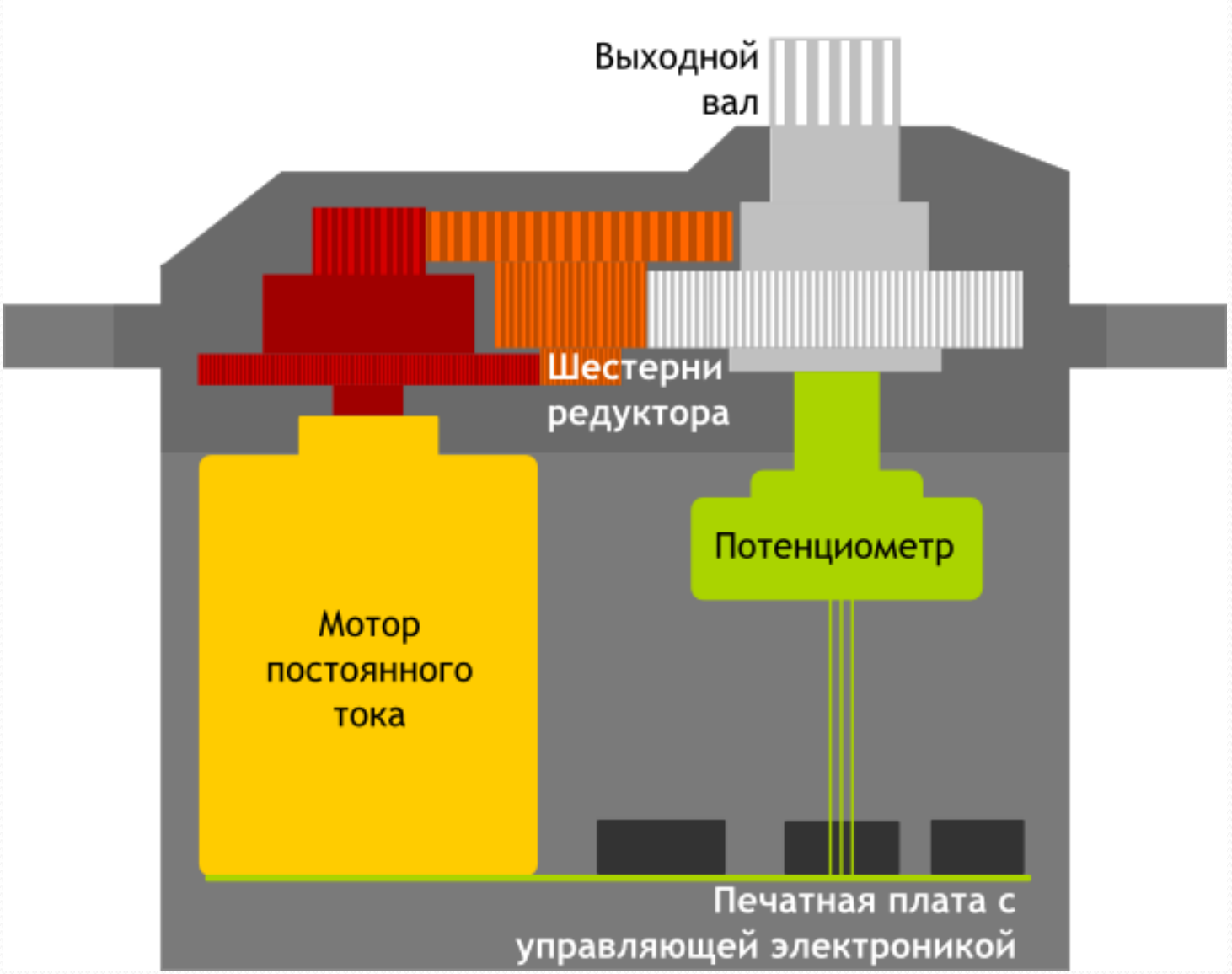
Основы технического волшебства

(Проектная электроника / Электроника для начинающих)

