Муниципальное бюджетное общеобразовательное учреждение
Лицей с. Толбазы муниципального района Аургазинский район Республики Башкортостан

РАССМОТРЕНО
На заседании кафедры
Протокол № 1 от «30» 08 2017 г

СОГЛАСОВАНО
Зам.директора по УВР
/Л.Р.Чаньшева
30 08 2017

УТВЕРЖДАЮ
Директор Лицея Л.Д.Васильева
Приказ № 286 от «30» 08 2017 г.

Образовательная программа

дополнительного образования «Робототехника»

на 2017-2018 учебный год
количество часов в неделю - 1
Общее количество часов – 34
5-8 классы

Составитель: Фасиков Айдар Галиевич

Толбазы — 2017 год

Результаты освоения курса внеурочной деятельности.

Цель: обучение воспитанников основам робототехники, программирования. Развитие творческих способностей в процессе конструирования и проектирования.

Задачи:

Обучающие:

- дать первоначальные знания о конструкции робототехнических устройств;
- научить приемам сборки и программирования робототехнических устройств;
- сформировать общенаучные и технологические навыки конструирования и проектирования;
- ознакомить с правилами безопасной работы с инструментами

Воспитывающие:

- формировать творческое отношение к выполняемой работе;
- воспитывать умение работать в коллективе, эффективно распределять обязанности.

Развивающие:

- развивать творческую инициативу и самостоятельность;
- развивать психофизиологические качества учеников: память, внимание, способность логически мыслить, анализировать, концентрировать внимание на главном.
- развивать умения излагать мысли в четкой логической последовательности, отстаивать свою точку зрения, анализировать ситуацию и самостоятельно находить ответы на вопросы путем логических рассуждений.

Материальные ресурсы:

1. Набор MRT-2;
2. Программное обеспечение ArduinoUNO
3. Руководство пользователя «Платформа Arduino»
4. АРМ учителя (компьютер, проектор, сканер, принтер)
5. Образовательный конструктор «Мастер ARDUINO XXL»

Личностные, метапредметные и предметные результаты освоения курса:

Личностными результатами изучения является формирование следующих умений:

- самостоятельно и творчески реализовывать собственные замыслы.
- повышение своего образовательного уровня и уровня готовности к продолжению обучения с использованием ИКТ.
- навыки взаимо - и самооценки, навыки рефлексии;
- сформированность представлений о мире профессий, связанных с робототехникой, и требованиях, предъявляемых различными востребованными профессиями, такими как инженер-механик, конструктор, архитектор, программист, инженер-конструктор по робототехнике;

Предметные образовательные результаты:

- Определять, различать и называть детали конструктора,
- Способность реализовывать модели средствами вычислительной техники;
- конструировать по условиям, заданным взрослым, по образцу, по чертежу, по заданной схеме и самостоятельно строить схему.
- Владение основами разработки алгоритмов и составления программ управления роботом;
- Умение проводить настройку и отладку конструкции робота.

Метапредметными результатами изучения является формирование следующих универсальных учебных действий (УУД):

Познавательные УУД:

- ориентироваться в своей системе знаний: отличать новое от уже известного.
- перерабатывать полученную информацию: делать выводы в результате совместной работы всего класса, сравнивать и группировать предметы и их образы;
- умение устанавливать взаимосвязь знаний по разным учебным предметам (математике, физике, природоведения, биологии, анатомии, информатике, технологии и др.) для решения прикладных учебных задач по Робототехнике.

Регулятивные УУД:

- уметь работать по предложенным инструкциям.
- умение излагать мысли в четкой логической последовательности, отстаивать свою точку зрения, анализировать ситуацию и самостоятельно находить ответы на вопросы путем логических рассуждений.
- определять и формулировать цель деятельности на занятии с помощью учителя;

Коммуникативные УУД:

- уметь работать в паре и в коллективе; уметь рассказывать о постройке.
- уметь работать над проектом в команде, эффективно распределять обязанности.

По окончанию курса обучения учащиеся должны:

Знать:

- правила безопасной работы;
- основные компоненты конструкторов Матрёшка;
- конструктивные особенности различных моделей, сооружений и механизмов;
- компьютерную среду, включающую в себя графический язык программирования;
- виды подвижных и неподвижных соединений в конструкторе;
- основные приемы конструирования роботов;
- конструктивные особенности различных роботов;
- как передавать программы в RCX;
- порядок создания алгоритма программы, действия робототехнических средств;
- как использовать созданные программы;
- самостоятельно решать технические задачи в процессе конструирования роботов (планирование предстоящих действий, самоконтроль, применять полученные знания, приемы и опыт конструирования с использованием специальных элементов, и других объектов и т.д.);
- создавать реально действующие модели роботов при помощи специальных элементов по разработанной схеме;
- создавать программы на компьютере для различных роботов;
- корректировать программы при необходимости;

Уметь:

- принимать или намечать учебную задачу, ее конечную цель.
- проводить сборку робототехнических средств, с применением LEGO конструкторов;
- создавать программы для робототехнических средств.
- прогнозировать результаты работы.
- планировать ход выполнения задания.
- рационально выполнять задание.
- руководить работой группы или коллектива.
- высказываться устно в виде сообщения или доклада.
- высказываться устно в виде рецензии ответа товарища.
- представлять одну и ту же информацию различными способами

Содержание курса внеурочной деятельности с указанием форм организации и видов деятельности.

Название раздела	Формы организации	Виды деятельности
1. Электричество		
1.1. Электричество. Основные элементы схемы	Изучение нового материала	Наблюдать и описывать
1.2. Схемы.	Изучение нового материала	Уметь разбирать схемы.
1.3. Резистор.	Изучение нового материала	Уметь пользоваться резистором.
1.4. Делитель напряжения.	Изучение нового материала	Знать определение напряжения.
1.5. Диод.	Изучение нового материала	Уметь подключать диод.
1.6. Светодиод.	Изучение нового материала	Уметь подключать светодиод.
1.7. Кнопка.	Изучение нового материала	Уметь пользоваться кнопкой.
1.8. Транзисторы.	Изучение нового материала	Знать устройство транзисторов.
1.9. Конденсатор.	Изучение нового материала	Знать устройство конденсаторов.
1.10. Пьезодинамик.	Изучение нового материала	Уметь пользоваться пьеходинамиком.
1.11. Мотор.	Изучение нового материала	Уметь пользоваться и подключать мотор.
1.12. Светопривод.	Изучение нового материала	Уметь пользоваться и подключать светопривод.
1.13. Микросхема.	Изучение нового материала	Знать устройство микросхемы.
2. Начало работы с Arduino		
2.1. Начало работы с Arduino.	Изучение нового материала	Знать основные понятия.
2.2. Маячок.	Практическое занятие	Уметь собирать и объяснять способ работы.
2.3. Маячок с нарастающей яркостью.	Практическое занятие	Уметь собирать и объяснять способ работы.
2.4. Светильник с управляемой яркостью.	Практическое занятие	Уметь собирать и объяснять способ работы.

2.5. Ночной светильник.	Практическое занятие	Уметь собирать и объяснять способ работы.
2.6. Пульсар.	Практическое занятие	Уметь собирать и объяснять способ работы.
2.7. Бегущий огонек.	Практическое занятие	Уметь собирать и объяснять способ работы.
2.8. Миксер.	Практическое занятие	Уметь собирать и объяснять способ работы.
2.9. Кнопочный переключатель.	Практическое занятие	Уметь собирать и объяснять способ работы.
2.10. Светильник с кнопочным управлением.	Практическое занятие	Уметь собирать и объяснять способ работы.
2.11. Секундомер.	Практическое занятие	Уметь собирать и объяснять способ работы.
2.12. Счетчик нажатий.	Практическое занятие	Уметь собирать и объяснять способ работы.
2.13. Комнатный термометр.	Практическое занятие	Уметь собирать и объяснять способ работы.
2.14. Метеостанция.	Практическое занятие	Уметь собирать и объяснять способ работы.
Итоговое занятие.	Обобщение	

Тематическое планирование.

№ п\п	Тема занятия	Количество часов	Дата проведения	
			План	Факт
Раздел 1. Электричество				
1.	Электричество. Основные элементы схемы. Техника безопасности при работе .	1	7.09	
2.	Схемы. Сборка схем.	1	14.09	
3.	Управление электричеством. Резистор. Делитель напряжения.	1	21.09	
4.	Диод. Светодиод. Кнопка.	1	28.09	
5.	Транзисторы. Конденсатор. Пьезодинамик. Мотор.	1	12.10	
6.	Светопривод. Микросхема.		19.10	
Раздел 2. Начало работы с Arduino				
7.	Начало работы с Arduino	1	26.10	
8.	Маячок	1	9.11	
9.	Маячок с нарастающей яркостью	1	16.11	
10.	Светильник с управляемой яркостью	1	23.11	
11.	Терменвокс	2	30.11, 7.12	
12.	Ночной светильник	2	14.12, 21.12	
13.	Пульсар	1	28.12	
14.	Бегущий огонек	1	18.01	

