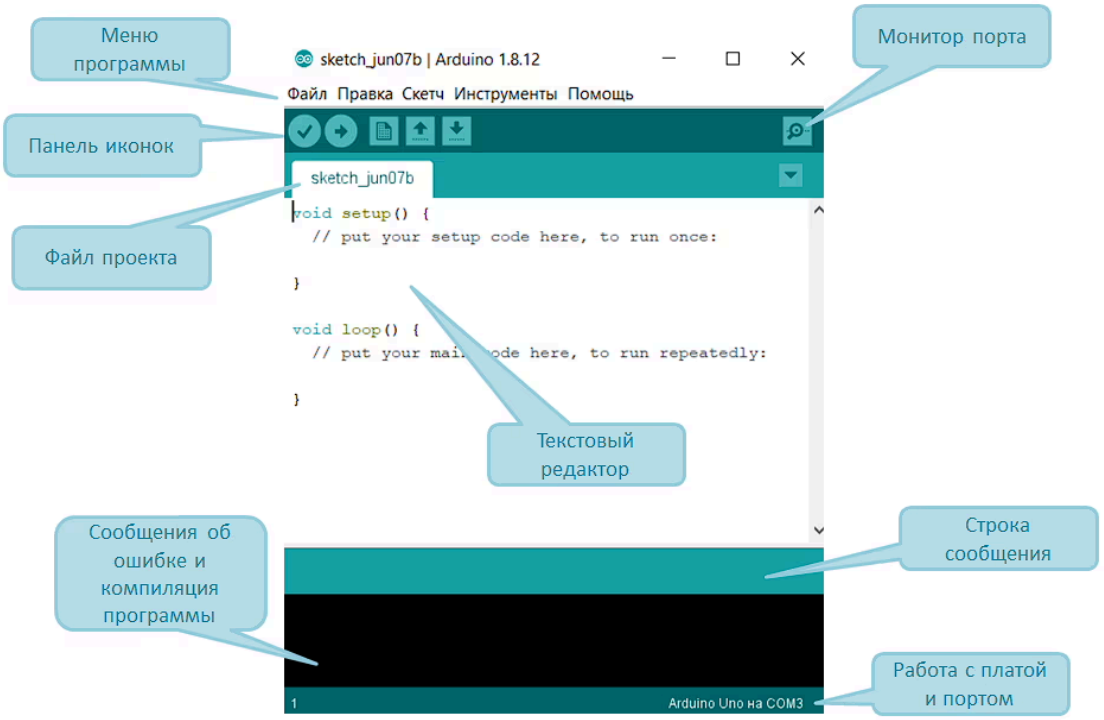
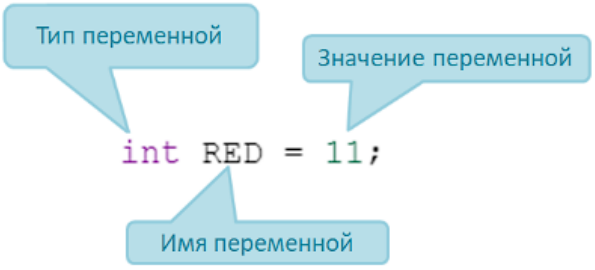
Основные понятия

Интерфейс Arduino IDE




```
1  void setup()  //название процедуры
2      //Эта процедура выполняется единожды в начале работы программы
3  {
4      /*Далее последовательно выполняются все команды,
5       включенные в фигурные скобки, и каждая команда должна
6       завершиться символом «;» */
7  }
8
9  void loop()   //название процедуры
10 {
11     /* В отличие от setup, процедура loop выполняется постоянно.
12      Т е, как только последовательно выполнятся все команды,
13      заключенные в фигурные скобки, она запускается заново*/
14 }
```

Общий вид объявления переменной выглядит так:



"=" - присваивание переменной.

