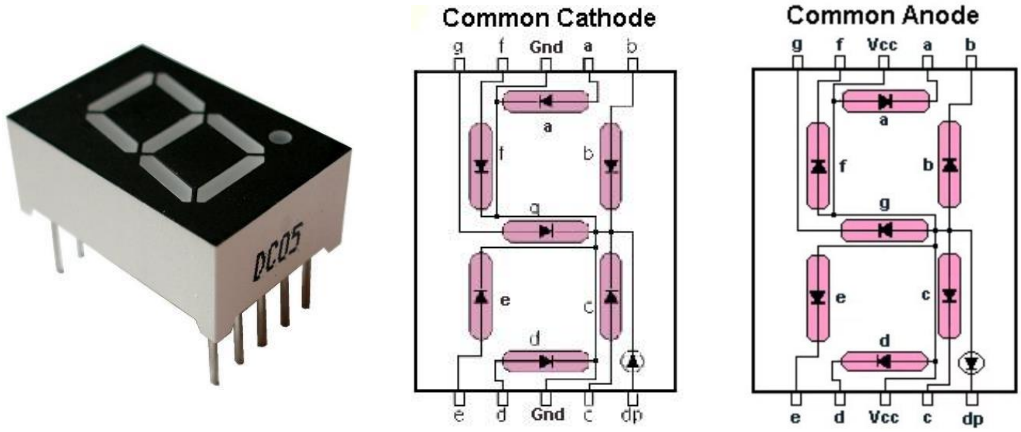
Есть 8 светодиодов, подключенных к Arduino через сдвиговый регистр.

Выполнить следующее (по вариантам):

1. Светодиоды загораются через один, гаснут, загораются другие через один.
2. Светодиоды загораются по одному, гаснут все вместе.
3. Загораются первые 4 светодиода, гаснут, загораются другие 4 светодиода.
4. Светодиоды загораются и гаснут парами.
5. Светодиоды загораются все вместе, а гаснут по одному.
6. Каждый светодиод загорается и тухнет, потом следующий и т.д.
7. Светодиоды загораются по 1 с каждой стороны, сходясь к центру.

7. Семисегментный индикатор

Семисегментный индикатор — устройство отображения цифровой информации, может отображать арабские цифры. Семисегментный индикатор, как говорит его название, состоит из семи элементов индикации (сегментов), включающихся и выключающихся по отдельности (рис. 23). Различают семисегментные индикаторы с общей землей и общим питанием.



а) б)

Рис. 23. Семисегментный индикатор и схема его внутреннего устройства (а – с общей землей, б – с общим питанием).

Для подключения к Arduino есть минимум 2 варианта:

1. **Напрямую** (рис. 24).

Для подключения понадобятся 8 проводов (1 для питания или земли, в зависимости от вида индикатора, и 7 для индикаторов, плюс еще 1, если понадобится включать точку).

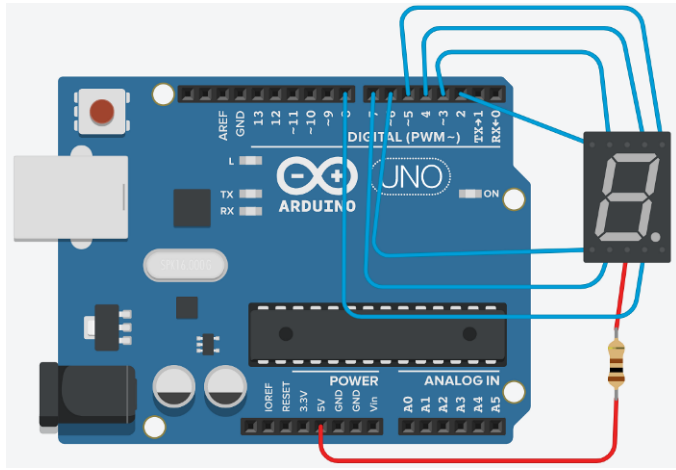


Рис. 24. Присоединение к Arduino семисегментного индикатора напрямую.

