Эксперимент № 1. Управление моторами. Arduino, L293D и фотоэлементы.
Вам понадобится

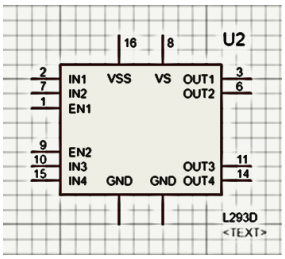
1. Proteus 8.1 Professinal
2. http://tinkercad.com

Список деталей для эксперимента

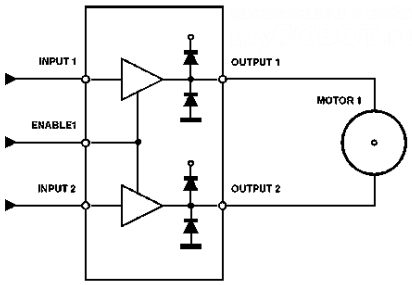
- 1 микросхема драйвера моторов L293D
- 1 беспаечная макетная плата
- 1 фоторезистор, или 1 фотодиод
- 1 переменный резистор (потенциометр(POT-HG(Proteus))
- 1 резистор номиналом 4 КОм (ERJ-1WYJ752U(Proteus))
- провода «папа-мама»
- мотор DC (Motor-DC)
- Arduino UNO R3

Пошаговое руководство для проведения эксперимента

Для управления двигателями робота необходимо устройство, которое бы преобразовывало управляющие сигналы малой мощности в токи, достаточные для управления моторами. Такое устройство называют **драйвером двигателей.** Мы остановимся на самом простом драйвере управления двигателями, выполненном в виде полностью готовой к работе микросхемы. Эта микросхема называется **L293D** и является одной из самых распространенных микросхем, предназначенных для этой цели.



L293D содержит сразу два драйвера для управления электродвигателями небольшой мощности (четыре независимых канала, объединенных в две пары). Имеет две пары входов (INPUT) для управляющих сигналов и две пары выходов (OUTPUT) для подключения электромоторов. Кроме того, у **L293D** есть два входа для включения каждого из драйверов (ENABLE). Эти входы используются для управления скоростью вращения электромоторов с помощью широтно модулированного сигнала (ШИМ). **L293D** обеспечивает подключение электродвигателей с большим напряжением питания, чем у микросхемы. Принцип работы каждого из драйверов, входящих в состав микросхемы, идентичен, поэтому рассмотрим принцип работы одного из них.



К выходам OUTPUT1 и OUTPUT2 подключим электромотор MOTOR1. На вход ENABLE1, включающий драйвер, подадим сигнал (соединим с положительным полюсом источника питания +5V). Если при этом на входы INPUT1 и INPUT2 не подаются сигналы, то мотор вращаться не будет. Если вход INPUT1 соединить с положительным полюсом источника питания, а вход INPUT2 - с отрицательным,

то мотор начнет вращаться. Теперь попробуем соединить вход INPUT1 с отрицательным полюсом источника питания, а вход INPUT2 - с положительным. Мотор начнет вращаться в другую сторону (Рисунок 1. Подключение мотора к микросхеме L293D).

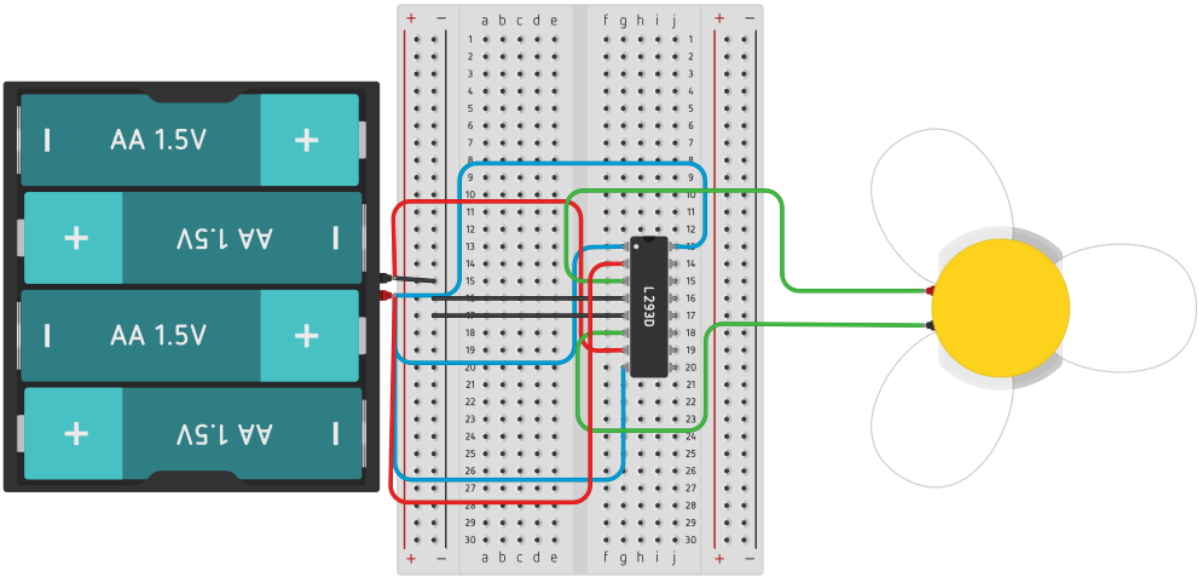
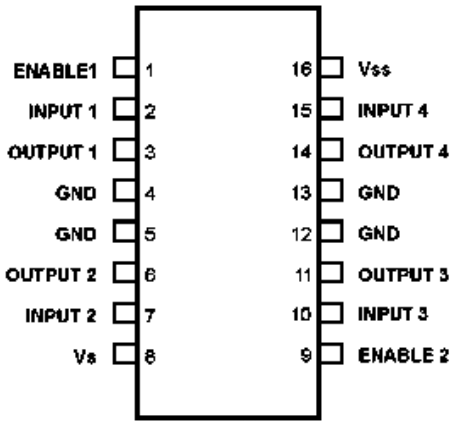


Рисунок 1. Подключение мотора к микросхеме L293D.

Попробуем подать сигналы одного уровня сразу на оба управляющих входа INPUT1 и INPUT2 (соединить оба входа с положительным полюсом источника питания или с отрицательным) - мотор вращаться не будет. Если мы уберем сигнал с входа ENABLE1, то при любых вариантах наличия сигналов на входах INPUT1 и INPUT2 мотор вращаться не будет. Представить лучше принцип работы драйвера двигателя можно, рассмотрев следующую таблицу:

ENABLE1	INPUT1	INPUT2	OUTPUT1	OUTPUT2
1	0	0	0	0
1	1	0	1	0
1	0	1	0	1
1	1	1	1	1

Теперь рассмотрим назначение выводов микросхемы L293D.



- Входы ENABLE1 и ENABLE2 отвечают за включение каждого из драйверов, входящих в состав микросхемы.
- Входы INPUT1 и INPUT2 управляют двигателем, подключенным к выходам OUTPUT1 и OUTPUT2.
- Входы INPUT3 и INPUT4 управляют двигателем, подключенным к выходам OUTPUT3 и OUTPUT4.
- Контакт Vs соединяют с положительным полюсом источника электропитания двигателей или просто с положительным полюсом питания, если питание схемы и двигателей единое. Проще говоря, этот контакт отвечает за питание электродвигателей.
- Контакт Vss соединяют с положительным полюсом источника питания. Этот контакт обеспечивает питание самой микросхемы.
- Четыре контакта GND соединяют с "землей" (общим проводом или отрицательным полюсом источника питания). Кроме того, с помощью этих контактов обычно обеспечивают теплоотвод от микросхемы, поэтому их лучше всего распаивать на достаточно широкую контактную площадку.