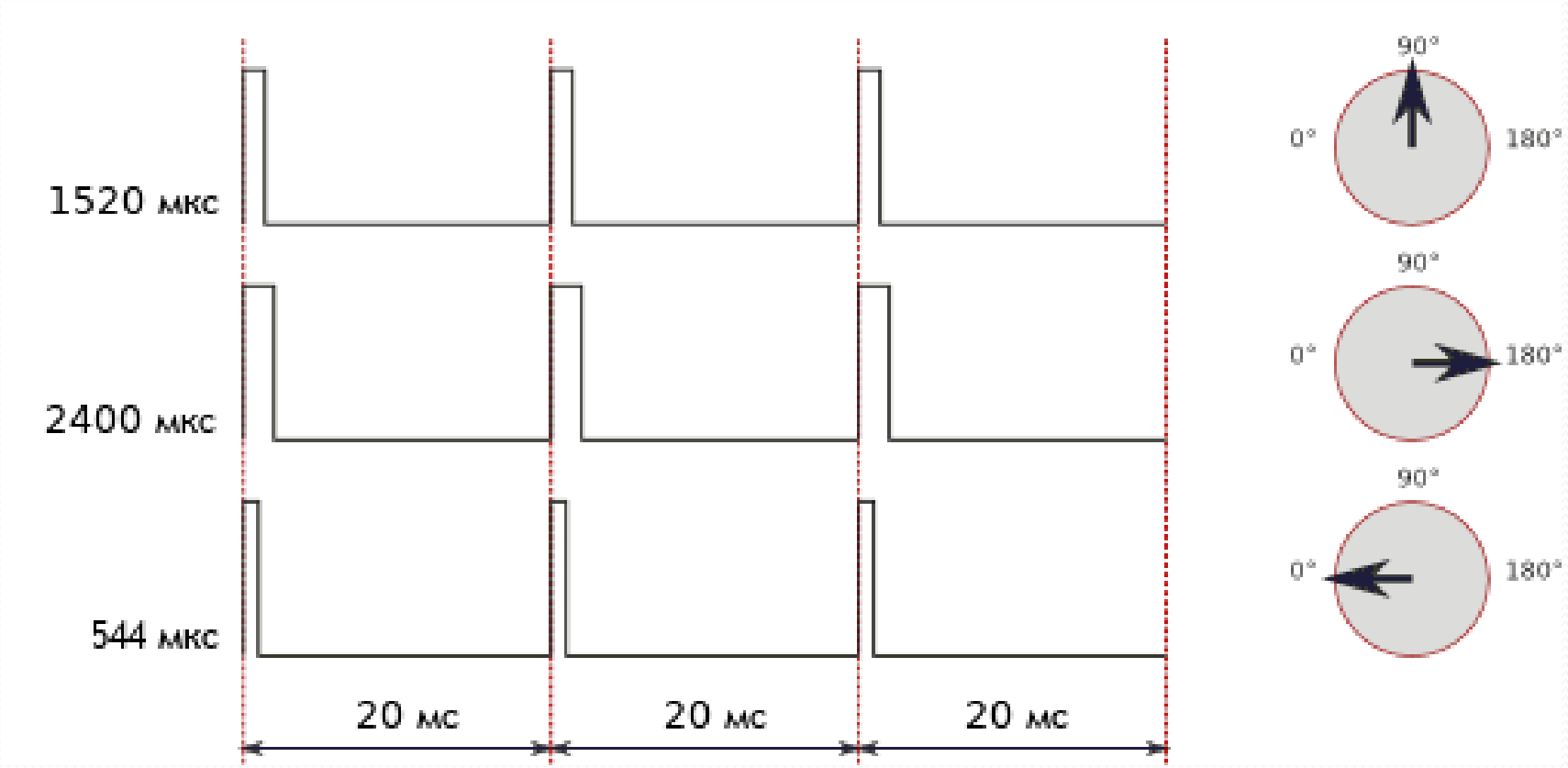
Занятие 13

Сервопривод.