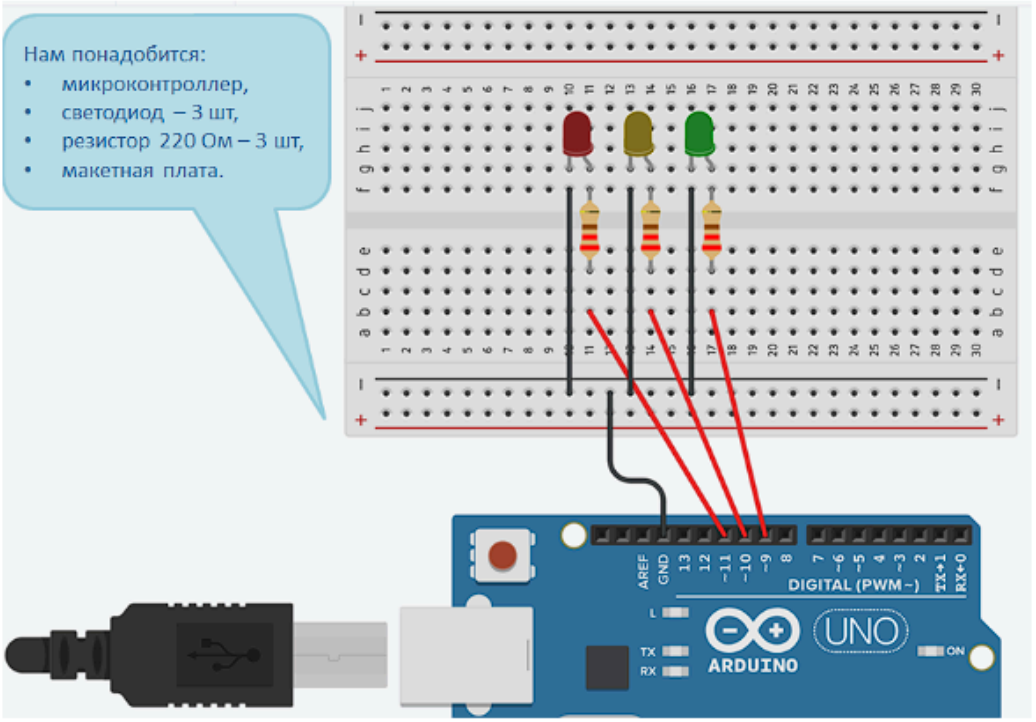
pinMode(13, OUTPUT); - это команда, которая сообщает плате как настроить цифровой вывод. Процедура void setup() содержит одну команду *pinMode* с данными *13* и *OUTPUT*, по другому они называются *аргументами.* Аргумент – это то, что передается в процедуру для уточнения того, что мы хотим сказать командой. Т.е. мы указываем с каким пином мы будем работать, в нашем случае 13 пин, и как ее настроить: на входные(INPUT) или выходные (*OUTPUT*) данные.

digitalWrite(13, HIGH); / digitalWrite(13, LOW); - это команда, которая может включать или выключать любой вывод, настроенный как ВЫХОД (*OUTPUT*). Первый аргумент (*13*) указывает, какой вывод должен быть включен или выключен. Второй аргумент указывает включить вывод (*HIGH*) или выключить его (*LOW*). Иначе можно сказать, *HIGH* - подача высокого сигнала, то есть 5 Вольт (рабочее напряжение контроллера) , *LOW* - подача низкого сигнала, то есть 0 Вольт.

delay(1000); - это команда, которая указывает плате ничего не делать в течение стольких миллисекунд, сколько указано в аргументе (*1000*). 1000 миллисекунд - это 1 секунда.

digitalRead() - команда, которая проверяет цифровой вывод.

Светодиоды.