Характеристики микросхемы L293D

- напряжение питания двигателей (Vs) - 4,5...36V
- напряжение питания микросхемы (Vss) - 5V
- допустимый ток нагрузки - 600mA (на каждый канал)
- пиковый (максимальный) ток на выходе - 1,2A (на каждый канал)
- логический "0" входного напряжения - до 1,5V
- логическая "1" входного напряжения - 2,3...7V
- скорость переключений до 5 kHz.
- защита от перегрева

Сделать робота можно, используя лишь одну микросхему драйвера моторов и пару фотоэлементов. В зависимости от способа соединения моторов, микросхемы и фотоэлементов робот будет двигаться на свет или, наоборот, прятаться в темноту, бежать вперед в поисках света или пятиться, как крот, назад. Принцип поведения робота основывается на "фоторецепции" и является типичным для целого класса **BEAM-роботов**. В живой природе, которой будет подражать наш робот, фоторецепция - одно из основных фотобиологических явлений, в котором свет выступает как источник информации. В качестве первого опыта обратимся к устройству **BEAM-робота**, двигающегося вперед, когда на него падает луч света, и останавливающегося, когда свет перестает его освещать. Поведение такого робота называется фотокинезисом - ненаправленным увеличением или уменьшением подвижности в ответ на изменения уровня освещённости.

В устройстве робота, кроме микросхемы драйвера моторов **L293D**, будет использоваться только один фотоэлемент и один электромотор (Рисунок 2. Подключение фотодиода для управления мотором к микросхеме L293D). В качестве фотоэлемента можно применить не только фототранзистор, но и фотодиод или фоторезистор. В конструкции робота мы используем фототранзистор n-p-n структуры в качестве фотосенсора. Фототранзисторы на сегодняшний день являются, пожалуй, одним из самых распространенных видов оптоэлектронных приборов и отличаются хорошей чувствительностью и вполне приемлемой ценой.

Принципиальная схема

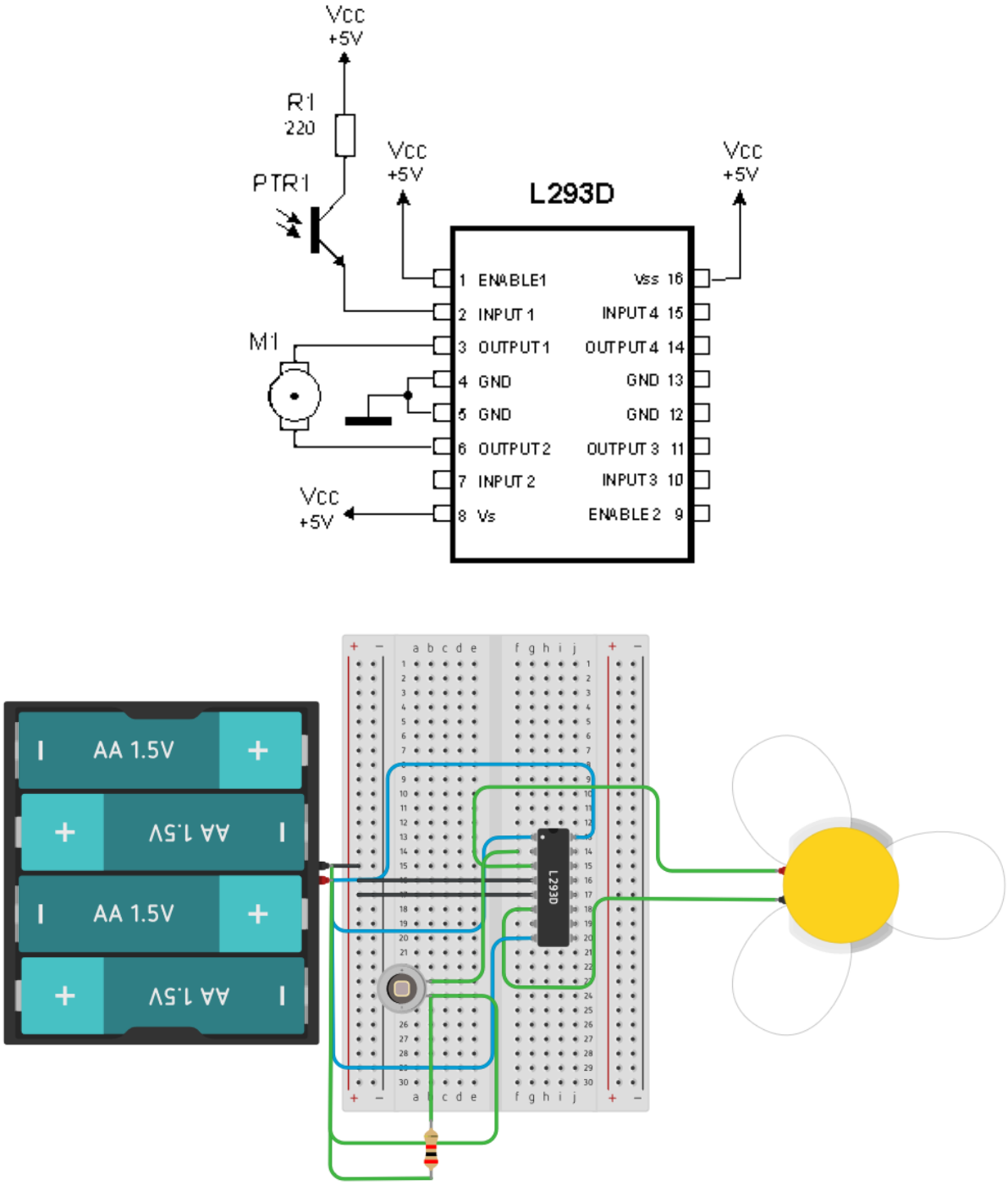


Рисунок 2. Подключение фотодиода для управления мотором к микросхеме L293D (мотор начинает движение после реакции на свет)