Для программирования работы индикатора удобнее будет сделать функции для показа каждой нужной цифры и вызывать их в нужный момент:

```
//задаем номера портов, к которым подключены сегменты, соответствующие буквам
int g = 2, f = 3, a = 4, b = 5, e = 6, d = 7, c = 8;

void clear(){ //функция для выключения всех сегментов
    for (int i = 2; i<=8; i++){
      digitalWrite(i, HIGH);  //если будет с общей землей - LOW
    }
}

void zero(){  //функция для вывода нуля
    clear();
    digitalWrite(a, LOW);  //если будет с общей землей - HIGH во всех строках здесь и далее
```

```
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
}

void one(){  //функция для вывода единицы
    clear();
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
}

void two(){  //функция для вывода двойки
    clear();
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(g, LOW);
    digitalWrite(e, LOW);
    digitalWrite(d, LOW);
}

void three(){  //функция для вывода тройки
    clear();
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(g, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
}
//создаем массив из функций, сами функции будут выводить нужную цифру
void (*functptr[])() = {zero, one, two, three}; //и да, в c++ такое можно делать ☺
void setup()
{  //в цикле настраиваем порты (так меньше кода) и выключаем сегмент
  for (int i = 2; i<=8; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, HIGH); //HIGH - гасим (если общий анод, т.е. питание)
  }                        //если будет с общей землей - LOW
}

void loop(){
  for (int i = 0; i<4; i++){ //в цикле будем вызывать по очереди все функции из массива
    (*functptr[i])();
     delay(1000);
  }
}
```

В результате запуска данного скетча на индикаторе будут последовательно появляться 0, 1, 2, 3. Соответствующим образом можно сделать и работу с другими цифрами, а также подключить еще один индикатор, но тогда понадобятся еще дополнительные пины на Arduino. Чтобы сократить кол-во пинов, можно воспользоваться другим вариантом подключения индикатора.

2. Через сдвиговый регистр (рис. 25).

Подключаем сегменты индикатора как обычные светодиоды к сдвиговому регистру.

При написании кода важно учитывать вид индикатора – с общим анодом или

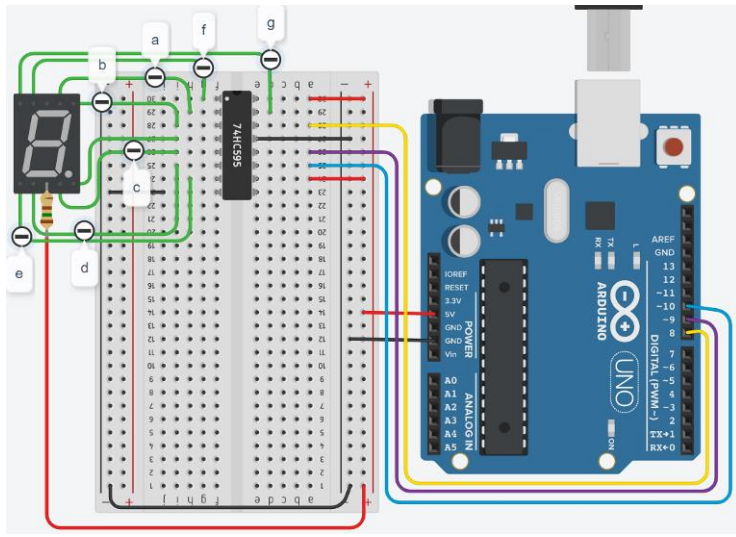


Рис. 25. Присоединение к Arduino семисегментного индикатора через сдвиговый регистр.

катодом, от этого будет зависеть передаваемый сигнал на включение нужного сегмента: 1 или 0.