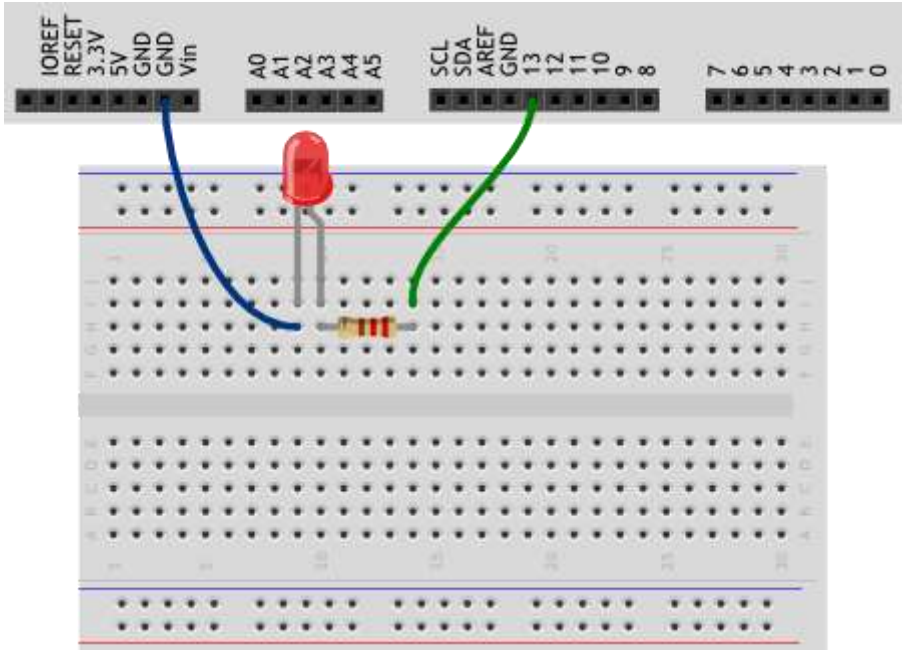
15.	Пианино	2	25.01, 1.02	
16.	Миксер	2	8.02, 15.02	
17.	Кнопочный переключатель	1	22.02	
18.	Светильник с кнопочным управлением	1	1.03	
19.	Кнопочные ковбои	2	15.03, 22.03	
20.	Секундомер	1	5.04	
21.	Счетчик нажатий	1	12.04	
22.	Комнатный термометр	2	19.04, 26.04	
23.	Метеостанция	2	3.05, 10.05	
24.	Тестер батареек	1	17.05	
25.	Светильник, управляемый по USB	1	24.05	
26.	Итоговое занятие	1	31.05	
Итого		34 часа		

Эксперимент 1. Маячок

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 светодиод
1 резистор номиналом 220 Ом
2 провода «папа-папа»

Схема на макетке



Скетч

p010_blink.ino

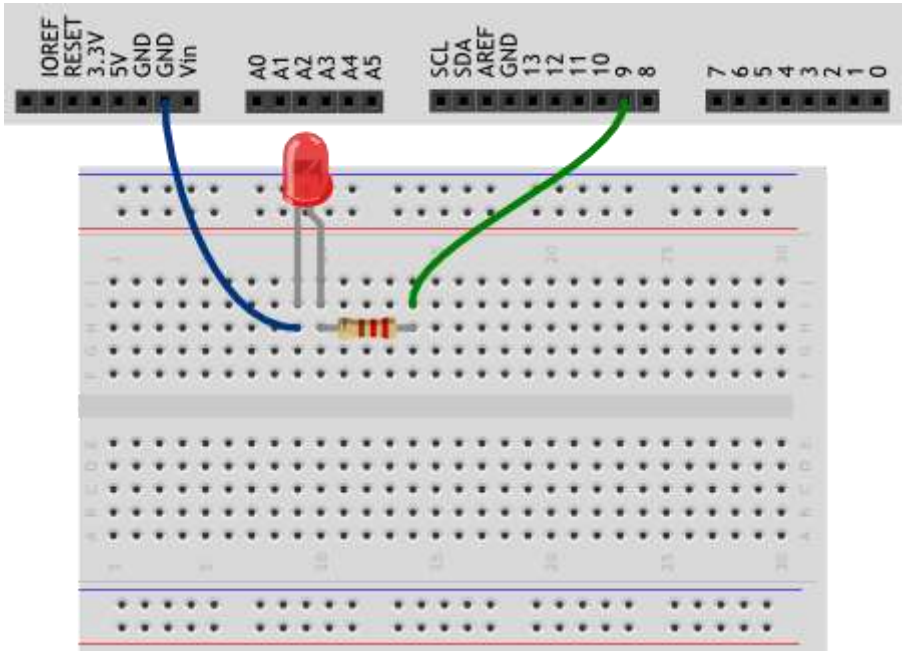
```
void setup()
{
// настраиваем пин №13 в режим выхода,
// т.е. в режим источника напряжения
pinMode(13, OUTPUT);
}
 void loop()
{
// подаём на пин 13 «высокий сигнал» (англ. «high»), т.е.
// выдаём 5 вольт. Через светодиод побежит ток.
// Это заставит его светиться
digitalWrite(13, HIGH);

// задерживаем (англ. «delay») микроконтроллер в этом
// состоянии на 100 миллисекунд
delay(100);

// подаём на пин 13 «низкий сигнал» (англ. «low»), т.е.
// выдаём 0 вольт или, точнее, приравниваем пин 13 к земле.
// В результате светодиод погаснет
digitalWrite(13, LOW);

// замираем в этом состоянии на 900 миллисекунд
delay(900);

// после «размораживания» loop сразу же начнёт исполняться
// вновь, и со стороны это будет выглядеть так, будто
// светодиод мигает раз в 100 мс + 900 мс = 1000 мс = 1 сек
}
```

Эксперимент 2. Маячок с нарастающей яркостью

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 светодиод
1 резистор номиналом 220 Ом
2 провода «папа-папа»

Схема на макетке



p020_pulse_light.ino

```
// даём разумное имя для пина №9 со светодиодом
// (англ. Light Emitting Diode илипросто «LED»)
// Так нам не нужно постоянно вспоминать куда он подключён
#define LED_PIN 9
 void setup()
{
  // настраиваем пин со светодиодом в режим выхода,
// какираньше
pinMode(LED_PIN, OUTPUT);
}
void loop()
{
  // выдаём неполное напряжение на светодиод
  // (он же ШИМ-сигнал, он же PWM-сигнал).
  // Микроконтроллер переводит число от 0 до 255 к напряжению
  // от 0 до 5 В. Например, 85 — это 1/3 от 255,
// т.е. 1/3 от 5 В, т.е. 1,66В.
analogWrite(LED_PIN, 85);
// держим такую яркость 250 миллисекунд
delay(250);

  // выдаём 170, т.е. 2/3 от 255, или иными словами — 3,33 В.
  // Больше напряжение — выше яркость!
analogWrite(LED_PIN, 170);
delay(250);
   // все 5 В — полный накал!
analogWrite(LED_PIN, 255);
  // ждём ещё немного перед тем, как начать всё заново
delay(250);
}
```

Эксперимент 3. Светильник с управляемой яркостью

Список деталей для эксперимента

1 плата ArduinoUno

1 беспаечная макетная плата

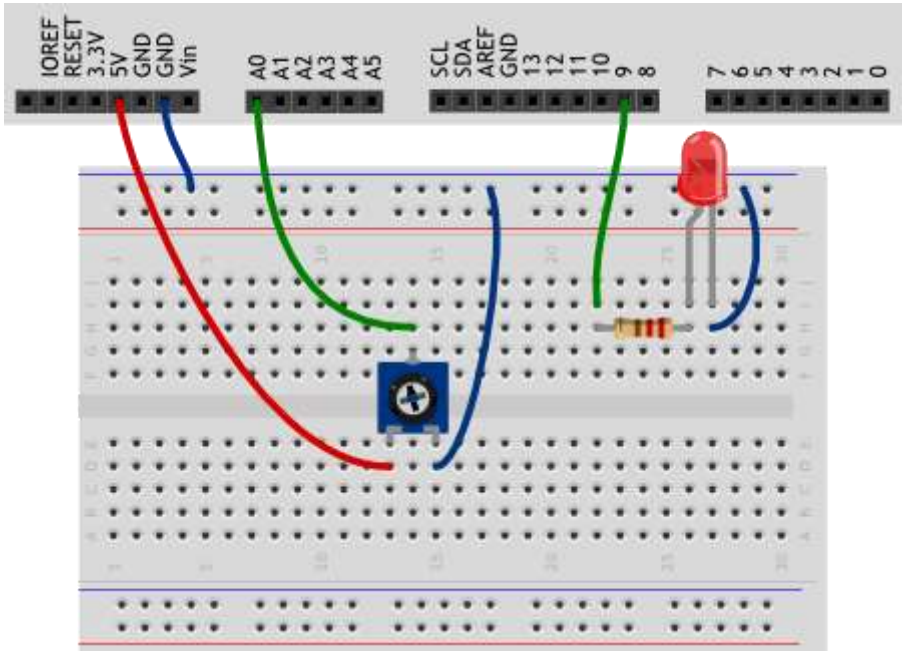
1 светодиод

1 резистор номиналом 220 Ом

6 проводов «папа-папа»

1 потенциометр

Схема на макетке




```
// даём разумные имена для пинов со светодиодом
// и потенциометром (англpotentiometer или просто «pot»)
#define LED_PIN     9
#define POT_PIN     A0

voidsetup()
{
// пин со светодиодом — выход, как и раньше...
pinMode(LED_PIN, OUTPUT);

// ...а вот пин с потенциометром должен быть входом
// (англ. «input»): мы хотим считывать напряжение,
// выдаваемоеим
pinMode(POT_PIN, INPUT);
}

void loop()
{
// заявляем, что далее мы будем использовать 2 переменные с
// именами rotation и brightness, и что хранить в них будем
// целые числа (англ. «integer», сокращённо просто «int»)
introtation, brightness;

// считываем в rotation напряжение с потенциометра:
// микроконтроллер выдаст число от 0 до 1023
// пропорциональное углу поворота ручки
rotation=analogRead(POT_PIN);

// в brightness записываем полученное ранее значение rotation
// делённое на 4. Поскольку в переменных мы пожелали хранить
// целые значения, дробная часть от деления будет отброшена.
// В итоге мы получим целое число от 0 до 255
brightness=rotation/4;

// выдаём результат на светодиод
analogWrite(LED_PIN, brightness);
}
```