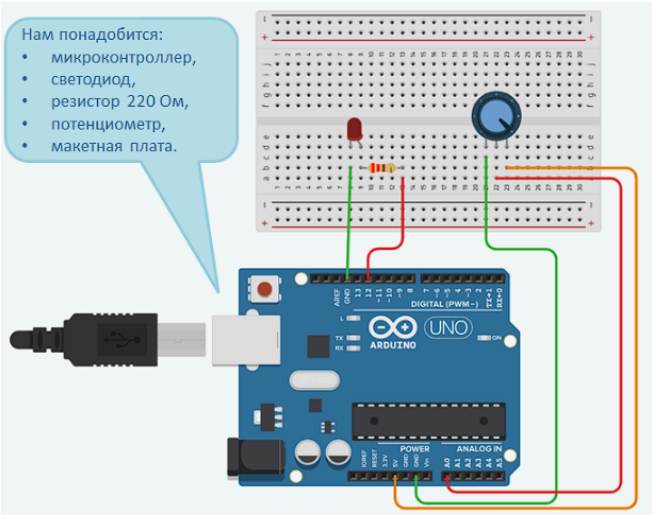


пример управления

```
int RED = 11;
int YELLOW = 10;        //Задаем каждому пину свою переменную целого типа (int)
int GREEN = 9;

void setup() {
pinMode(RED, OUTPUT);
pinMode(YELLOW, OUTPUT);  //Определяем выводы переменных как ВЫХОД
pinMode(GREEN, OUTPUT);
}
void loop() {
    digitalWrite(RED, HIGH);  //Подаем высокий сигнал (HIGH) на светодиод
          delay(10000); // Ждем
    digitalWrite(YELLOW, HIGH);
          delay(2000);// Ждем
    digitalWrite(GREEN, HIGH);
    digitalWrite(RED, LOW);    //Подаем низкий сигнал (LOW) на светодиод
    digitalWrite(YELLOW, LOW);
          delay(10000);// Ждем
    digitalWrite(YELLOW, HIGH);
    digitalWrite(GREEN, LOW);
        delay(2000);// Ждем
    digitalWrite(YELLOW, LOW);
}
```

Потенциометр

Потенциометр - это электронный компонент, представляющий из себя переменный резистор, предназначенный для изменения напряжения. Он часто используется в различных устройствах управления. Например, громкости звука, мощности или вращая потенциометр можно регулировать интенсивность света в своей квартире.