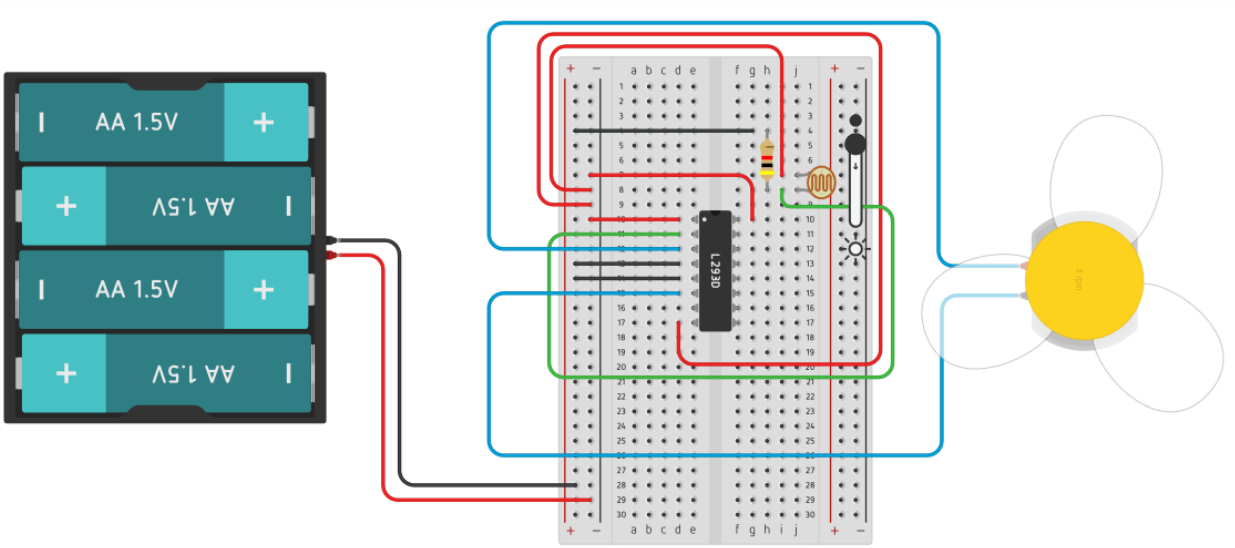


Рисунок 2.1. Подключение фоторезистора для управления мотором к микросхеме L293D (мотор начинает вращение после реакции на свет)

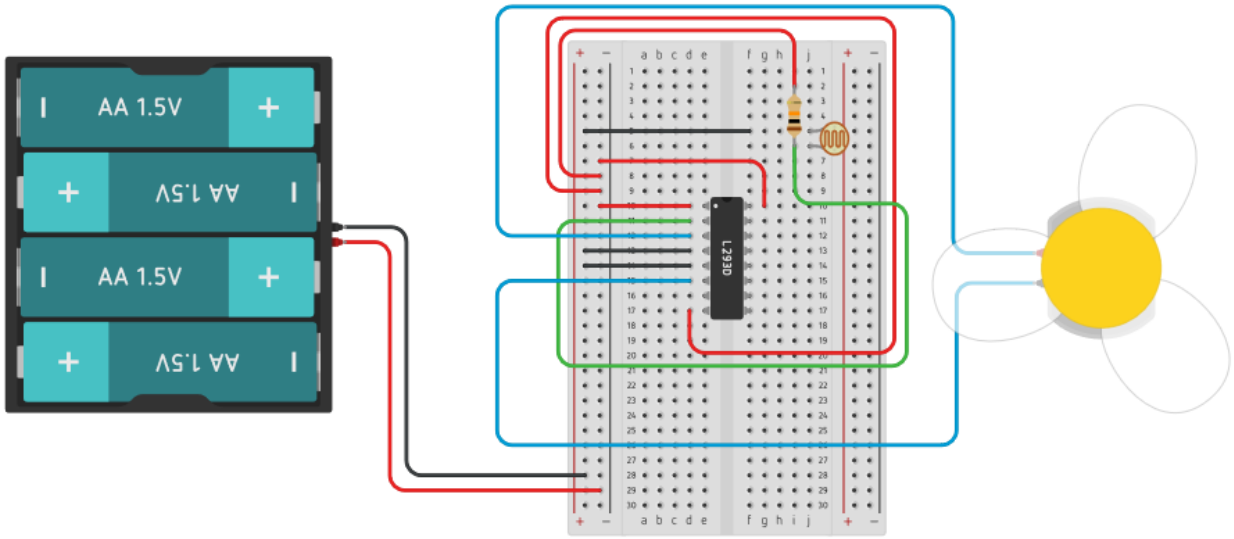


Рисунок 2.2. Подключение фоторезистора для управления мотором к микросхеме L293D (мотор начинает вращение после реакции на темноту)

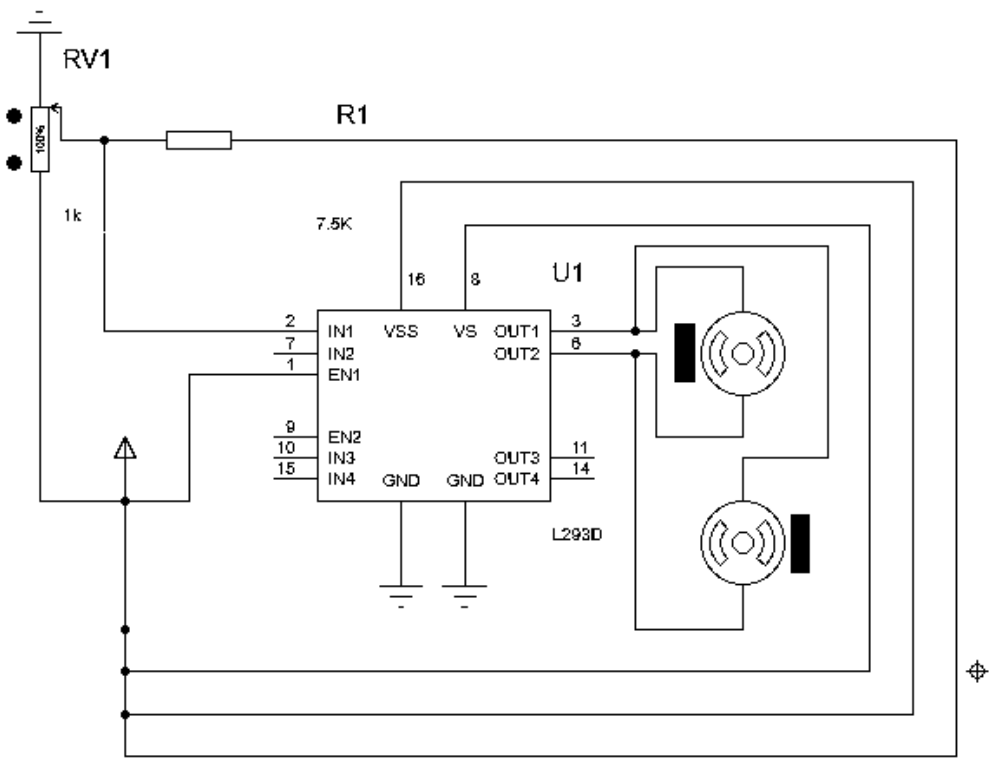


Рисунок 2.3. Подключение потенциометра для управления двумя моторами к микросхеме L293D (Proteus)

На рисунке приведены принципиальная схемы робота, и если Вы еще не очень хорошо знакомы с условными обозначениями, то, исходя из схем, несложно понять принцип обозначения и соединения элементов. Провод, соединяющий различные части схемы с "землей" (отрицательным полюсом источника питания), обычно не изображают полностью, а на схеме рисуют небольшую черточку, обозначающую, что это место соединяется с "землей". Иногда рядом с такой черточкой пишут три буквы "GND", что означает "землю" (ground). Vcc обозначает соединение с положительным полюсом источника питания. Вместо букв Vcc часто пишут +5V, показывая тем самым напряжение источника питания.

Принцип действия схемы робота очень простой. Когда на фототранзистор PTR1 упадет луч света, то на входе INPUT1 микросхемы драйвера двигателей появится положительный сигнал и мотор M1 начнет вращаться. Когда фототранзистор перестанут освещать, сигнал на входе INPUT1 исчезнет, мотор перестанет вращаться и робот остановится. Чтобы скомпенсировать проходящий через фототранзистор ток, в схему введен резистор R1, номинал которого можно выбрать около 200 Ом. От номинала резистора R1 будет зависеть не только нормальная работа фототранзистора, но и чувствительность робота. Если сопротивление резистора будет большим, то робот будет реагировать только на очень яркий свет, если - небольшим, то чувствительность будет более высокой. В любом случае не следует использовать резистор с сопротивлением менее 100 Ом, чтобы предохранить фототранзистор от перегрева и выхода из строя.