Эксперимент 4. Терменвокс

Список деталей для эксперимента

1 плата ArduinoUno

1 беспаечная макетная плата

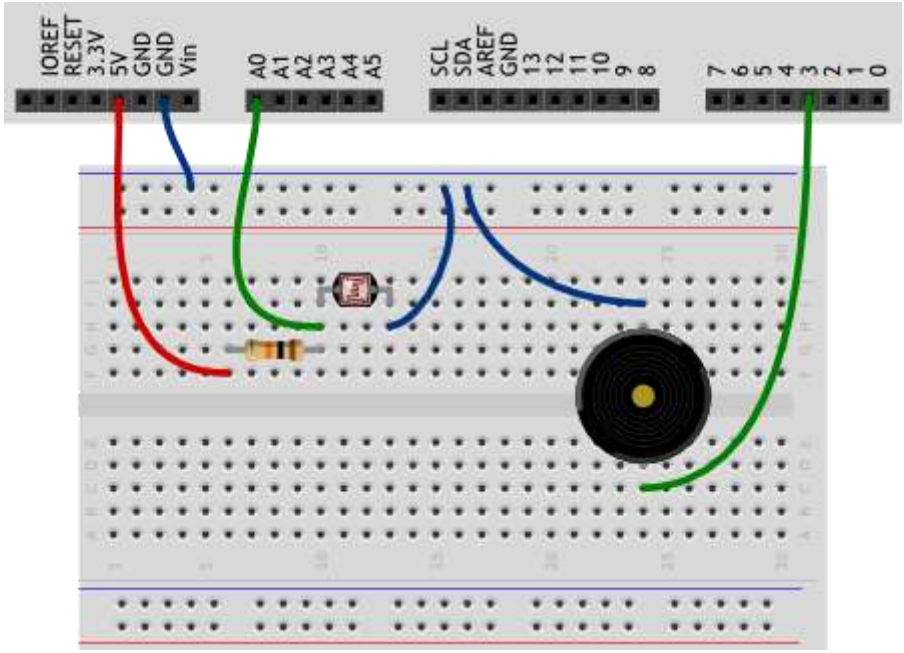
1 пьезопищалка

6 проводов «папа-папа»

1 резистор номиналом 10 кОм

1 фоторезистор

Схема на макетке



Скетч

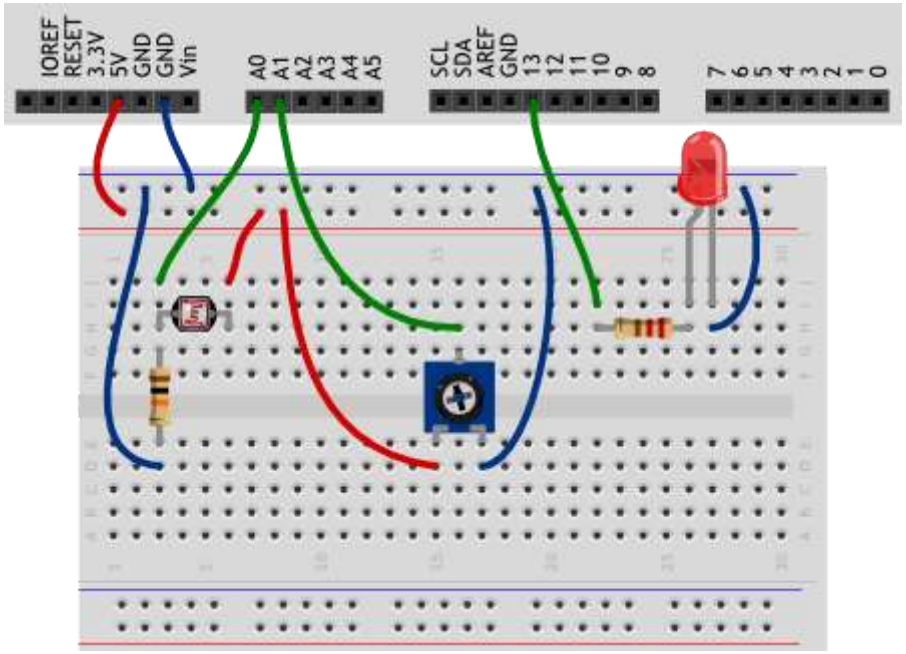
```
// даём имена для пинов с пьезопищалкой (англ. buzzer) и фото-
// резистором (англ. Light Dependent Resistor илипросто LDR)
#define BUZZER_PIN  3
#define LDR_PIN     A0

voidsetup()
{
// пин с пьезопищалкой — выход...
pinMode(BUZZER_PIN, OUTPUT);

// ...а все остальные пины являются входами изначально,
// всякий раз при подаче питания или сбросе микроконтроллера.
// Поэтому, на самом деле, нам совершенно необязательно
// настраивать LDR_PIN в режим входа: он и так им является
}

void loop()
{
intval, frequency;

// считываем уровень освещённости так же, как для
// потенциометра: в виде значения от 0 до 1023.
val=analogRead(LDR_PIN);

// рассчитываем частоту звучания пищалки в герцах (ноту),
// используя функцию проекции (англ. map). Она отображает
// значение из одного диапазона на другой, строя пропорцию.
// В нашем случае [0; 1023] -> [3500; 4500]. Так мы получим
// частоту от 3,5 до 4,5 кГц.
frequency=map(val, 0, 1023, 3500, 4500);

// заставляем пин с пищалкой «вибрировать», т.е. звучать
// (англ. tone) на заданной частоте 20 миллисекунд. При
// следующих проходах loop, tone будет вызван снова и снова,
// и на деле мы услышим непрерывный звук тональностью, которая
// зависит от количества света, попадающего на фоторезистор
tone(BUZZER_PIN, frequency, 20);
}
```

Эксперимент 5. Ночной светильник

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 светодиод
1 фоторезистор
1 резистор номиналом 220 Ом
1 резистор номиналом 10 кОм
1 переменный резистор (потенциометр)
10 проводов «папа-папа»

Схема на макетке



Скетч

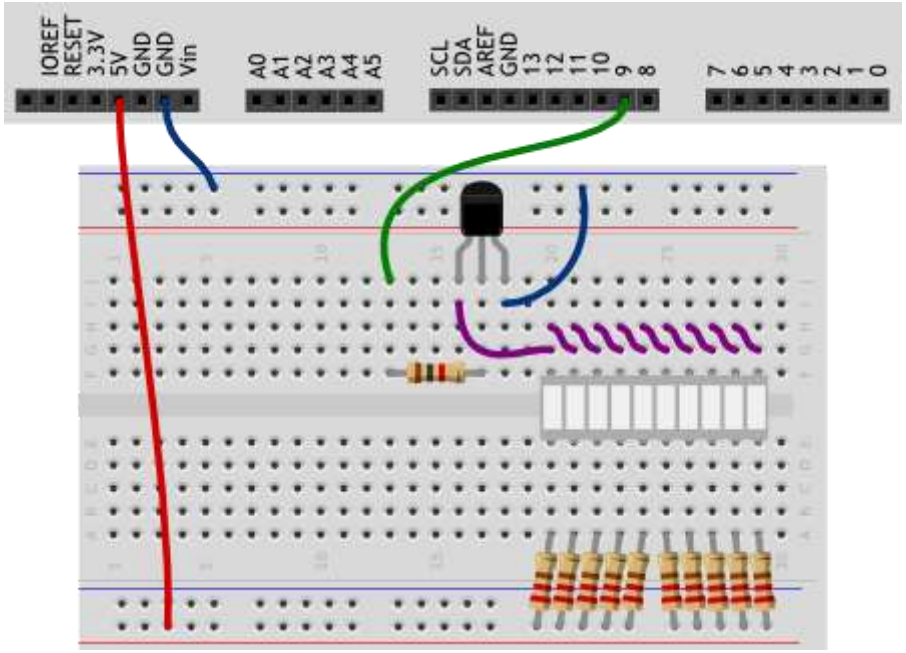
```
#define LED_PIN  13
#define LDR_PIN  A0
#define POT_PIN  A1

void setup()
{
pinMode(LED_PIN, OUTPUT);
}

void loop()
{
// считываем уровень освещённости. Кстати, объявлять
// переменную и присваивать ей значение можно разом
int lightness =analogRead(LDR_PIN);

// считываем значение с потенциометра, которым мы регулируем
// пороговое значение между условными темнотой и светом
int threshold =analogRead(POT_PIN);

// объявляем логическую переменную и назначаем ей значение
// «темно ли сейчас». Логические переменные, в отличие от
// целочисленных, могут содержать лишь одно из двух значений:
// истину (англ. true) или ложь (англ. false). Такие значения
// ещё называют булевыми (англ. boolean).
booleantooDark=(lightness<threshold);

// используем ветвление программы: процессор исполнит один из
// двух блоков кода в зависимости от исполнения условия.
// Если (англ. «if») слишком темно...
if(tooDark){
// ...включаемосвещение
digitalWrite(LED_PIN, HIGH);
}else{
// ...иначе свет не нужен — выключаем его
digitalWrite(LED_PIN, LOW);
}
}
```

Эксперимент 6. Пульсар

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 биполярный транзистор
1 светодиодная шкала
1 резистор номиналом 1 кОм
10 резисторов номиналом 220 Ом
13 проводов «папа-папа»