Код для показа 0, 1 и 2:

```
int DS_pin = 8; //назначаем пины 8,9,10 для управления
int STCP_pin = 9; //сдвиговым регистром
int SHCP_pin = 10;
//массивы для передачи на индикатор
// 0 - сегмент вкл, 1 - выкл
// сегменты          g f a b P c d e    P - точка
boolean clear[8] =  {1,1,1,1,1,1,1,1}; //для очистки
boolean zero[8] =   {1,0,0,0,1,0,0,0}; //ноль
boolean one[8] =    {1,1,1,0,1,0,1,1}; //один
boolean two[8] =    {0,1,0,0,1,1,0,0}; //два

void writereg(boolean *toSend) //функция для передачи массива в сдвиговый регистр
{//она позволяет синхронизировать передачу данных в сдвиговый регистр
  digitalWrite(STCP_pin, LOW); //открываем сдвиговый регистр на прием данных
  for (int i = 7; i >= 0; i--)
  {
    digitalWrite(SHCP_pin, LOW); //открываем порт регистра для приема 1 значения
    digitalWrite(DS_pin, toSend[i] ); //посылаем i-й элемент в регистр
    digitalWrite(SHCP_pin, HIGH); //закрываем порт регистра для приема
  }
  digitalWrite(STCP_pin, HIGH); //закрываем сдвиговый регистр
}

void setup()
{
  pinMode(DS_pin, OUTPUT); //назначаем пины для управления сдвиговым регистром
  pinMode(STCP_pin, OUTPUT);
  pinMode(SHCP_pin, OUTPUT);
  writereg(clear);
}

void loop()
{
  writereg(clear);
  delay(1000);
  writereg(zero);
  delay(1000);
  writereg(one);
  delay(1000);
  writereg(two);
  delay(1000);
}
```

В результате выполнения данного скетча на индикаторе будут последовательно появляться 0, 1, 2. Главное преимущество данного варианта – легкость подключения второго индикатора.

Задания

(схема состоит из семисегментного индикатора и кнопки)

1. По нажатию кнопки начинается отсчет цифр от 0 до 9, после 9 останавливается.
2. После запуска скетча начинается отсчет цифр от 0 до 9, при нажатии кнопки он останавливается.
3. После запуска скетча начинается отсчет четных цифр (от 0 до 8, потом по кругу), при нажатии кнопки отчет переключается на нечетные цифры.
4. Первый раз нажимаем кнопку – начинается отсчет от 0 до 9 (по кругу), второй раз

нажимаем – в обратную сторону (тоже по кругу).

5. При нажатии на кнопку меняется цифра на индикаторе.
6. Пока держим кнопку – идет отсчет цифр (по кругу), отпускаем кнопку – отсчет останавливается.
7. Добавляем вторую кнопку. При нажатии первой кнопки цифры меняются от меньшей к большей, при нажатии второй – от большей к меньшей.

8. Использование ультразвукового датчика для определения расстояния до объекта

Ультразвуковые датчики (см. рис. 26) позволяют определить расстояние до объекта, это можно использовать, например, для выявления препятствий на пути робота. Такой датчик определяет расстояние до объектов точно так же, как это делают дельфины или летучие мыши. Он генерирует звуковые импульсы на частоте 40 кГц и слушает эхо. По времени распространения звуковой волны туда и обратно можно однозначно определить расстояние до объекта.

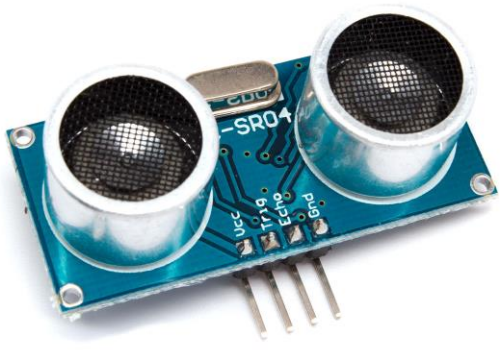


Рис. 26. Ультразвуковой датчик HC-SR04.