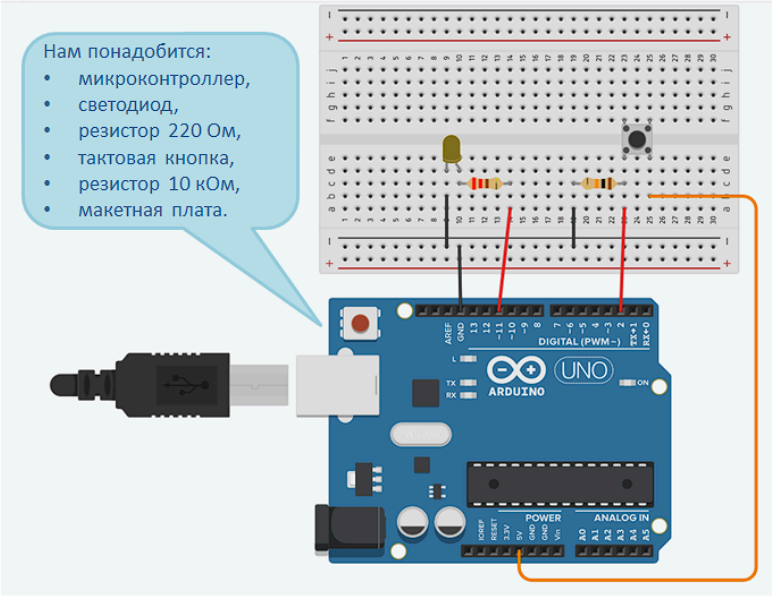


Катод светодиода отправляем на землю, а ножку анода через резистор на 220 Ом подключаем к 12 пину. Крайние ножки потенциометра подсоединим к 5 В и GND (*неважно какую ножку куда. Поменяется лишь направление увеличения и уменьшения напряжения*). Аналоговый сигнал мы будем считывать со средней ножки, подключенной к пину A0.

```
int LED = 12;
int POTEN = A0;
int VRA;
int SVET;

void setup()
{
  pinMode(LED, OUTPUT);   //указываем LED на выход
  pinMode(POTEN, INPUT);  //указываем POTEN на вход
}

void loop()
{
  VRA = analogRead(POTEN);//считываем значение с пина A0 от 0-1023
  SVET = VRA / 4;         //считаем значение для аналогового
                          //выхода для светодиода
  analogWrite(LED, SVET);//можно подать значение от 0-255
}
```

Тактовая кнопка

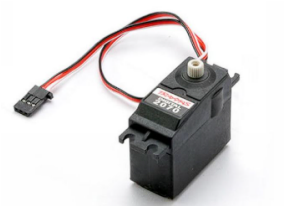
Тактовая кнопка - это самый простой датчик нажатия или касания. Кнопка имеет два состояния: включена и выключена. Она замыкает цепь при нажатии и размыкает цепь, когда вы ее отпускаете. У тактовой кнопки есть четыре ножки, соединенные попарно. При нажатии кнопки, все четыре контакта замыкаются.



Катод светодиода отправляем на землю, а ножку анода через резистор на 220 Ом подключаем к 11 пину. Одну пару ножек кнопки подключаем к напряжению 5 В, другую пару ножек подключаем к пину 2 для считывания сигналов, и эту же пару контактов через резистор на 10 кОм отправляем на GND.

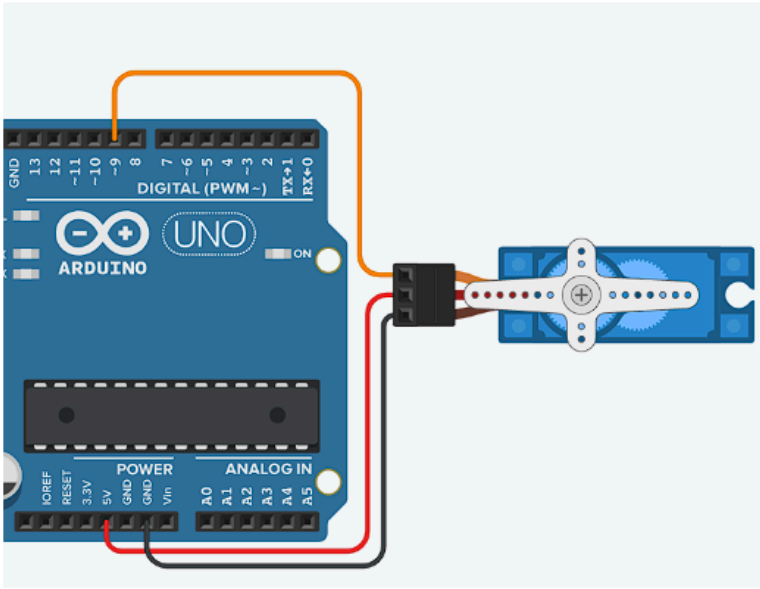
```
int LED = 11;
int BUTTON = 2;
int val = 0;

void setup()
{
  pinMode(LED, OUTPUT);     //значение LED устанавливаем на выход
  pinMode(BUTTON, INPUT);   //значение BUTTON устанавливаем на ВХОД
}
void loop()
{
  val = digitalRead(BUTTON); //считываем значение с кнопки
                             //и помещаем в переменную val
  if (val == HIGH)           //если кнопка нажата,
  {
    digitalWrite(LED, HIGH); //то светодиод загорится
  } else {                   //иначе
    digitalWrite(LED, LOW);  //светодиод не загорится
  }
}
```

Сервомотор



Сервомотор - это механический привод с автоматической корректировкой положения.