Схема на макетке



Скетч

p060_pulse_bar.ino

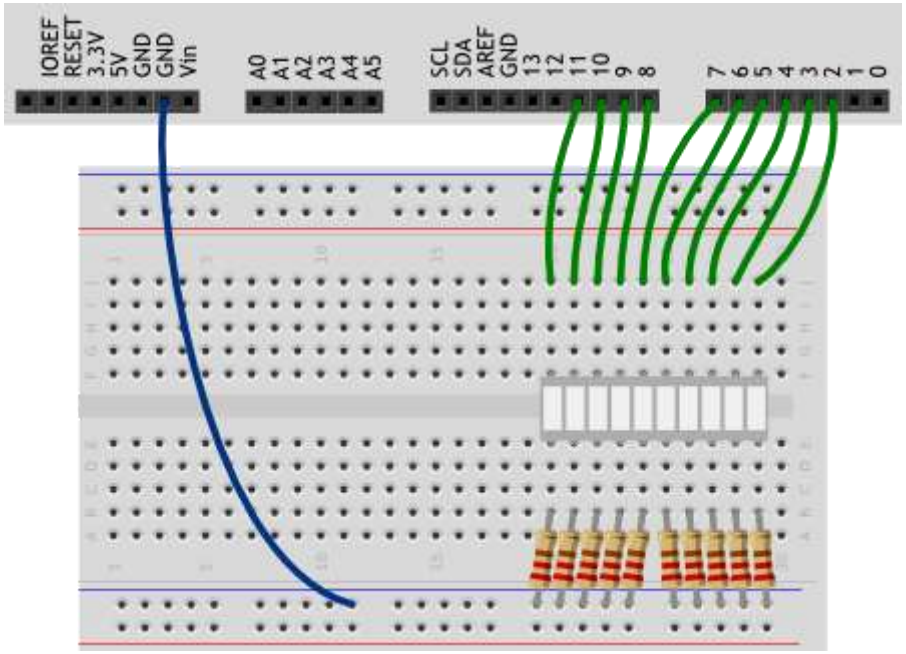
```
#define CONTROL_PIN  9

// переменные верхнего уровня, т.е. объявленные вне функций,
// называют глобальными. Их значения сохраняются всё время,
// пока работает микроконтроллер
intbrightness=0;

void setup()
{
pinMode(CONTROL_PIN, OUTPUT);
}

voidloop()
{
// увеличиваем значение яркости на единицу, чтобы нарастить
// яркость. Однако яркость не должна быть более 255, поэтому
// используем операцию остатка от деления, чтобы при
// достижении значения 255, следующим значением снова стал 0
// Y % X — это остаток от деления Y на X;
// плюс, минус, делить, умножить, скобки — как в алгебре.
brightness=(brightness+1)%256;

// подаём вычисленный ШИМ-сигнал яркости на пин с базой
// управляющеготранзистора
analogWrite(CONTROL_PIN, brightness);

// ждём 10 мс перед следующим наращиванием яркости. Таким
// образом, полный накал будет происходить в течение
// 256×10 = 2560 мс
delay(10);
}
```

Эксперимент 7. Бегущий огонёк

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 светодиодная шкала
10 резисторов номиналом 220 Ом
11 проводов «папа-папа»

Схема на макетке



Скетч

```
// светодиодная шкала подключена к группе пиноврасположенных
// подряд. Даём понятные имена первому и последнему пинам
#define FIRST_LED_PIN  2
#define LAST_LED_PIN   11

voidsetup()
{
// в шкале 10 светодиодов. Мы бы могли написать pinMode 10
// раз: для каждого из пинов, но это бы раздуло код и
// сделало его изменение более проблематичным.
// Поэтому лучше воспользоваться циклом. Мы выполняем
// pinMode для (англ. for) каждого пина (переменная pin)
// от первого (= FIRST_LED_PIN) до последнего включительно
// (<= LAST_LED_PIN), всякий раз продвигаясь к следующему
// (++pin увеличивает значение pin на единицу)
// Так все пины от 2-го по 11-й друг за другом станут выходами
for(int pin = FIRST_LED_PIN; pin <= LAST_LED_PIN;++pin)
pinMode(pin, OUTPUT);
}

void loop()
{
// получаем время в миллисекундах, прошедшее с момента
// включения микроконтроллера
unsignedintms=millis();
// нехитрой арифметикой вычисляем, какой светодиод
// должен гореть именно сейчас. Смена будет происходить
// каждые 120 миллисекунд. Y % X — это остаток от
// деления Y на X; плюс, минус, скобки — как в алгебре.
int pin = FIRST_LED_PIN +(ms/120)%10;
// включаем нужный светодиод на 10 миллисекунд, затем —
// выключаем. На следующем проходе цикла он снова включится,
// если гореть его черёд, и мы вообще не заметим отключения
digitalWrite(pin, HIGH);
  delay(10);
digitalWrite(pin, LOW);
}
```

Эксперимент 8. Мерзкое пианино

Список деталей для эксперимента

1 плата ArduinoUno

1 беспаечная макетная плата

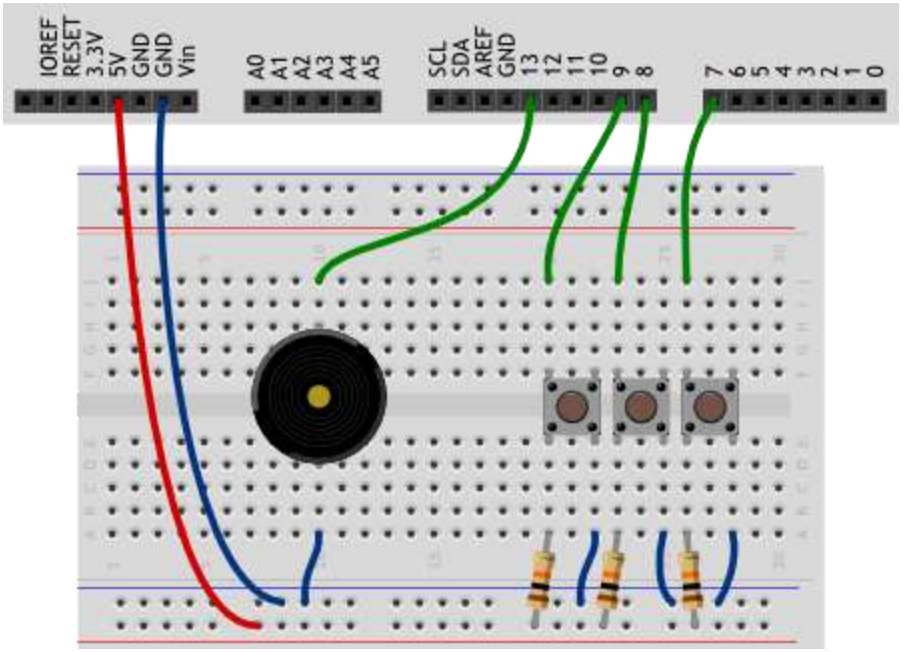
1 пьезопищалка

3 тактовых кнопки

3 резистора номиналом 10 кОм

10 проводов

Схема на макетке



Скетч

```
#define BUZZER_PIN    13 // пин с пищалкой (англ. «buzzer»)
#define FIRST_KEY_PIN 7  // первый пин с клавишей (англ. «key»)
#define KEY_COUNT     3  // общее количество клавиш

voidsetup()
{
pinMode(BUZZER_PIN, OUTPUT);
}

voidloop()
{
  // в цикле бежим по всем номерам кнопок от 0-го по 2-й
for (int i = 0; i < KEY_COUNT; ++i) {
   // на основе номера кнопки вычисляем номер её пина
intkeyPin = i + FIRST_KEY_PIN;

   // считываем значение с кнопки. Возможны всего 2 варианта:
   //  * высокий сигнал, 5 вольт, истина — кнопка отпущена
   //  * низкий сигнал, земля, ложь — кнопка зажата
booleankeyUp = digitalRead(keyPin);

   // проверяем условие «если не кнопка отпущена». Знак «!»
   // перед булевой переменной означает отрицание, т.е. «не».
if (!keyUp) {
     // рассчитываем высоту ноты в герцах в зависимости от
     // клавиши, которую рассматриваем на данном этапе цикла.
     // Мы получим значение 3500, 4000 или 4500
intfrequency = 3500 + i * 500;

     // Заставляем пищалку пищать с нужной частотой в течение
     // 20 миллисекунд. Если клавиша останется зажатой, пищалка
     // вновь зазвучит при следующем проходе loop, а мы услышим
     // непрерывный звук
tone(BUZZER_PIN, frequency, 20);
```

Эксперимент 9. Миксер

Список деталей для эксперимента

1 плата ArduinoUno

беспаечная макетная плата

3 тактовых кнопки

1 коллекторный двигатель

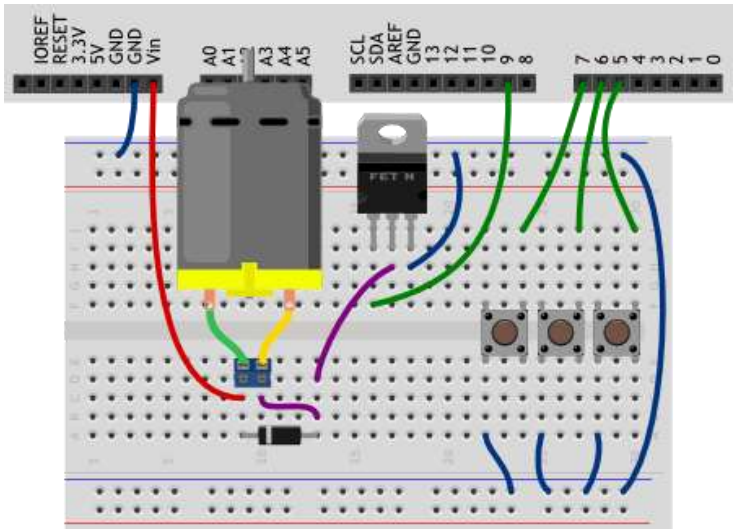
1 выпрямительный диод

1 полевой MOSFET-транзистор

15 проводов

1 клеммник, если вы используете мотор с проводами, которые плохо втыкаются в макетку