Сделать робота, реализующего реакцию фототаксиса (направленного движения к свету или от света), можно с использованием двух фотосенсоров. Когда на один из фотосенсоров такого робота попадает свет, включается соответствующий сенсору электромотор и робот поворачивает в сторону света до тех пор, пока свет не осветит оба фотосенсора и не включится второй мотор. Когда оба сенсора освещены, робот движется навстречу источнику света. Если один из сенсоров перестает освещаться, то робот снова поворачивает в сторону источника света и, достигнув положения, при

котором свет падает на оба сенсора, продолжает свое движение на свет. Если свет перестает падать на фотосенсоры, робот останавливается.

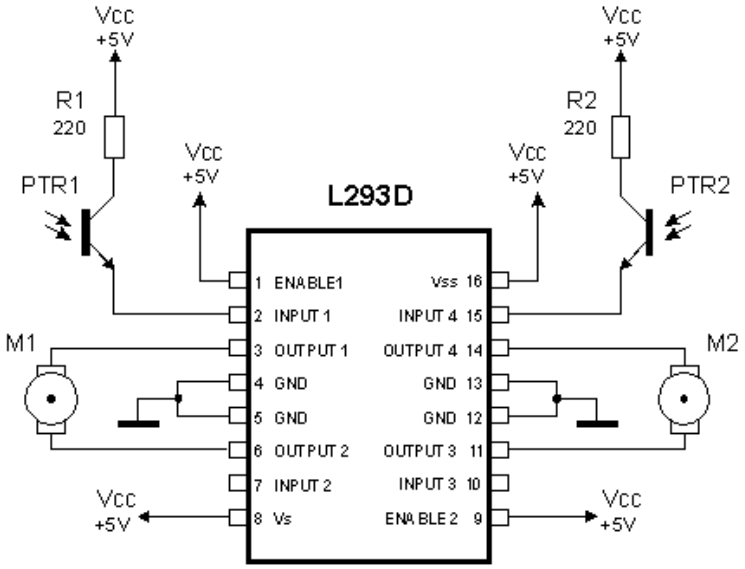


Схема робота симметричная и состоит из двух частей, каждая из которых управляет соответствующим электромотором. По сути, она является как бы удвоенной схемой предыдущего робота. Фотосенсоры следует располагать крест-накрест по отношению к электромоторам так, как показано на рисунке робота выше.

Чтобы **сделать робота**, "бегающего" за рукой, нам понадобятся два ярких светодиода (на схеме LED1 и LED2). Подключим их через резисторы R1 и R4, чтобы скомпенсировать протекающий через них ток и предохранить от выхода из строя. Расположим светодиоды рядом с фотосенсорами, направив их свет в ту же сторону, в которую ориентированы фотосенсоры, и уберем сигнал с входов INPUT2 и INPUT3 (Рисунок 3. Схема робота, движущегося на отраженный свет).

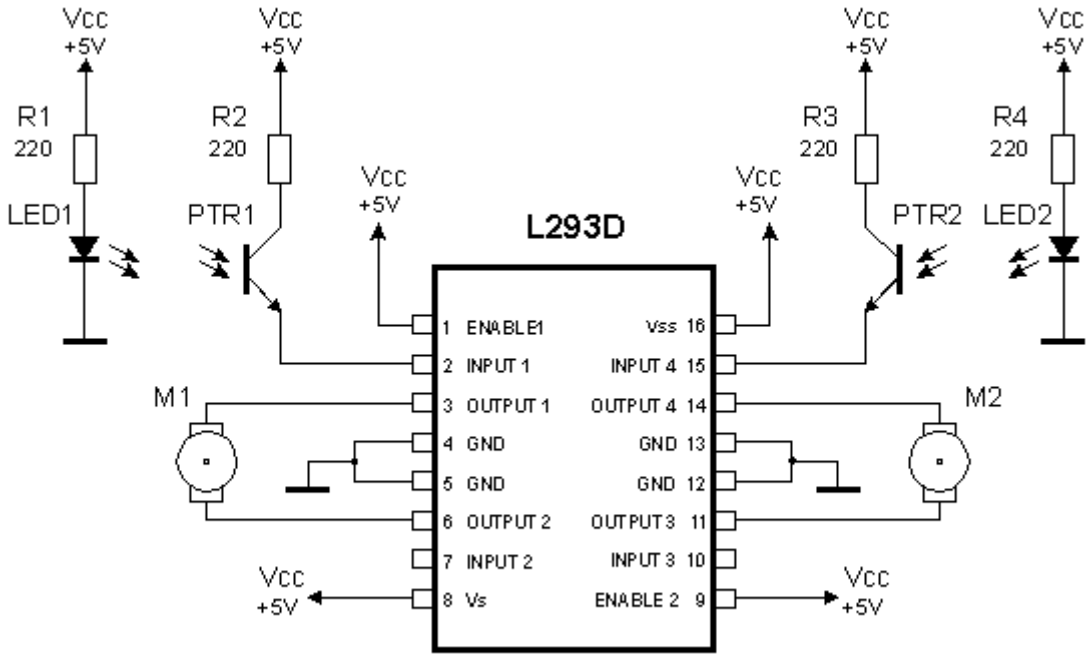


Рисунок 3. Схема робота, движущегося на отраженный свет

Задача получившегося робота - реагировать на отраженный свет, который излучают светодиоды. Включим робота и поставим ладонь перед одним из фотосенсоров. Робот повернет в сторону ладони. Переместим ладонь немного в сторону так, чтобы она скрылась из поля "зрения" одного из фотосенсоров, в ответ робот послушно, как собачка, повернет за ладонью. Светодиоды следует подбирать достаточно яркие, чтобы отраженный свет устойчиво улавливался фототранзисторами. Хороших результатов можно достичь при использовании красных или оранжевых светодиодов с яркостью более 1000 мКд.

Если робот реагирует на вашу руку только тогда, когда она почти касается фотосенсора, то можно попробовать поэкспериментировать с листочком белой бумаги: отражающие способности белого листа намного выше, чем у человеческой руки, и реакция робота на белый листок будет намного лучше и устойчивее.

Проверь себя

Задание для самостоятельного решения

- Белый цвет обладает самыми высокими отражающими свойствами, черный - наименьшими. **Основываясь на этом, можно сделать робота, следующего по линии.** Сенсоры при этом следует расположить так, чтобы они были направлены вниз. Расстояние между сенсорами должно быть немного больше, чем ширина линии. Схема робота, следующего по черной линии, идентична предыдущей. Чтобы робот не терял черную линию, нарисованную на белом поле, ее ширина должна быть около 30 мм или шире. Алгоритм поведения робота достаточно прост. Когда оба фотосенсора улавливают отраженный от белого поля свет, робот движется вперед. Когда один из сенсоров заезжает на черную линию, соответствующий электромотор останавливается и робот начинает поворачиваться, выравнивая свое положение. После того как оба сенсора снова находятся над белым полем, робот продолжает свое движение вперед.