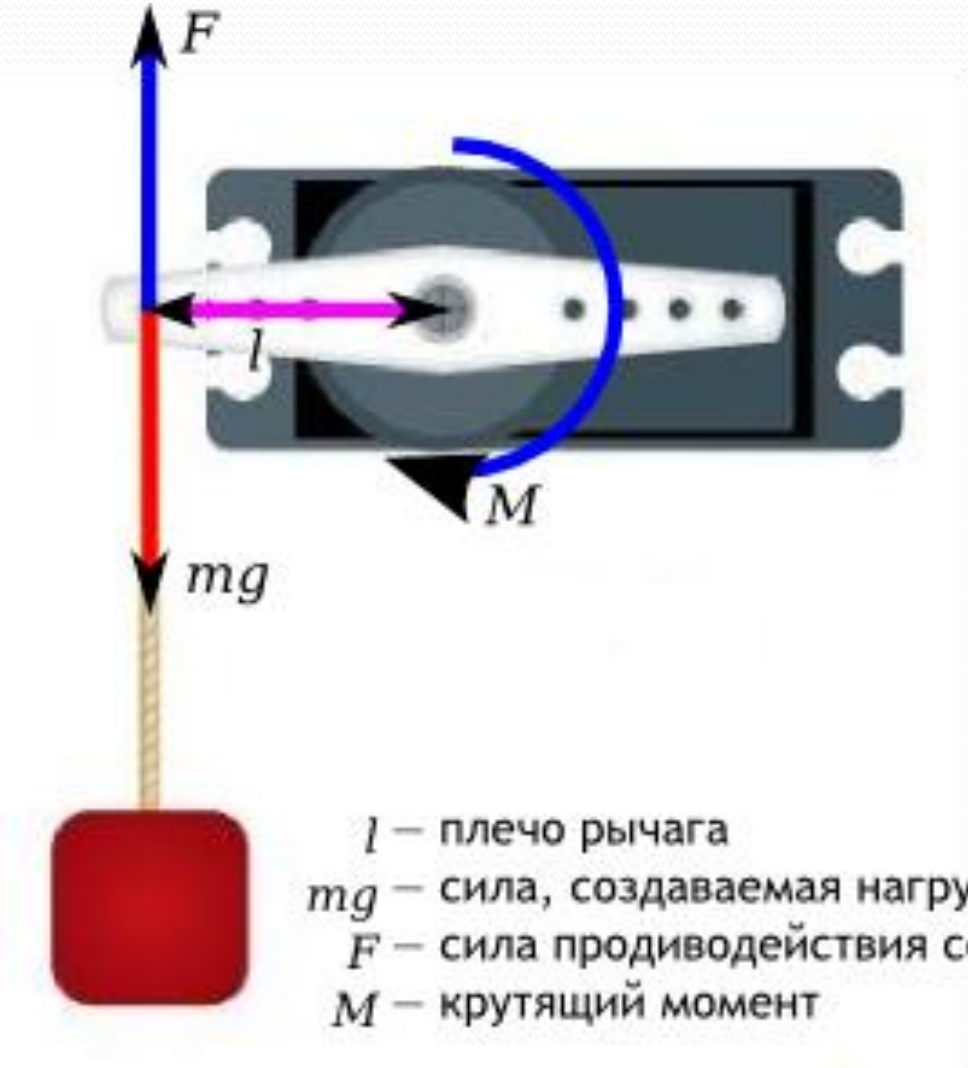
Устройство сервопривода.



Управление сервоприводом.