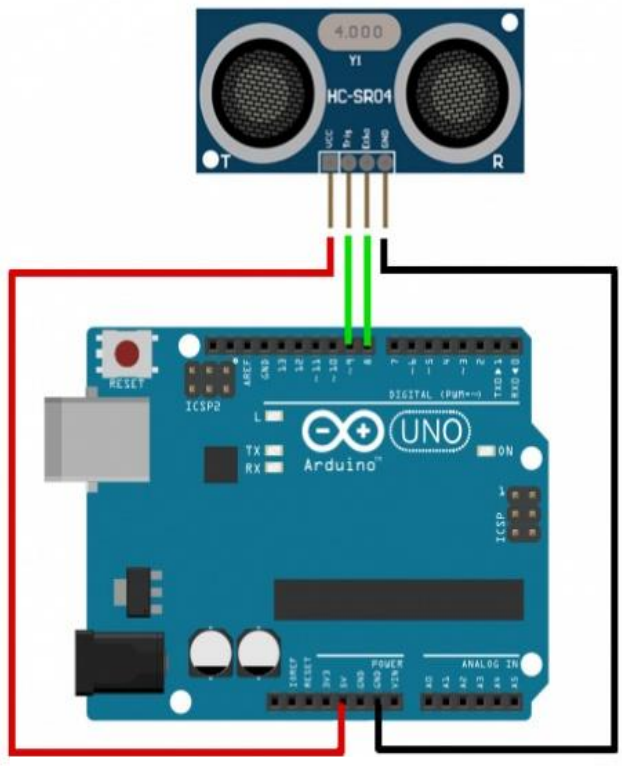


Рис. 27. Соединение HC-SR04 с Arduino.

У таких датчиков есть один нюанс – могут возникнуть трудности с определением

расстояния до пушистых или очень тонких предметов.

Разберем на примере датчика HC-SR04 как его использовать с Arduino.

Подключим его по схеме как на рис. 27: пин Vcc соединяем с портом 5V Arduino (это питание датчика), пин Gnd соединяем с портом Gnd Arduino (это земля), порты Trig и Echo соединяем соответственно с портами 9 и 8 Arduino.

Теперь можно писать скетч для Arduino:

```
#define Trig 9 //обозначаем порты, к которым подключили датчик
#define Echo 8

void setup()
{
  pinMode(Trig, OUTPUT); //назначаем пин на выход
  pinMode(Echo, INPUT);  //назначаем пин на вход
  Serial.begin(9600);
}

void loop()
{
  digitalWrite(Trig, HIGH);  //посылаем ультразвук
  delayMicroseconds(10); //ждем 10мс
  digitalWrite(Trig, LOW);  //отключаем ультразвук
  unsigned int impulse=pulseIn(Echo, HIGH);  //с помощью специальной функции
                    //принимаем сигнал ультразвука
  unsigned int distance=impulse/58;  //и по формуле вычисляем дистанцию
  Serial.println(distance); //выводим дистанцию до объекта
  delay(1000);  //пауза в 1 секунду
}
```

Загрузив скетч в Arduino и открыв монитор последовательного порта, можно увидеть появляющиеся раз в секунду значения расстояния до объекта. Можно поднести руку к датчику и менять расстояние до него, наблюдая изменения в показаниях

Можно использовать для измерения расстояния до объектов специальную библиотеку (https://bitbucket.org/teckel12/arduino-new-ping/downloads/) NewPing, упрощающую работу с датчиком. Архив с библиотекой нужно разархивировать в папку с Arduino IDE в подкаталог \libraries или добавить средствами Arduino IDE (Скетч –> Подключить библиотеку –> Добавить ZIP библиотеку). После этого, в самой среде Arduino IDE в меню Файл –> Примеры появится вкладка NewPing и сама библиотека станет доступна для использования в скетчах. Пример использования библиотеки:

```
#include <NewPing.h> //подключаем библиотеку

#define TRIGGER_PIN 9 //обозначаем порты, к которым подключили датчик
#define ECHO_PIN  8
#define MAX_DISTANCE 200 // максимальная дальность определния объекта (в см)
// обычно максимум где-то 400-500 см

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); //создаем объект sonar с
//нужными параметрами, этот объект и будет отвечать за измерения

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