- Управление моторами с использованием платы Arduino и микросхемы L293D

Пример 1. Управление моторами можно осуществлять не только на основе логики микросхемы управления моторами, но и подавать напряжение с отладочной платы программы путем (Рисунок 4).

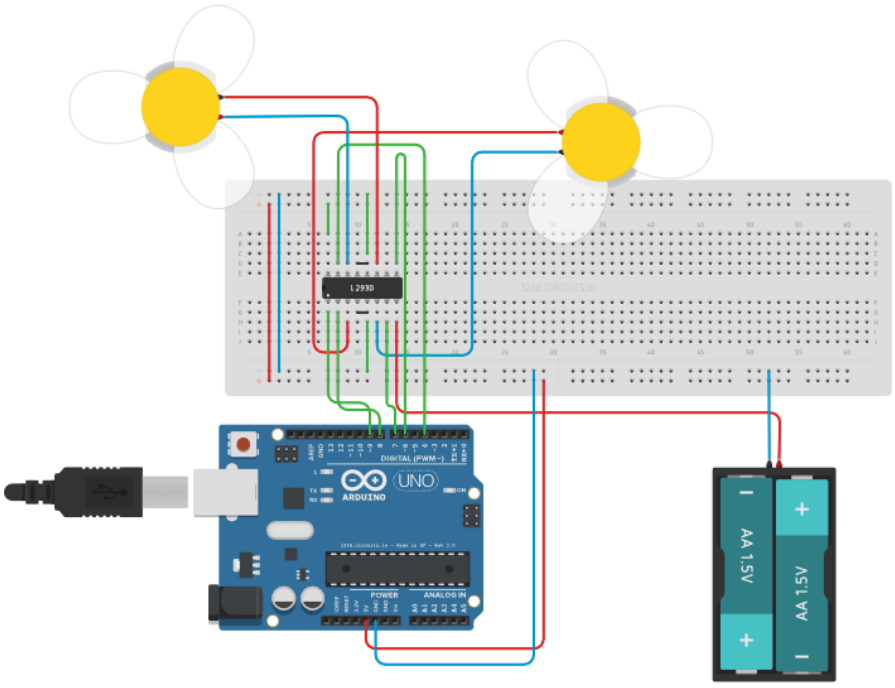


Рисунок 4. Подключение моторов с использованием Arduino и L293D

```
int IN1 = 8; //input1 подключен к выводу 8
int IN2 = 7;
int IN3 = 2;
int IN4 = 4;
int EN1 = 9;
int EN2 = 6;
int i;
void setup()
{
  pinMode (EN1, OUTPUT);
  pinMode (IN1, OUTPUT);
  pinMode (IN2, OUTPUT);
  pinMode (EN2, OUTPUT);
  pinMode (IN4, OUTPUT);
  pinMode (IN3, OUTPUT);
}
void loop()
{
  digitalWrite (IN2, HIGH);
  digitalWrite (IN1, LOW);
  digitalWrite (IN4, HIGH);
  digitalWrite (IN3, LOW);
  for (i = 50; i <= 180; ++i)
  {
     analogWrite(EN1, i);
     analogWrite(EN2, i);
     delay(30);
  }
  analogWrite (EN1, 0);
  analogWrite (EN2, 0);
  delay(500);
  digitalWrite (IN1, HIGH);
  digitalWrite (IN2, LOW);
  digitalWrite (IN3, HIGH);
  digitalWrite (IN4, LOW);
  for (i = 50; i <= 180; ++i)
  {
     analogWrite(EN1, i);
     analogWrite(EN2, i);
     delay(30);
  }
  analogWrite (EN1, 0);
  analogWrite (EN2, 0);
  delay(8000);
}
```

Измените код нашей программы так, чтобы оба мотора вращались одновременно в одну сторону с плавным увеличением скорости, и по достижению максимальной скорости стали замедляться до полной остановки моторов.

Пример 2. «Миксер». Кнопочное управление скоростью моторов (Mosfet-NPN транзистор).

Кроме основных компонентов указанных в начале эксперимента вам понадобится 3 тактовых кнопки, 1 выпрямительный диод и 1 полевой MOSFET-транзистор (Рисунки 5-7).

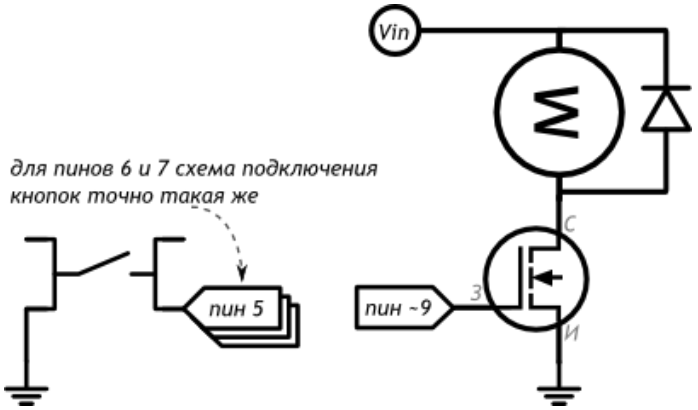


Рисунок 5. Схема подключения «Миксера» к плате Arduino.