Схема на макетке



Скетч

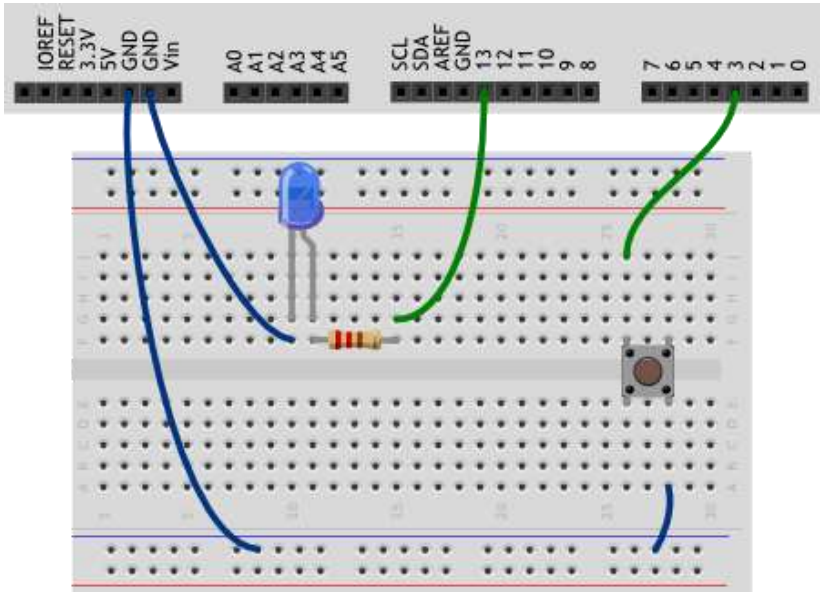
```
#define MOTOR_PIN       9
#define FIRST_BUTTON_PIN 5
#define BUTTON_COUNT     3
// имена можно давать не только числам, но и целым выражениям.
// Мы определяем с каким шагом (англ. step) нужно менять
// скорость (англ. speed) мотора при нажатии очередной кнопки
#define SPEED_STEP  (255 / (BUTTON_COUNT - 1))

voidsetup()
{
pinMode(MOTOR_PIN, OUTPUT);
  // на самом деле, в каждом пине уже есть подтягивающий
  // резистор. Для его включения необходимо явно настроить пин
  // как вход с подтяжкой (англ. inputwithpullup)
for (int i = 0; i < BUTTON_COUNT; ++i)
pinMode(i + FIRST_BUTTON_PIN, INPUT_PULLUP);
}
 voidloop()
{
for (int i = 0; i < BUTTON_COUNT; ++i) {
    // если кнопка отпущена, нам она не интересна. Пропускаем
    // оставшуюся часть цикла for, продолжая (англ. continue)
    // его дальше, для следующего значения i
if (digitalRead(i + FIRST_BUTTON_PIN))
continue;

    // кнопка нажата — выставляем соответствующую ей скорость
    // мотора. Нулевая кнопка остановит вращение, первая
    // заставит крутиться в полсилы, вторая — на полную
intspeed = i * SPEED_STEP;

    // подача ШИМ-сигнала на мотор заставит его крутиться с
    // указанной скоростью: 0 — стоп машина, 127 — полсилы,
    // 255 — полный вперёд!
analogWrite(MOTOR_PIN, speed);
  }
}
```

Эксперимент 10. Кнопочный переключатель

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 тактовая кнопка
1 резистор номиналом 220 Ом
1 светодиод
5 проводов

Схема на макетке



Скетч

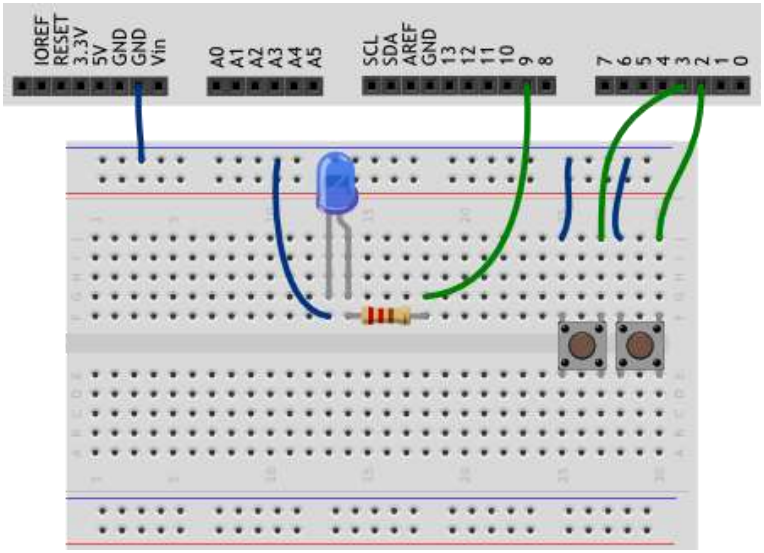
```
#define BUTTON_PIN  3
#define LED_PIN     13

booleanbuttonWasUp = true;  // была ли кнопка отпущена?
booleanledEnabled = false;  // включен ли свет?

voidsetup()
{
pinMode(LED_PIN, OUTPUT);
pinMode(BUTTON_PIN, INPUT_PULLUP);
}

voidloop()
{
  // определить момент «клика» несколько сложнее, чем факт того,
  // что кнопка сейчас просто нажата. Для определения клика мы
  // сначала понимаем, отпущена ли кнопка прямо сейчас...
booleanbuttonIsUp = digitalRead(BUTTON_PIN);

  // ...если «кнопка была отпущена и (&&) не отпущена сейчас»...
if (buttonWasUp&& !buttonIsUp) {
    // ...может это «клик», а может и ложный сигнал (дребезг),
// возникающий в момент замыкания/размыкания пластин кнопки,
    // поэтому даём кнопке полностью «успокоиться»...
delay(10);
    // ...и считываем сигнал снова
buttonIsUp = digitalRead(BUTTON_PIN);
if (!buttonIsUp) {  // если она всё ещё нажата...
      // ...это клик! Переворачиваем сигнал светодиода
ledEnabled = !ledEnabled;
digitalWrite(LED_PIN, ledEnabled);
    }
  }

  // запоминаем последнее состояние кнопки для новой итерации
buttonWasUp = buttonIsUp;
}
```

Эксперимент 11. Светильник с кнопочным управлением

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
2 тактовых кнопки
1 резистор номиналом 220 Ом
1 светодиод
7 проводов

Схема на макетке



Скетч

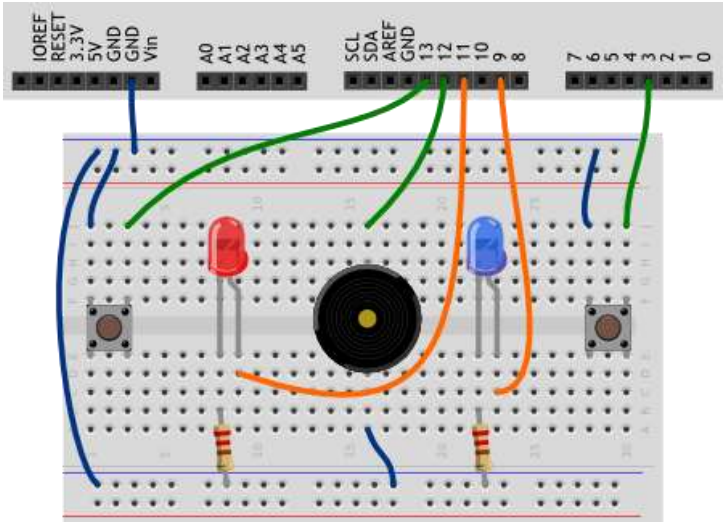
```
#define PLUS_BUTTON_PIN     2
#define MINUS_BUTTON_PIN    3
#define LED_PIN             9

intbrightness = 100;
booleanplusUp = true;
booleanminusUp = true;

voidsetup()
{
pinMode(LED_PIN, OUTPUT);
pinMode(PLUS_BUTTON_PIN, INPUT_PULLUP);
pinMode(MINUS_BUTTON_PIN, INPUT_PULLUP);
}
 voidloop()
{
analogWrite(LED_PIN, brightness);
  // реагируем на нажатия с помощью функции, написанной нами
plusUp = handleClick(PLUS_BUTTON_PIN, plusUp, +35);
minusUp = handleClick(MINUS_BUTTON_PIN, minusUp, -35);
}
// Собственная функция с 3 параметрами: номером пина с кнопкой
// (buttonPin), состоянием до проверки (wasUp) и градацией
// яркости при клике на кнопку (delta). Функция возвращает
// (англ. return) обратно новое, текущее состояние кнопки
booleanhandleClick(intbuttonPin, booleanwasUp, intdelta)
{
booleanisUp = digitalRead(buttonPin);
if (wasUp&& !isUp) {
delay(10);
isUp = digitalRead(buttonPin);
    // если был клик, меняем яркость в пределах от 0 до 255
if (!isUp)
brightness = constrain(brightness + delta, 0, 255);
  }
returnisUp; // возвращаем значение обратно, в вызывающий код
}
```