```
}

void loop() {
  delay(50);  // ждем 50мс между замерами (можно делать 20 замеров в секунду).
              //минимальное ожидание между замерами не должно быть меньше 29мс
  Serial.print("Ping: "); //выводим в монитор порта текст
  Serial.print( sonar.ping_cm() ); // и  результаты замера в сантиметрах
                                   // (0 = вне зоны замера)
  Serial.println("cm");
}
```

Задания

1. Схема состоит из мотора и ультразвукового датчика. Чем ближе что-то к датчику, тем медленнее крутится мотор.
2. Схема состоит из 8 светодиодов, подключенных через сдвиговый регистр, и ультразвукового датчика. Чем ближе препятствие, тем больше светодиодов загорается.
3. Схема состоит из сервомотора и ультразвукового датчика. В зависимости от дальности препятствия меняется угол поворота серво.
4. Схема состоит из светодиода и ультразвукового датчика. Чем ближе препятствие к датчику, тем меньше горит светодиод.
5. Схема состоит из мотора и ультразвукового датчика. Чем ближе препятствие к датчику, тем быстрее крутится мотор.
6. Схема состоит из светодиода и ультразвукового датчика. Чем ближе препятствие к датчику, тем ярче горит светодиод.
7. Схема состоит из 8 светодиодов, подключенных через сдвиговый регистр, и ультразвукового датчика. Чем ближе препятствие, тем меньше светодиодов горит.

9. Связь Arduino по bluetooth с телефоном или компьютером

Модуль Bluetooth позволяет легко связать Arduino и компьютер или смартфон по беспроводному каналу. Попробуем связать Arduino и смартфон на базе ОС Android по беспроводному каналу на основе

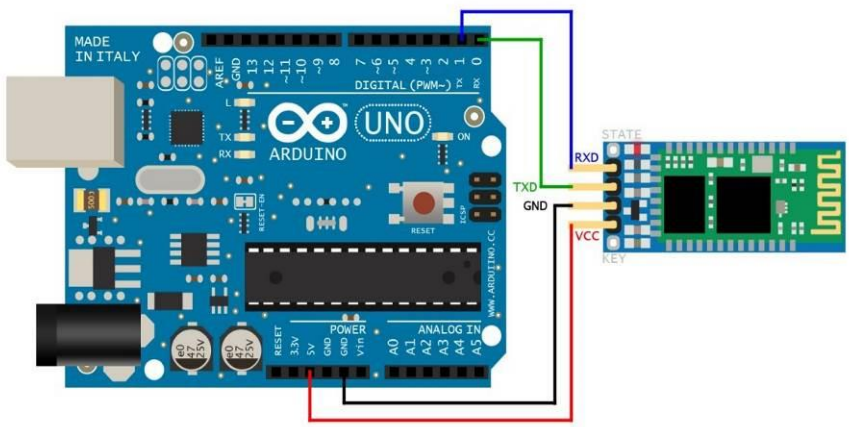


Рис. 28. Схема подключения модуля Bluetooth к Arduino.

Bluetooth. Сначала соединим модуль Bluetooth и плату Arduino как на рис. 28.

Выход с обозначением RXD соединяем с пином 1 (TX) Arduino, выход TXD - с пином 0 (RX). GND и VCC соединяем с пинами GND и 5V на Arduino соответственно. После подключения всех проводов модуль Bluetooth начнет мигать встроенным светодиодом.

Для проверки соединения можно использовать любую Android-программу, предоставляющую Bluetooth-терминал, например, Bluetooth Terminal (бесплатное приложение в Google Play). Если модуль Bluetooth подключен к Arduino, то можно на смартфоне или планшете запустить Bluetooth-терминал и в его меню выбрать Connect a device - Secure, выбрать устройство (HC-05 или HC-06, в зависимости от модели модуля Bluetooth), после этого на экране смартфона появится надпись Connected to device, а модуль Bluetooth, подсоединенный к Arduino, перестанет мигать светодиодом (светодиод будет просто гореть) - см. рис. 29.

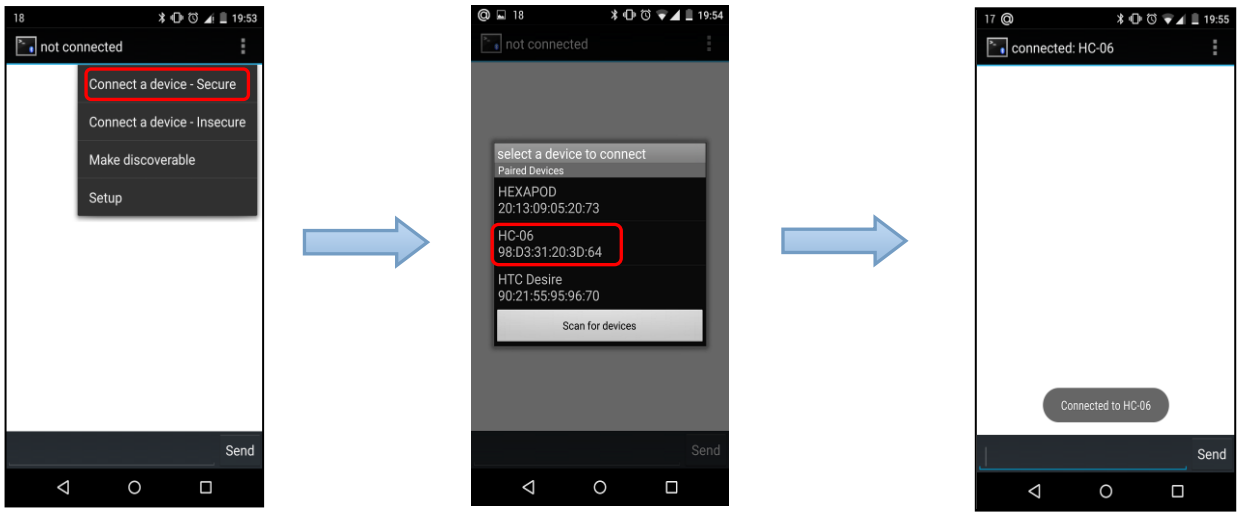


Рис. 29. Подключение Android-смартфона к Arduino через модуль Bluetooth.

Если на смартфоне не был включен Bluetooth, то программа спросит о его включении. У этой программы есть небольшая особенность - при изменении ориентации экрана (с горизонтальной на вертикальную и наоборот) теряется подключение к устройству и нужно снова выполнять подключение по рис. 29.

Теперь можно создать скетч для управления встроенным в Arduino светодиодом через Bluetooth-терминал:

```
char val; //переменная, в которую будем считывать значения из
        //последовательного порта
int LED = 13; //пин, к которому подключен встроенный светодиод
//модуль Bluetooth передает данные через обычный последовательный порт,
//с которым мы уже работали ранее, но использовали через Arduino IDE. Теперь
//будем напрямую считывать значения из последовательного порта, который
//будет принимать их через модуль Bluetooth
```

```
void setup()
{
  Serial.begin(9600); //открываем последовательный порт
  pinMode(LED, OUTPUT); //назначаем пин светодиода на вывод
}
void loop()
{
  if (Serial.available()) //если есть данные в последовательном порту
  {  //то считываем первое значение в переменную val
    val = Serial.read();
    if (val == 'W')  //если это значение буква W
    {
      digitalWrite(LED, HIGH); //то включаем светодиод
    }
    if ( val == 'S')  //если это буква S, то выключаем светодиод
    {
      digitalWrite(LED, LOW);
    }
  }
}
```

Некоторые модули Bluetooth препятствуют компиляции скетча, если у них подключены линии RXD и TXD, поэтому отсоединяем их перед компиляцией, а потом подсоединяем снова. После того, как скетч заработает, модуль Bluetooth будет подсоединен и смартфон соединится с платой Arduino, можно в Bluetooth-терминале на смартфоне посылать букву W (обязательно большую) для включения светодиода на Arduino, или букву S (обязательно большую) для выключения светодиода.

Для плат Arduino **Leonardo**, **Iskra Neo** и некоторых других пины rx и tx соотносятся не с Serial, а с **Serial1** (или Serial2 и др., см. спецификации плат в сети Интернет), поэтому вместо `Serial.available()` нужно писать `Serial1.available()` и т.д. для всех соответствующих команд. И также необходимо в функции setup добавить `Serial1.begin(9600)`.

Задание

Сделать для 2-х выполненных ранее заданий управление через Bluetooth с помощью bluetooth-терминала.