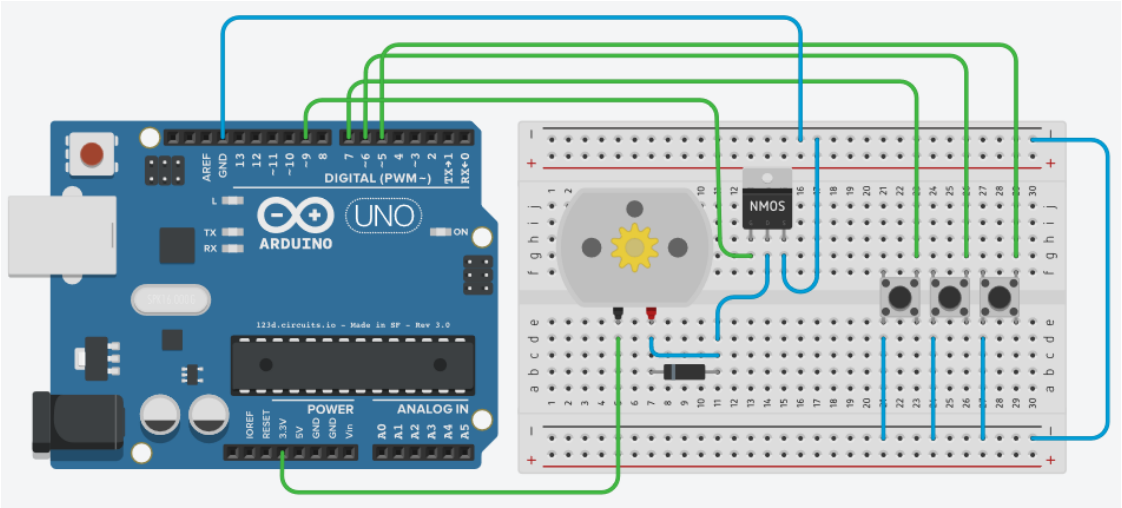


Рисунок 6. Схема в 123D.Circuits

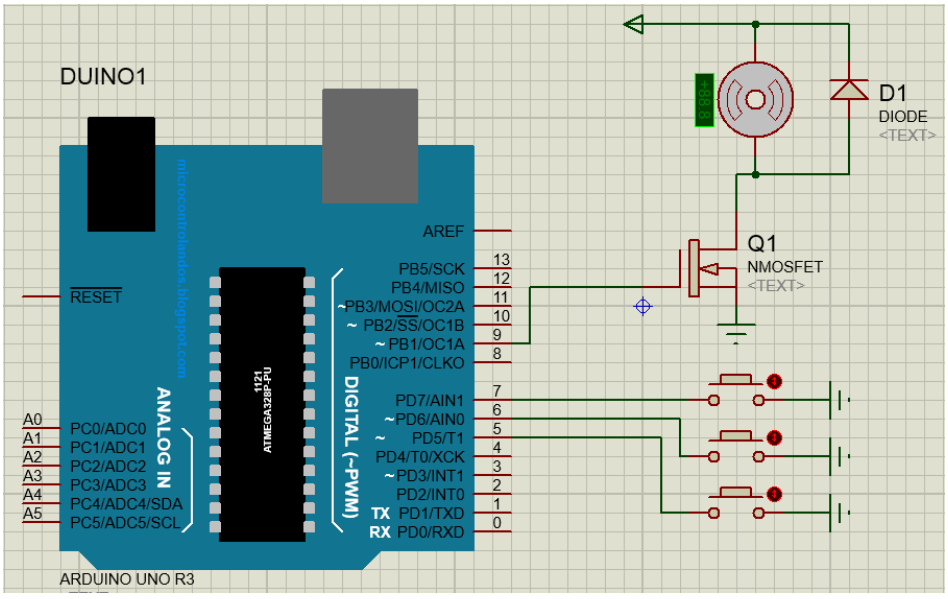


Рисунок 7. Схема в Proteus.

- Защитный диод нам нужен для того, чтобы ток обратного направления, который начнет создавать двигатель, вращаясь по инерции, не вывел из строя транзистор.
- Не перепутайте полярность диода, иначе, открыв транзистор, вы устроите короткое замыкание!
- Причину отсутствия подтягивающих/стягивающих резисторов в схеме вы поймете, ознакомившись с программой.
- Мы подключили питание схемы к выходу Vin платы микроконтроллера, потому что, в отличие выхода 5V, отсюда можно получить напряжение, подключенное к плате, без изменений и без ограничений по величине тока.

```
#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))

void setup()
{
 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. input with pull up)
 for (int i = 0; i < BUTTON_COUNT; ++i)
   pinMode(i + FIRST_BUTTON_PIN, INPUT_PULLUP);
}

void loop()
{
 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
   int speed = 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);
 }
}
```

Пояснения к коду

- Мы использовали новый режим работы портов: INPUT_PULLUP. На цифровых портах Arduino есть встроенные подтягивающие резисторы, которые можно включить указанным образом одновременно с настройкой порта на вход. Именно поэтому мы не использовали резисторы при сборке схемы.
- На каждой итерации цикла мы задаем мотору скорость вращения, пропорциональную текущему значению счетчика. Но выполнение инструкций не дойдет до назначения новой скорости, если при проверке нажатия кнопки она окажется отпущенной. Инструкция continue, которая выполнится в этом случае, отменит продолжение данной итерации цикла и выполнение программы продолжится со следующей. А мотор будет крутиться со скоростью, заданной при последнем нажатии на какую-то из кнопок.

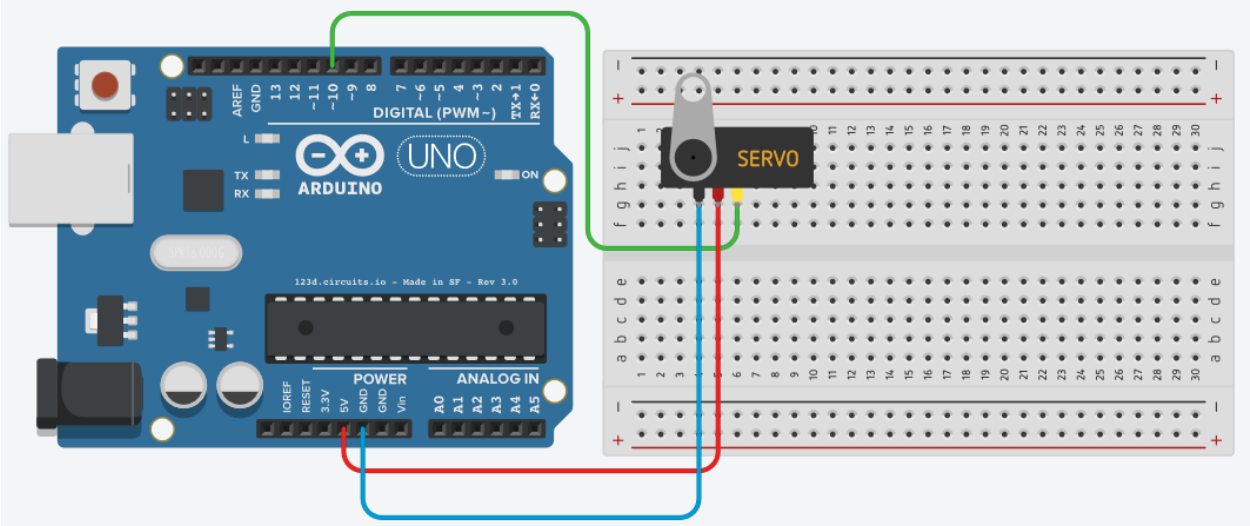
Вопросы для проверки себя

1. Зачем в схеме использован диод?
2. Почему мы использовали полевой MOSFET-транзистор, а не биполярный?
3. Как работает инструкция continue, использованная в цикле for?

Задания для самостоятельного решения

1. Внесите единственное изменение в программу, после которого максимальной скоростью вращения мотора составит половину от возможной.
2. Перепишите программу без использования инструкции continue.
3. Добавьте в схему еще одну кнопку, чтобы у миксера стало три режима. Понадобилось ли изменять что-либо в программе?

Пример 3. «Манипулятор». Использование сервоприводов.



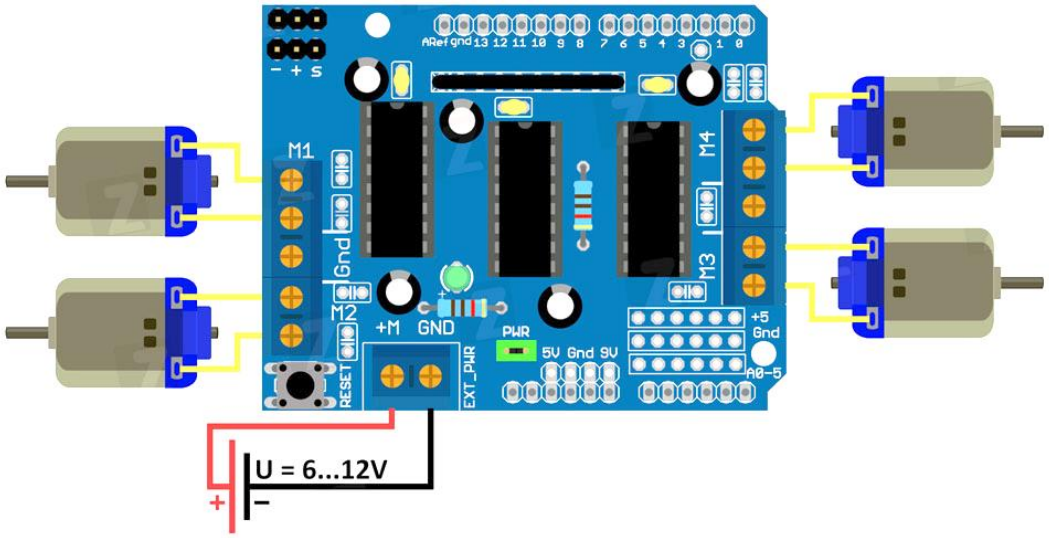

```
#include <Servo.h> //используем библиотеку для работы с сервоприводом
Servo servo; //объявляем переменную servo типа Servo
void setup() //процедура setup

{
servo.attach(10); //привязываем привод к порту 10
}
```

```
void loop() //процедура loop

{
  servo.write(0); //ставим вал под 0
delay(2000); //ждем 2 секунды
servo.write(180); //ставим вал под 180
delay(2000); //ждем 2 секунды
  }
```

Закрепление. Постройте эксперимент таким образом, чтобы несколько сервомоторов симулировали манипулятор захвата. Используйте для эксперимента 4 сервомотора. Один сервомотор для вращения манипулятора на основании, второй для подъема-опускания «роборуки», третий и четвертый для захвата.