Сервопривод состоит из мотора постоянного тока, редуктора, датчика положения и контроллера управления, благодаря этому можно заставить серводвигатель повернуться на заданный угол и удерживать его в этом положении. Диапазон углов поворота у большинства сервомоторов ограничиваются 180 градусами.




```
#include <Servo.h>  // подключаем беблиотеку

Servo Servo1;       //создаем объект для работы с серво

void setup()
{
  Servo1.attach(9); //активируем сервопривод на 9 пин
}

void loop()
{
  // 0 градусов
  Servo1.write(0);
  delay(2000);
  // 90 градусов
  Servo1.write(90);
  delay(2000);
  // 180 градусов
  Servo1.write(180);
  delay(2000);
}
```

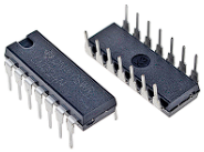
Для управления сервоприводом в Arduino нужно подключить библиотеку <Servo.h> . Далее создаем объект на основе этой библиотеки командой Servo и задаем имя Servo1. Далее активируем наш сервопривод командой attach и указываем через какой порт будем управлять сервомотором (9).

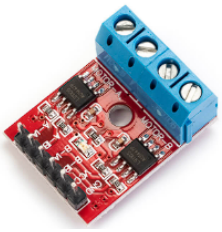
В основной цикле с периодом в 2 секунды мы задаем три положения: 0, 90 и 180 градусов.

Управление электродвигателем постоянного тока

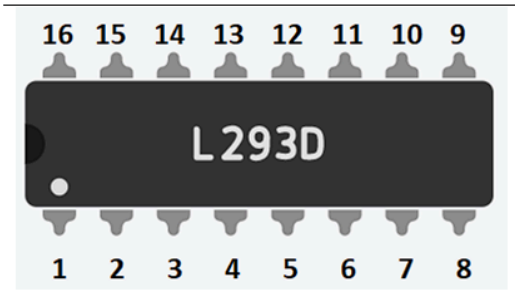
Для управления двигателем постоянного тока необходим драйвер. Драйвер - это такая микросхема, которая позволяет подавать более высокое напряжение на мотор и управлять направлением вращения мотора и его мощностью.