Эксперимент 12. Кнопочные ковбои

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
2 тактовых кнопки
2 резистора номиналом 220 Ом
2 светодиода
1 пьезопищалка
10 проводов

Схема на макетке



Скетч

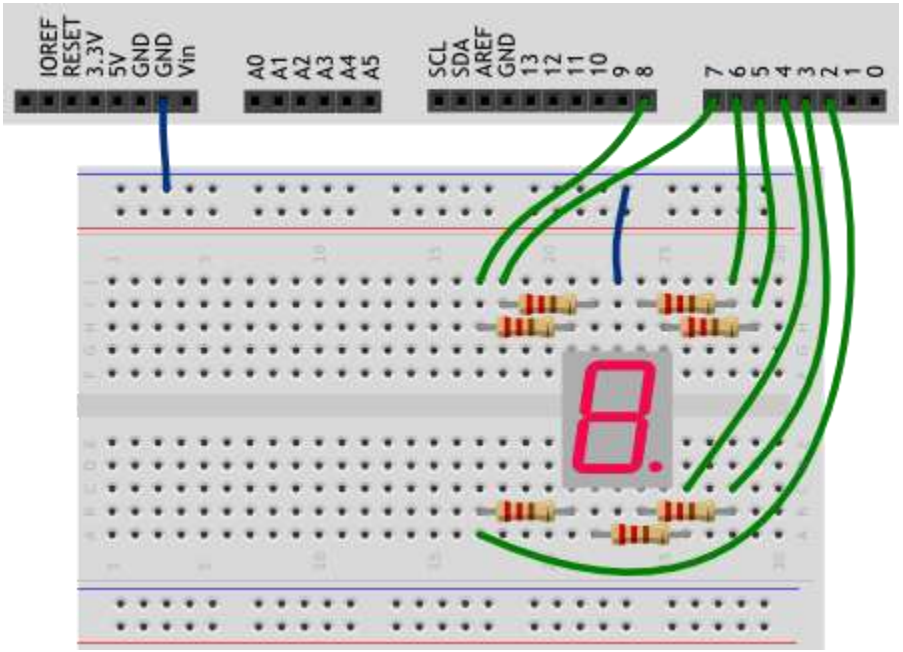
```
#define BUZZER_PIN   12  // пин с пищалкой
#define PLAYER_COUNT 2   // количество игроков-ковбоев
// вместо перечисления всех пиновпо-одному, мы объявляем пару
// списков: один с номерами пинов с кнопками, другой — со
// светодиодами. Списки также называют массивами (англ. array)
intbuttonPins[PLAYER_COUNT] = {3, 13};
intledPins[PLAYER_COUNT] = {9, 11};

voidsetup()
{
pinMode(BUZZER_PIN, OUTPUT);
for (intplayer = 0; player< PLAYER_COUNT; ++player) {
   // при помощи квадратных скобок получают значение в массиве
   // под указанным в них номером. Нумерация начинается с нуля
pinMode(ledPins[player], OUTPUT);
pinMode(buttonPins[player], INPUT_PULLUP);
  }
}

voidloop()
{
  // даём сигнал «пли!», выждав случайное время от 2 до 7 сек
delay(random(2000, 7000));
tone(BUZZER_PIN, 3000, 250); // 3 килогерца, 250 миллисекунд

for (intplayer = 0; ;player = (player+1) % PLAYER_COUNT) {
   // если игрок номер «player» нажал кнопку...
if (!digitalRead(buttonPins[player])) {
     // ...включаем его светодиод и сигнал победы на 1 сек
digitalWrite(ledPins[player], HIGH);
tone(BUZZER_PIN, 4000, 1000);
delay(1000);
digitalWrite(ledPins[player], LOW);
break; // Есть победитель! Выходим (англ. break) из цикла
   }
  }
}
```

Эксперимент 13. Секундомер

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 семисегментный индикатор
7 резисторов номиналом 220 Ом
9 проводов

Схема на макетке



Скетч

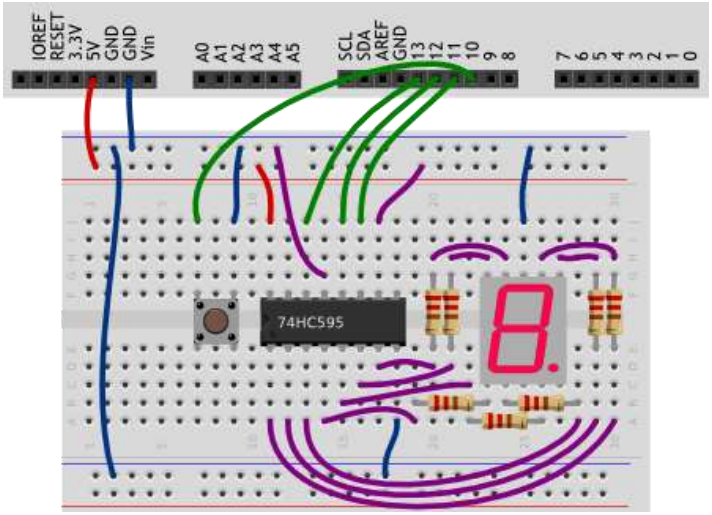
```
#define FIRST_SEGMENT_PIN  2
#define SEGMENT_COUNT      7

// префикс «0b» означает, что целое число за ним записано в
// в двоичном коде. Единицами мы обозначим номера сегментов
// индикатора, которые должны быть включены для отображения
// арабской цифры. Всего цифр 10, поэтому в массиве 10 чисел.
// Нам достаточно всего байта (англ. byte, 8 бит) для хранения
// комбинации сегментов для каждой из цифр.
bytenumberSegments[10] = {
  0b00111111, 0b00001010, 0b01011101, 0b01011110, 0b01101010,
  0b01110110, 0b01110111, 0b00011010, 0b01111111, 0b01111110,
};

voidsetup()
{
for (int i = 0; i < SEGMENT_COUNT; ++i)
pinMode(i + FIRST_SEGMENT_PIN, OUTPUT);
}

voidloop()
{
  // определяем число, которое собираемся отображать. Пусть им
  // будет номер текущей секунды, зацикленный на десятке
intnumber = (millis() / 1000) % 10;
  // получаем код, в котором зашифрована арабская цифра
intmask = numberSegments[number];
  // для каждого из 7 сегментов индикатора...
for (int i = 0; i < SEGMENT_COUNT; ++i) {
    // ...определяем: должен ли он быть включён. Для этого
    // считываем бит (англ. readbit), соответствующий текущему
    // сегменту «i». Истина — он установлен (1), ложь — нет (0)
booleanenableSegment = bitRead(mask, i);
    // включаем/выключаем сегмент на основе полученного значения
digitalWrite(i + FIRST_SEGMENT_PIN, enableSegment);
  }
}
```

Эксперимент 14. Счётчик нажатий

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 тактовая кнопка
1 выходной сдвиговый регистр 74HC595
1 семисегментный индикатор
7 резисторов номиналом 220 Ом
24 провода

Схема на макетке



Скетч

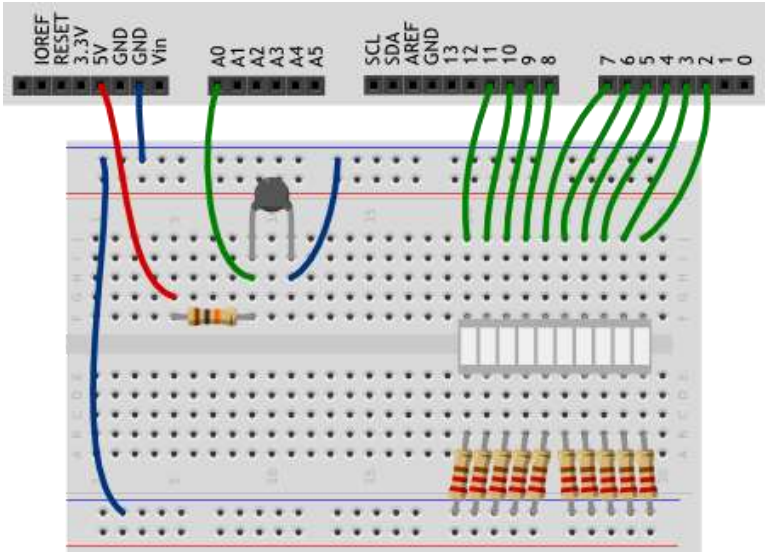
```
#define DATA_PIN    13 // пин данных (англ. data)
#define LATCH_PIN   12 // пин строба (англ. latch)
#define CLOCK_PIN   11 // пин такта (англ. clock)
#define BUTTON_PIN  10

intclicks = 0;
booleanbuttonWasUp = true;
bytesegments[10] = {
  0b01111101, 0b00100100, 0b01111010, 0b01110110, 0b00100111,
  0b01010111, 0b01011111, 0b01100100, 0b01111111, 0b01110111
};

voidsetup()
{
pinMode(DATA_PIN, OUTPUT);
pinMode(CLOCK_PIN, OUTPUT);
pinMode(LATCH_PIN, OUTPUT);
pinMode(BUTTON_PIN, INPUT_PULLUP);
}
 voidloop()
{
  // считаем клики кнопки, как уже делали это раньше
if (buttonWasUp&& !digitalRead(BUTTON_PIN)) {
delay(10);
if (!digitalRead(BUTTON_PIN))
clicks = (clicks + 1) % 10;
  }
buttonWasUp = digitalRead(BUTTON_PIN);
  // для записи в 74HC595 нужно притянуть пин строба к земле
digitalWrite(LATCH_PIN, LOW);
  // задвигаем (англ. shiftout) байт-маску бит за битом,
  // начиная с младшего (англ. LeastSignificantBitfirst)
shiftOut(DATA_PIN, CLOCK_PIN, LSBFIRST, segments[clicks]);
  // чтобы переданный байт отразился на выходах Qx, нужно
  // подать на пин строба высокий сигнал
digitalWrite(LATCH_PIN, HIGH);
}
```