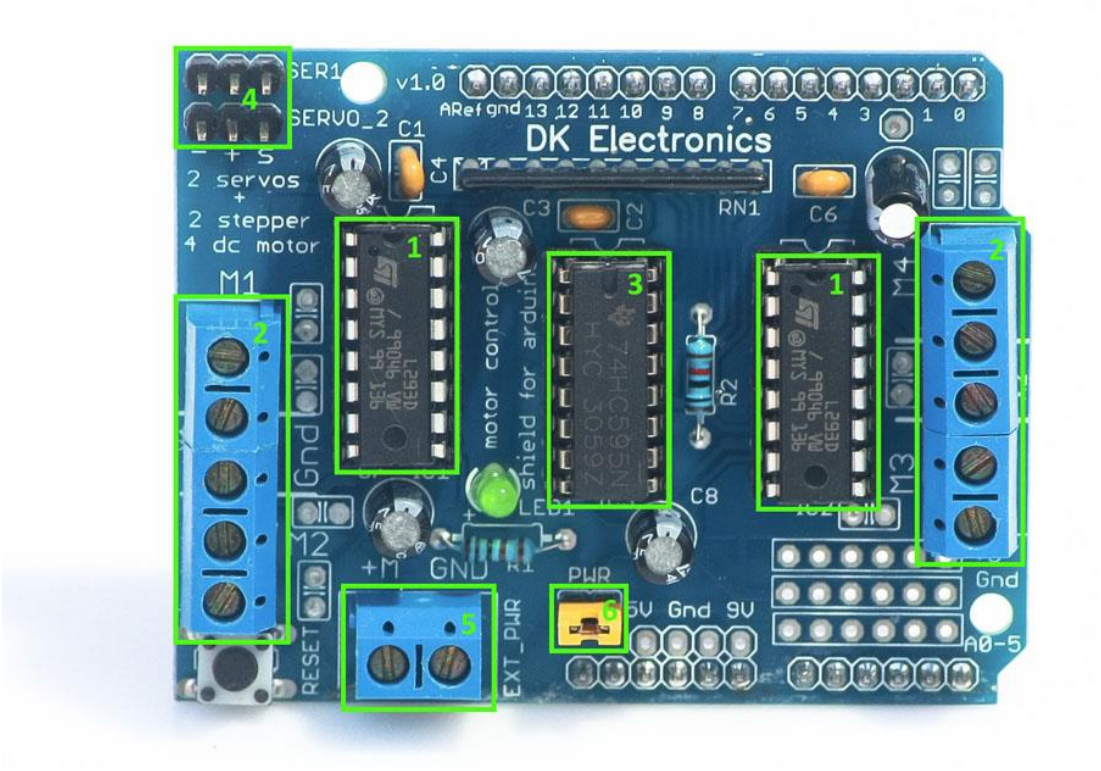
Пример 4. Adafruit motor shield подключение к Arduino.

Рассмотрим устройство Motor Shield'a, разработанного компанией Adafruit, а также научимся управлять с его помощью различными типами двигателей. Данная плата позволит подключить одновременно к Arduino до четырех коллекторных DC двигателей, либо до двух биполярных шаговых двигателей. К тому же на плате имеются разъемы для подключения двух сервоприводов.



На борту данного шилда имеется две микросхемы L293D (1). L-ка позволяет управлятьслаботочными двигателями с током потребления до 600 мА на канал. На двух пятипиновыхклеммниках (2) можно насчитать 4 разъема для подключения двигателей (M1, M2, M3, M4),центральные выводы на пятипиновых клеммниках соединены с землей и служат для удобствапри подключении пятипроводных шаговый двигателей. Использование двух микросхем L293D позволяет одновременно подключить 4 моторчика постоянного тока либо 2 шаговых моторалибо два моторчика и шаговый. Для управления на прямую выводами L-ки (IN1, IN2, IN3, IN4),отвечающимими за выбор направления вращения, необходимо 4 вывода, а для двух микросхемцелых 8. Для уменьшения количества управляющих выводов в игру вступает сдвиговый регистр74HC595 (3). Благодаря регистру управление сводится с 8ми пинов к 4ем.Также, на плату выведены 2 разъема для подключения сервоприводов (4). Управлениесервоприводами стандартное с помощью библиотеки Servo.h и никак не связано с библиотекойкоторую мы будем рассматривать далее.Питание силовой части производится либо от внешнего клеммника (6) либо

замыканиемджампера (5) (питание от клеммника моторов +M соединяется с выводом Vin Arduino). При замкнутом джампере напряжение для объединенного питания должно лежать в пределах от 6 до 12Вольт



К явным минусам данного шилда можно отнести то, что он задействует практически все цифровые пины:

Выводы, отвечающие за скорость вращения двигателей

Цифровой вывод 11- DC Мотор №1 / Шаговый №1

Цифровой вывод 3- DC Мотор №2 / Шаговый №1

Цифровой вывод 5- DC Мотор №3 / Шаговый №2

Цифровой вывод 6- DC Мотор №4 / Шаговый №2

Выводы, отвечающие за выбор направления вращения двигателей:

Цифровые выводы 4, 7, 8 и 12

Выводы для управления сервоприводами (выведены на штырьки на краю платы):