Микросхема **Драйвер**



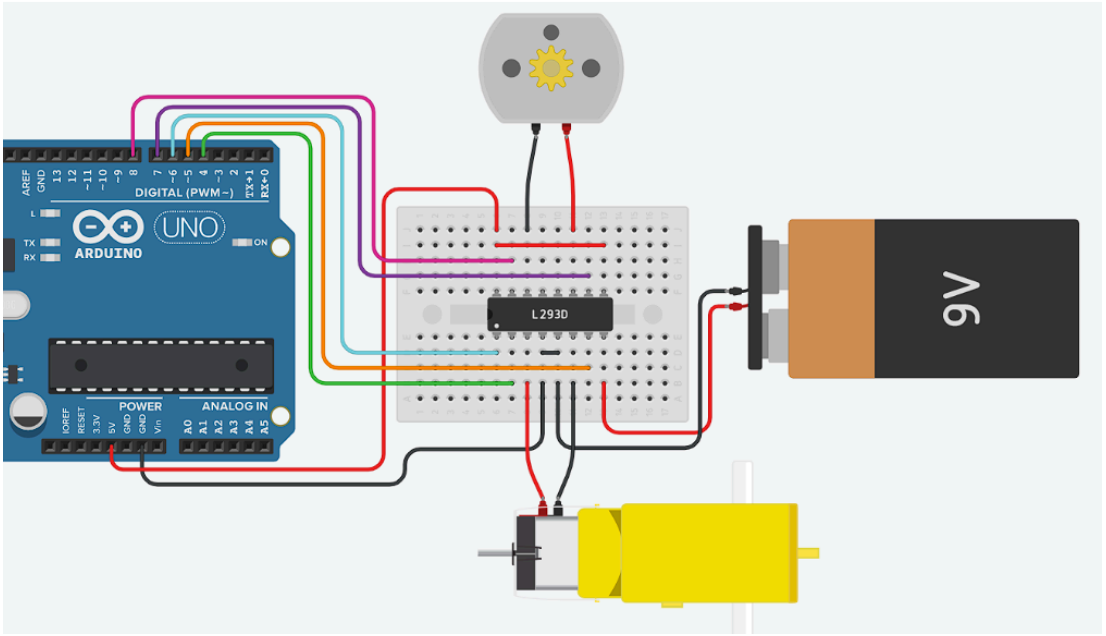


Для работы с микросхемами часто требуется документация, что бы знать какие выводы за что отвечают. Рассмотрим контакты подробнее:



4, 5, 12, 13	Земля	0 вольт
8	Питание 2	Питание моторов от источника питания
16	Питание 1	Питание микросхемы
3, 6, 11, 14	Выход 1, 2, 3, 4	Подключение моторов
2, 7, 10, 15	Вход 1, 2, 3, 4	Управление направления вращения моторов
1, 9	Включение 1 и 2, 3 и 4	Управление мощности моторов

Соберем схему управления двигателями через микросхему, подавая дополнительно напряжение с батареи на 9 вольт. Причем двигателю постоянного тока будем менять направление, а мотору-редуктору будем менять направление и мощность.



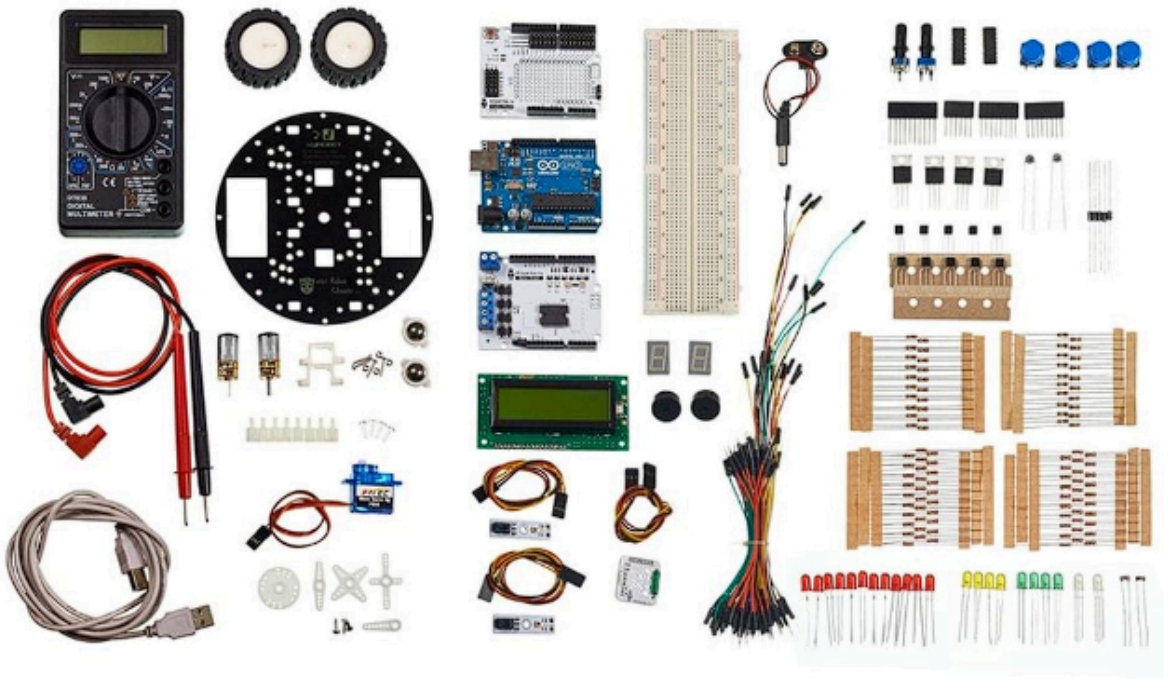
Напишем скетч для управления двигателями. На двигатели мы будем подавать цифровой сигнал (т.е. высокое и низкое напряжение). Чтобы менять мощность мотора - редуктора, мы будем использовать ШИМ сигнал (т.е. значения от 0 до 255): **#define - удобная директива, благодаря ей присвоенные значения (константы) не занимают программной памяти.**

```
#define IN1 4  // Создаем переменные для управления
#define IN2 5  // первым мотором
#define EN1 6  // ШИМ-пин для управления мощностью мотора

#define IN3 7  // Создаем переменные для управления
#define IN4 8  // вторым мотором

void setup()
{
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);  //Все пины работают на выход
  pinMode(IN4, OUTPUT);
  pinMode(EN1, OUTPUT);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);  // Все управляющие пины устанавливаем
  digitalWrite(IN3, LOW);  // на низкое напряжения
  digitalWrite(IN4, LOW);
}

void loop()
{
  digitalWrite(IN1, HIGH);  //Мотор-редуктор вращается в
  digitalWrite(IN2, LOW);   //одном направлении
  analogWrite(EN1, 100);    //с мощностью 100 об/м

  digitalWrite(IN3, HIGH);  //Устанавливаем направление вращения
  digitalWrite(IN4, LOW);   //на двигатель постоянного тока
  delay(2000);        // ждем 2 секунды

  digitalWrite(IN1, LOW);  //Меняем направление вращения
  digitalWrite(IN2, HIGH); //мотора-редуктора
  analogWrite(EN1, 200);   //Устанавливаем мощность 200 об/м

  digitalWrite(IN3, LOW);  //Меняем направление вращения
  digitalWrite(IN4, HIGH); //на двигателю постоянного тока
  delay(2000);        // ждем 2 секунды
}
```