Эксперимент 15. Комнатный термометр

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 светодиодная шкала
1 резистор номиналом 10 кОм
1 термистор
10 резисторов номиналом 220 Ом
14 проводов

Схема на макетке



Скетч

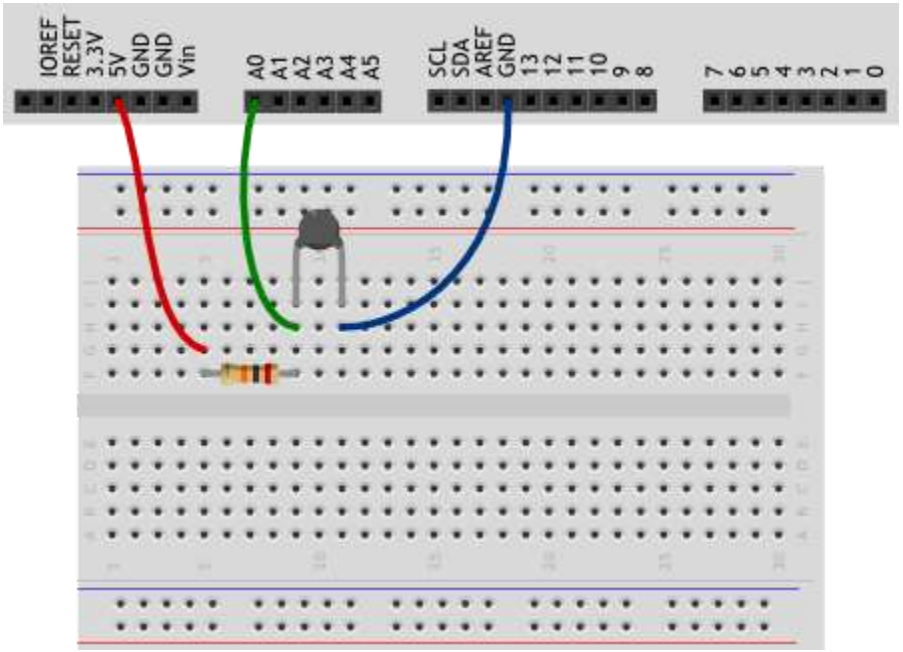
```
// Огромное количество готового кода уже написано другими людьми
// и хранится в виде отдельных файлов, которые называются
// библиотеками. Для использования кода из библиотеки, её нужно
// подключить (англ. include). Библиотека «math» даёт разные
// математические функции, в том числе функцию логарифма
// (англ. log), которая нам понадобится далее
#include<math.h>

#define FIRST_LED_PIN 2
#define LED_COUNT     10

// Параметр конкретного типа термистора (из datasheet):
#define TERMIST_B 4300

#define VIN 5.0
 voidsetup()
{
for (int i = 0; i < LED_COUNT; ++i)
pinMode(i + FIRST_LED_PIN, OUTPUT);
}
 voidloop()
{
  // вычисляем температуру в °Cс помощью магической формулы.
  // Используем при этом не целые числа, а вещественные. Их ещё
  // называют числами с плавающей (англ. float) точкой. В
  // выражениях с вещественными числами обязательно нужно явно
  // указывать дробную часть у всех констант. Иначе дробная
  // часть результата будет отброшена

floatvoltage = analogRead(A0) * VIN / 1023.0;
float r1 = voltage / (VIN - voltage);


floattemperature = 1./( 1./(TERMIST_B)*log(r1)+1./(25. + 273.) ) - 273;

for (int i = 0; i < LED_COUNT; ++i) {
    // при 21°C должен гореть один сегмент, при 22°C — два и
    // т.д. Определяем должен ли гореть i-й нехитрым способом
booleanenableSegment = (temperature>= 21+i);
digitalWrite(i + FIRST_LED_PIN, enableSegment);
  }
}
```

Эксперимент 16. Метеостанция

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 резистор номиналом 10 кОм
1 термистор
3 провода

Схема на макетке



Скетч

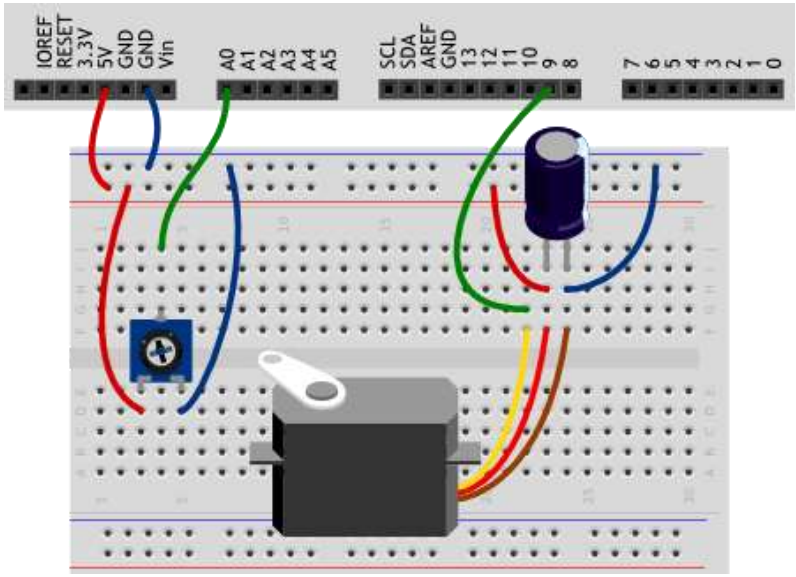
```
#include<math.h>
intminute = 1;

// Параметр конкретного типа термистора (из datasheet):
#define TERMIST_B 4300

#define VIN 5.0

voidsetup()
{
Serial.begin(9600);
Serial.println("Minute\tTemperature");
}

voidloop()
{

floatvoltage = analogRead(A0) * VIN / 1024.0;
float r1 = voltage / (VIN - voltage);


floattemperature = 1./( 1./(TERMIST_B)*log(r1)+1./(25. + 273.) ) - 273;
  // печатаем текущую минуту и температуру, разделяя их табом.
  // println переводит курсор на новую строку, а print — нет
Serial.print(minute);
Serial.print("\t");
Serial.println(temperature);

delay(60000); // засыпаем на минуту
  ++minute;     // увеличиваем значение минуты на 1

  // откройте окно SerialMonitor в среде Arduino, оставьте на
  // сутки, скопируйте данные в Excel, чтобы построить графики
}
```

Эксперимент 17. Пантограф

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 сервопривод
1 конденсатор емкостью 220 мкФ
1 потенциометр
11 проводов

Схема на макетке



Скетч

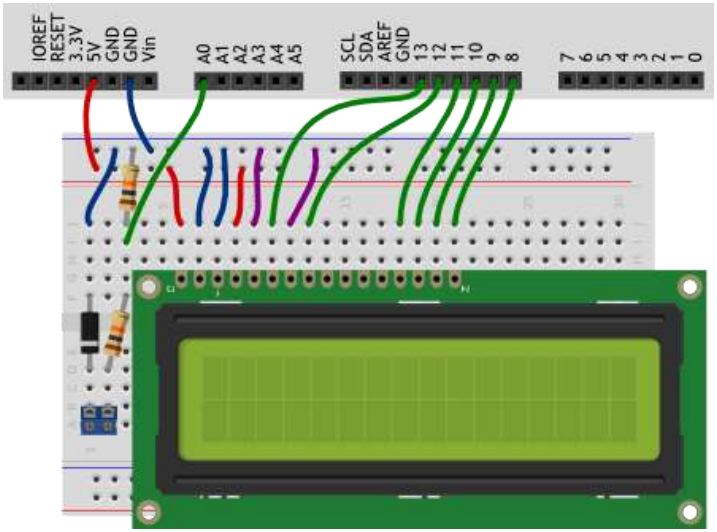
```
#include<Servo.h>

#define POT_MAX_ANGLE 270.0 // макс. угол поворота потенциометра

// объявляем объект типа Servo с именем myServo. Ранее мы
// использовали int, boolean, float, а теперь точно также
// используем тип Servo, предоставляемый библиотекой. В случае
// Serial мы использовали объект сразу же: он уже был создан
// для нас, но в случае с Servo, мы должны сделать это явно.
// Ведь в нашем проекте могут быть одновременно несколько
// приводов, и нам понадобится различать их по именам
ServomyServo;

voidsetup()
{
  // прикрепляем (англ. attach) нашу серву к 9-му пину. Явный
  // вызов pinMode не нужен: функция attach сделает всё за нас
myServo.attach(9);
}

voidloop()
{
intval = analogRead(A0);
  // на основе сигнала понимаем реальный угол поворота движка.
  // Используем вещественные числа в расчётах, но полученный
  // результат округляем обратно до целого числа
intangle = int(val / 1024.0 * POT_MAX_ANGLE);
  // обычнаясерва не сможет повторить угол потенциометра на
  // всём диапазоне углов. Она умеет вставать в углы от 0° до
  // 180°. Ограничиваем угол соответствующе
angle = constrain(angle, 0, 180);
  // и, наконец, подаём серве команду встать в указанный угол
myServo.write(angle);
}
```

Эксперимент 18. Тестер батареек

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
2 резистора номиналом 10 кОм
1 выпрямительный диод
1 текстовый экран
16 проводов 1 клеммник