Цифровой вывод 9- Сервопривод №1

Цифровой вывод 10- Сервопривод №2

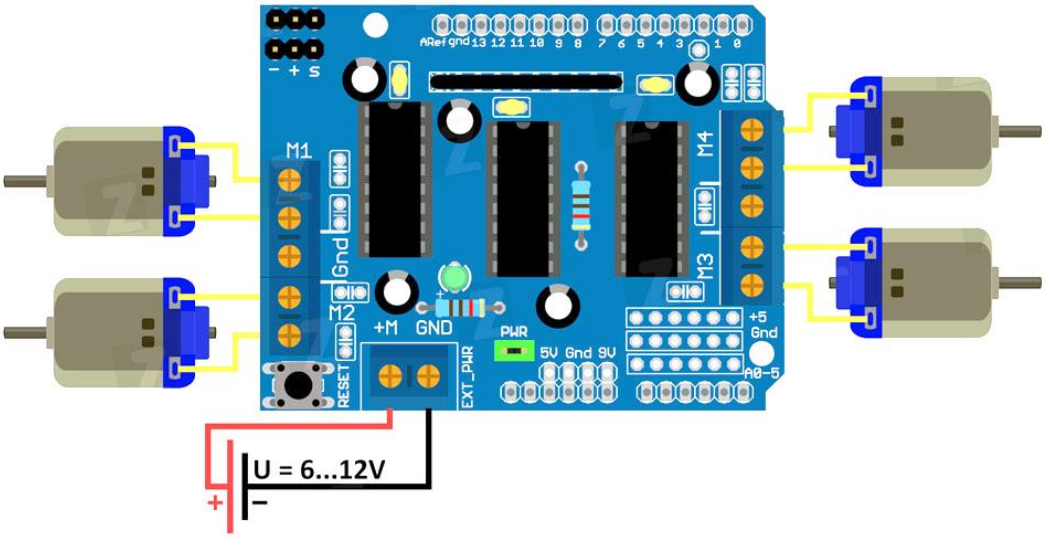
В итоге незадействованными цифровыми выводами остаются только пины 2, 13 и пины интерфейса UART- 0, 1. Однако есть выход из данной ситуации. У нас остались незадействованные аналоговые входы A0-A6, их можно использовать как цифровые. В коде они будут записываться как цифровые с 14 по 19.

Подключение к Arduino

Библиотека необходимая для работы с модулем AFMotor.h. Её необходимо распаковать и добавить в папку "libraries" в папке с Arduino IDE. Не забывайте перезагрузить среду, если на момент добавления IDEшка была открыта.

Втыкаем шилд в плату, подключаем моторы и поехали!

Подключение моторов постоянного тока (4 DC Motors)



// Тестировалось на Arduino IDE 1.0.1

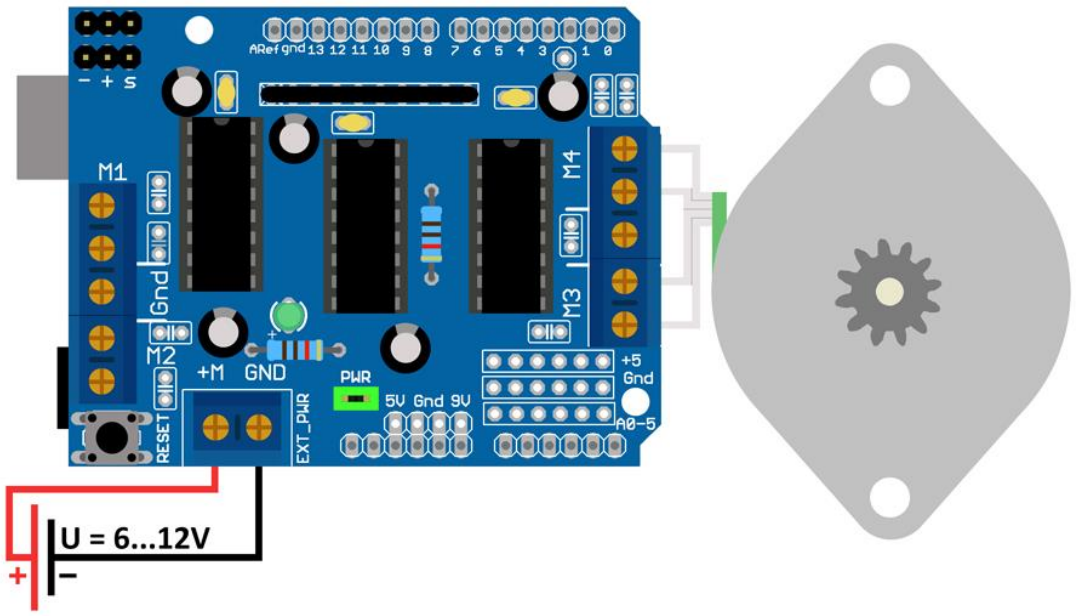
```
#include <AFMotor.h>  // Подключаем библиотеку для работы с шилдом

#include <Servo.h>  // Подключаем библиотеку для работы с сервоприводами, можно не подключать

// Подключаем моторы к клеммникам M1, M2, M3, M4

AF_DCMotor motor1(1);

AF_DCMotor motor2(2);

AF_DCMotor motor3(3);

AF_DCMotor motor4(4);

void setup() {

 // Задаем максимальную скорость вращения моторов (аналог работы PWM)

 motor1.setSpeed(255);

 motor1.run(RELEASE);

 motor2.setSpeed(255);
```

```
  motor2.run(RELEASE);
  motor3.setSpeed(255);
  motor3.run(RELEASE);
  motor4.setSpeed(255);
  motor4.run(RELEASE);
}
int i;
void loop() {
  // Двигаемся условно вперед одну секунду
  motor1.run(FORWARD); // Задаем движение вперед
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
  motor1.setSpeed(255); // Задаем скорость движения
  motor2.setSpeed(255);
  motor3.setSpeed(255);
  motor4.setSpeed(255);
  delay(1000);
    // Останавливаем двигатели
  /* Очень не рекомендуем резко переключать направление вращения двигателей.
  Лучше дать небольшой промежуток времени.*/
    motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
  delay(500);
    // Двигаемся в обратном направлении
  motor1.run(BACKWARD);  // Задаем движение назад
  motor2.run(BACKWARD);
```

```
 motor3.run(BACKWARD);
 motor4.run(BACKWARD);
 motor1.setSpeed(255);  // Задаем скорость движения
 motor2.setSpeed(255);
 motor3.setSpeed(255);
 motor4.setSpeed(255);
 delay(1000);
  // Останавливаем двигатели
 motor1.run(RELEASE);
 motor2.run(RELEASE);
 motor3.run(RELEASE);
 motor4.run(RELEASE);
 delay(500);
  // Разгоняем двигатели в одном направлении
 motor1.run(FORWARD);
 motor2.run(FORWARD);
 motor3.run(FORWARD);
 motor4.run(FORWARD);
 for (i=0; i<255; i++) {
  motor1.setSpeed(i);
  motor2.setSpeed(i);
  motor3.setSpeed(i);
  motor4.setSpeed(i);
  delay(10);
}
 // Останавливаем двигатели
 motor1.run(RELEASE);
 motor2.run(RELEASE);
 motor3.run(RELEASE);
```

```
  motor4.run(RELEASE);
  delay(500);
    // Разгоняем двигатели в обратном направлении
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
  for (i=255; i>=0; i--) {
    motor1.setSpeed(i);
    motor2.setSpeed(i);
    motor3.setSpeed(i);
    motor4.setSpeed(i);
    delay(10);
  }
  // Останавливаем движение
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
  delay(500);
}
```

Пример 5. Подключение шагового мотора (1 Stepper Motor).




```
//Тестировалось на Arduino IDE 1.0.1
#include <AFMotor.h>
#include <Servo.h>
// Подключаем шаговый двигатель к порту 1
// порт 1 - M1, M2
// порт 2 - M3, M4
// 48 - количество шагов для полного оборота
AF_Stepper stepper(48, 1);
void setup() {
}
int i;
void loop() {
  // делаем 48 шагов в одном направлении (FORWARD)
  // DOUBLE - тип шага
  for (i=0; i<48; i++) {
    stepper.step(1, FORWARD, DOUBLE);
    delay(50);
  }
```

```
 // и 48 шагов в обратном направлении (BACKWARD)
 for (i=50; i!=0; i--) {
    stepper.step(1, BACKWARD, DOUBLE);
    delay(50);
 }
}
```