Характеристики сервоприводов. Крутящий момент и скорость поворота



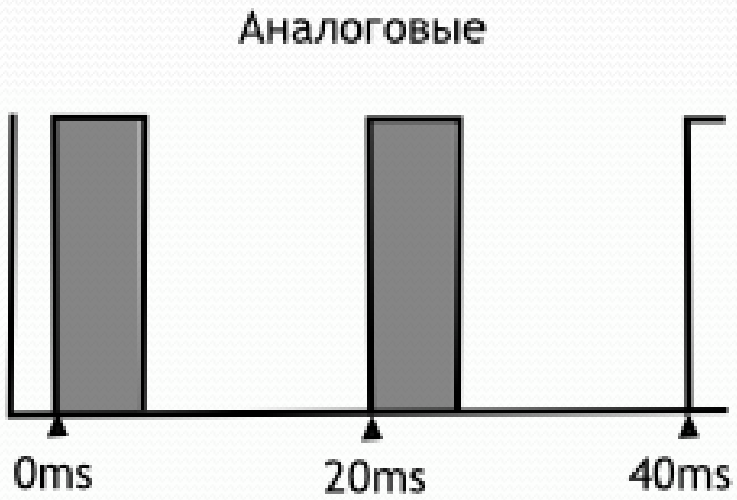
l – плечо рычага

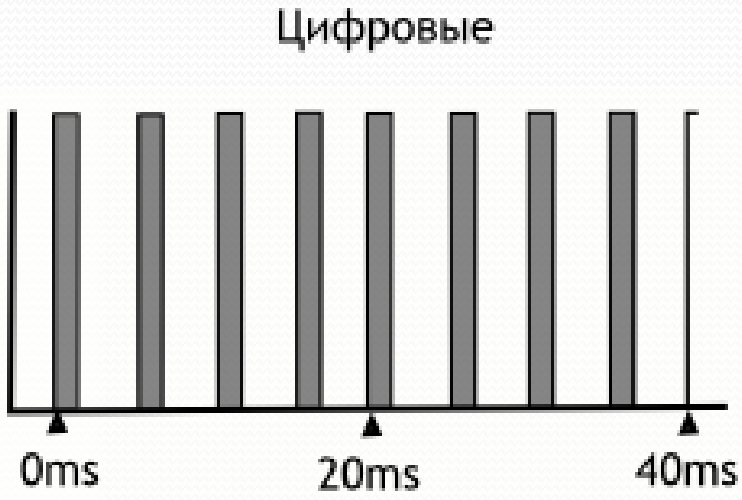
mg – сила, создаваемая нагрузкой

F – сила продиводействия сервопривода

M – крутящий момент

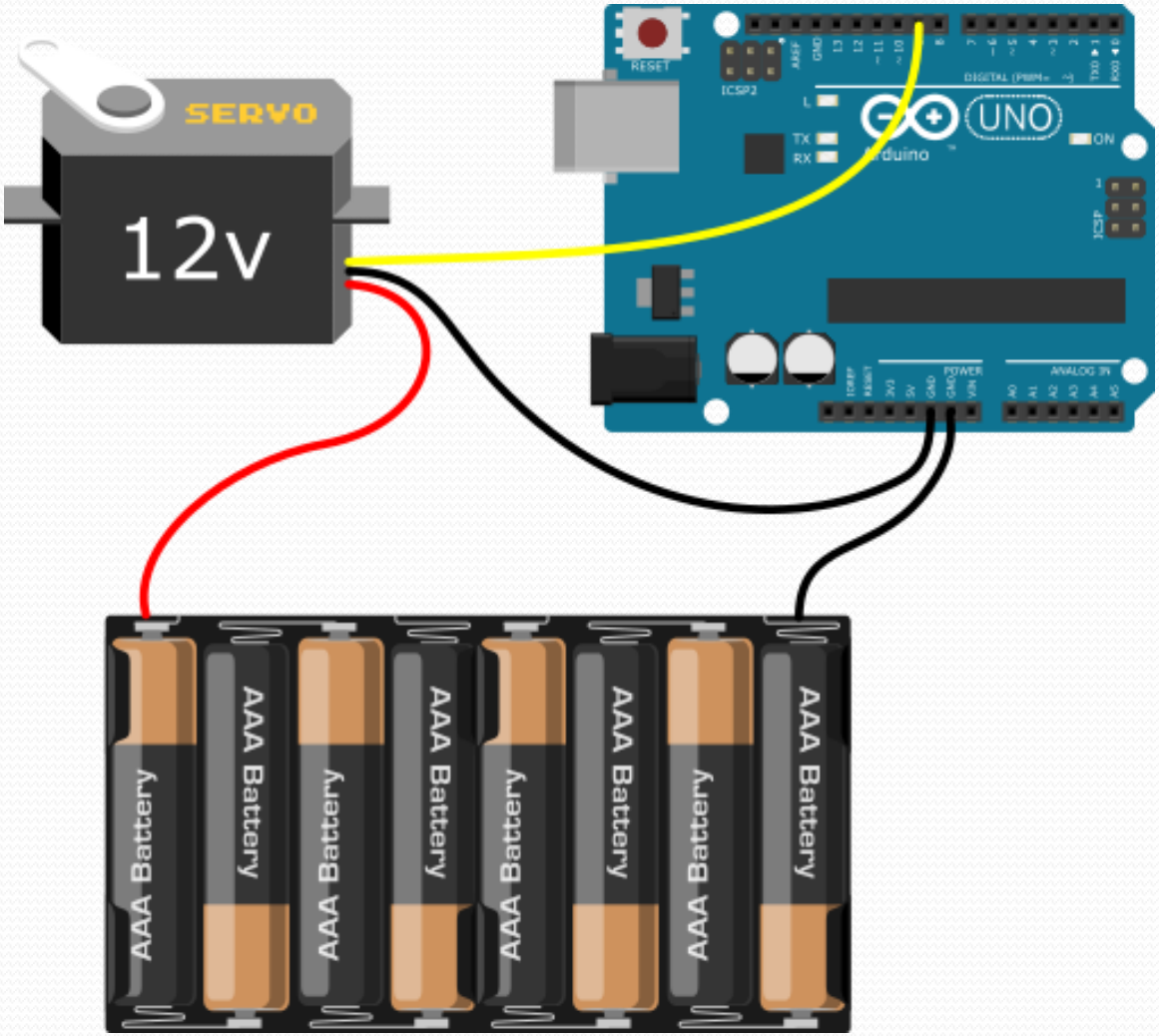
Характеристики сервоприводов. Крутящий момент и скорость поворота