Схема на макетке



Скетч

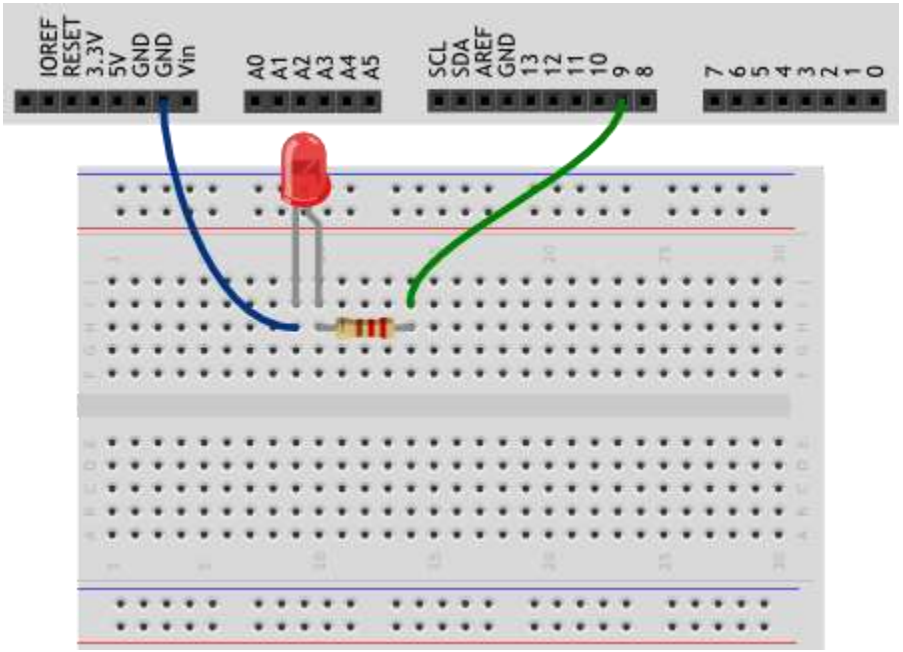
```
// экраном (англ. LiquidCrystalDisplay или просто LCD)
#include<LiquidCrystal.h>
// на диоде, защищающем от неверной полярности, падает доля
// напряжения (англ. voltagedrop). Необходимо это учитывать
#define DIODE_DROP  0.7
// Объявляем объект, для управления дисплеем. Для его создания
// необходимо указать номера пинов, к которым он подключен в
// порядке:       RS   E DB4 DB5 DB6 DB7
LiquidCrystallcd(13, 12, 11, 10,  9,  8);

voidsetup()
{
  // начинаем работу с экраном. Сообщаем объекту количество
  // строк и столбцов. Опять же, вызывать pinMode не требуется:
  // функция begin сделает всё за нас
lcd.begin(16, 2);
  // печатаем сообщение на первой строке
lcd.print("Batteryvoltage:");
}
 voidloop()
{
  // высчитываем напряжение подключенной батарейки
floatvoltage = analogRead(A0) / 1024.0 * 10.0;
  // если напряжение на делителе напряжения было зафиксировано,
  // нужно прибавить напряжение на диоде, т.к. оно было съедено
if (voltage> 0.1)
voltage += DIODE_DROP;
  // устанавливаем курсор, колонку 0, строку 1. На деле — это
  // левый квадрат 2-й строки, т.к. нумерация начинается с нуля
lcd.setCursor(0, 1);
  // печатаем напряжение в батарейке с точностью до сотых долей
lcd.print(voltage, 2);
  // следом печатаем единицы измерения
lcd.print(" Volts");
}
```

Эксперимент 19. Светильник, управляемый по USB

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 светодиод
1 резистор номиналом 220 Ом
2 провода

Схема на макетке



Скетч

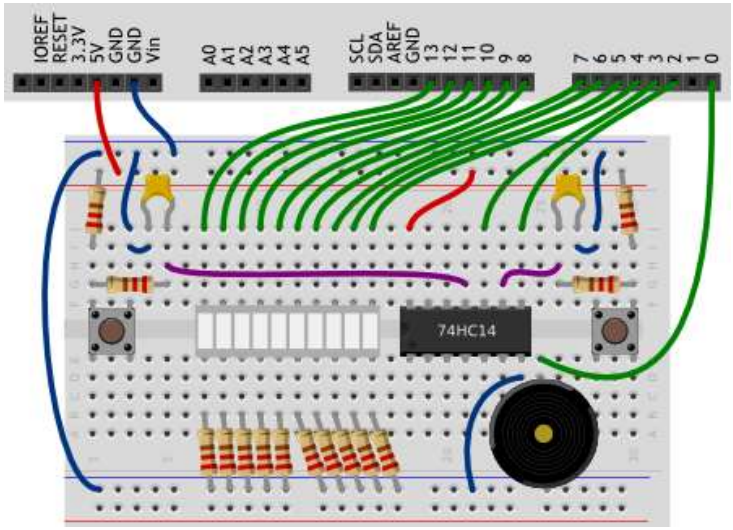
```
#define LED_PIN 9
// для работы с текстом существуют объекты-строки (англ. string)
Stringmessage;

voidsetup()
{
pinMode(LED_PIN, OUTPUT);
Serial.begin(9600);
}

voidloop()
{
while (Serial.available()) {
   // считываем (англ. read) пришедший символ в переменную
charincomingChar = Serial.read();
   // не стоит путать целые числа и символы. Они соотносятся
   // друг с другом по таблице, называемой кодировкой. Например
   // '0' — это 48, '9' — 57, 'A' — 65, 'B' — 66 и т.п. Символы
   // в программе записываются в одинарных кавычках
if (incomingChar>= '0' &&incomingChar<= '9') {
     // если пришёл символ-цифра, добавляем его к сообщению
message += incomingChar;
   } elseif (incomingChar == '\n') {
     // если пришёл символ новой строки, т.е. enter, переводим
     // накопленное сообщение в целое число (англ. tointeger).
     // Так последовательность символов '1', '2', '3' станет
     // числом 123. Результат выводим на светодиод
analogWrite(LED_PIN, message.toInt());
     // обнуляем накопленное сообщение, чтобы начать всё заново
message = "";
   }
  }
  // посылайте сообщения-числа с компьютера через SerialMonitor
}
```

Эксперимент 20. Перетягивание каната

Список деталей для эксперимента

1 плата ArduinoUno
1 беспаечная макетная плата
1 светодиодная шкала
10 резисторов номиналом 220 Ом
4 резисторов номиналом 100 кОм
2 тактовых кнопки
2 керамических конденсатора номиналом 100 нФ
1 пьезопищалка
1 инвертирующий триггер Шмитта
24 провода

Схема на макетке



Скетч

```
#define BUZZER_PIN     0
#define FIRST_BAR_PIN  4
#define BAR_COUNT      10
#define MAX_SCORE      20
// глобальные переменные, используемые в прерываниях (см. далее)
// должны быть отмечены как нестабильные (англ. volatile)
volatileintscore = 0;

voidsetup()
{
for (int i = 0; i < BAR_COUNT; ++i)
pinMode(i + FIRST_BAR_PIN, OUTPUT);
pinMode(BUZZER_PIN, OUTPUT);
attachInterrupt(INT1, pushP1, FALLING); // INT1 — это 3-йпин
attachInterrupt(INT0, pushP2, FALLING); // INT0 — это 2-й пин
}

void pushP1() { ++score; } // функция-прерывание 1-го игрока
void pushP2() { --score; } // функция-прерывание 2-го игрока
voidloop()
{
tone(BUZZER_PIN, 2000, 1000); // даём сигнал к старту.
  // пока никто из игроков не выиграл, обновляем «канат»
while (abs(score) < MAX_SCORE) {
intbound = map(score, -MAX_SCORE, MAX_SCORE, 0, BAR_COUNT);
intleft = min(bound, BAR_COUNT / 2 - 1);
intright = max(bound, BAR_COUNT / 2);
for (int i = 0; i < BAR_COUNT; ++i)
digitalWrite(i + FIRST_BAR_PIN, i >= left&& i <= right);
  }
tone(BUZZER_PIN, 4000, 1000); // даём сигнал победы
while (true) {} // «подвешиваем» плату до перезагрузки
}
```

Список использованной литературы.

1. Мир информатики: Базовое учебное пособие. Под ред. А.В. Могилева. Смоленск: Ассоциация XXI век, 2003, 80 с.
2. Горячев А.В. Информатика и ИКТ. (Мой инструмент компьютер). Учебник для учащихся 5-6 классов. – М.: Баласс, 2010. – 80 с
3. В.А. Козлова, Робототехника в образовании
4. Белиовская Л.Г., Белиовский А.Е. Программируем микрокомпьютер NXT в LabVIEW. – М.: ДМК, 2010, 278 стр.;
5. Ньютон С. Брага. Создание роботов в домашних условиях. – М.: NT Press, 2007, 345 стр.;
6. Филиппов С.А. Робототехника для детей и родителей. С-Пб, «Наука», 2011г.
7. http://amperka.ru/
8. Алексеев, Н,Г, Концепция развития исследовательской деятельности учащихся/ Н.Г.Алексеев, А.В. Леонтович, Л.Ф. Фомина// Исследовательская работа школьников.- 2001-№1.-С. 24-34.
9. Асмолов, А.Г. Как проектировать универсальные учебные действия в начальнойшколе. От действий к мысли : пособие для учителя.-3-е изд. / А.Г. Асмолов, Г.В.Бурменская, И.А. Володарская и др.; под ред. А.Г. асмолова.-М. : Просвещение, 2011.
10. Давыдов, В.В. Теория развивающего обучения / В.В. Давыдов.-М. : ИНТОР, 1996.