Задания для тренировки:

1. *Сделать светофор*
2. *Управлять светодиодом через кнопку*
3. *Управлять яркостью светодиода через потенциометр*
4. *Покрутить сервопривод*
5. *Подключить мотор*
6. Управлять сервоприводом через потенциометр
7. Управлять скоростью моторчика при помощи потенциометра
8. Собрать трехколесную машинку с двумя моторами и светодиодами на задней части корпуса (по одному красный желтый зеленый)
9. Моторы https://wiki.amperka.ru/продукты:arduino-motor-shield
10. https://wiki.amperka.ru/робототехника:робот-с-датчиками-линии-на-arduino?yscli d=mhdpsbl98g572618390

Что будет выдано?
https://wiki.amperka@Pepoui - отправить мне в лс ссылку на ваш диск.ru/



Компоненты набора

Контроллер
1× Плата Arduino Uno

Сенсоры
2× Датчик линии

1× Датчик наклона
2× Фоторезистор
2× Термистор
4× Кнопка тактовая
2× Потенциометр

Прототипирование и провода
1× Макетная доска
65× Соединительный провод
1× USB-кабель
1× Разъём для батарейки

Механика
1× Двухколёсное шасси робота
1× Сервопривод

Индикация и звук
1× Текстовый ЖК-экран
2× 7-сегментный индикатор
12× Светодиод красный
4× Светодиод жёлтый
4× Светодиод зелёный
2× Трёхцветный светодиод
2× Пьезоизлучатель звука

Базовые компоненты
60× Резистор 220 Ом
20× Резистор 1 кОм
20× Резистор 10 кОм
20× Резистор 100 кОм
10× Биполярный транзистор
4× Транзистор MOSFET
2× Микросхема CD4026
5× Выпрямительный диод

Инструменты
1× Мультиметр цифровой

Платы расширения
1× Драйвер моторов Motor Shield
1× Расширитель портов Troyka Shield