Характеристики сервоприводов. Материалы шестерней

Подключение к Arduino



Управление при помощи библиотеки Servo

Функционал библиотеки Servo

Библиотека `Servo` позволяет осуществлять программное управление сервоприводами. Для этого заводится переменная типа `Servo`. Управление осуществляется следующими функциями:

- `attach()` — присоединяет переменную к конкретному пину. Возможны два варианта синтаксиса для этой функции: `servo.attach(pin)` и `servo.attach(pin, min, max)`. При этом `pin` — номер пина, к которому присоединяют сервопривод, `min` и `max` — длины импульсов в микросекундах, отвечающих за углы поворота 0° и 180°. По умолчанию выставляются равными 544 мкс и 2400 мкс соответственно.
- `write()` — отдаёт команду сервоприводу принять некоторое значение параметра. Синтаксис следующий: `servo.write(angle)`, где `angle` — угол, на который должен повернуться сервопривод.
- `writeMicroseconds()` — отдаёт команду послать на сервоприводимульс определённой длины, является низкоуровневым аналогом предыдущей команды. Синтаксис следующий: `servo.writeMicroseconds(uS)`, где `uS` — длина импульса в микросекундах.
- `read()` — читает текущее значение угла, в котором находится сервопривод. Синтаксис следующий: `servo.read()`, возвращается целое значение от 0 до 180.
- `attached()` — проверка, была ли присоединена переменная к конкретному пину. Синтаксис следующий: `servo.attached()`, возвращается логическая истина, если переменная была присоединена к какому-либо пину, или ложь в обратном случае.
- `detach()` — производит действие, обратное действию `attach()`, то есть отсоединяет переменную от пина, к которому она была приписана. Синтаксис следующий: `servo.detach()`.

Пример управления сервоприводом.

```
// подключаем библиотеку для работы с сервоприводами
#include <Servo.h>
// создаём объект для управления сервоприводом
Servo myservo;

void setup()
{
  // подключаем сервопривод к 9 пину
  myservo.attach(9);
}

void loop()
{
  // устанавливаем сервопривод в серединное положение
  myservo.write(90);
  delay(500);
  // устанавливаем сервопривод в крайнее левое положение
  myservo.write(0);
  delay(500);
  // устанавливаем сервопривод в крайнее правое положение
  myservo.write(180);
  delay(500);
}
```

Сервопривод постоянного вращения.

Функция Arduino	Сервопривод 180°	Сервопривод 360°
Servo.write(0)	Крайне левое положение	Полный ход в одном направлении
Servo.write(90)	Середнее положение	Остановка сервопривода
Servo.write(180)	Крайне правое положение	Полный ход в обратном направлении

Пантограф

Принципиальная схема

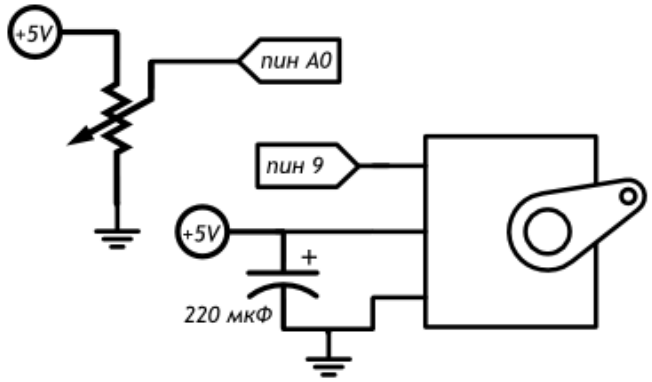
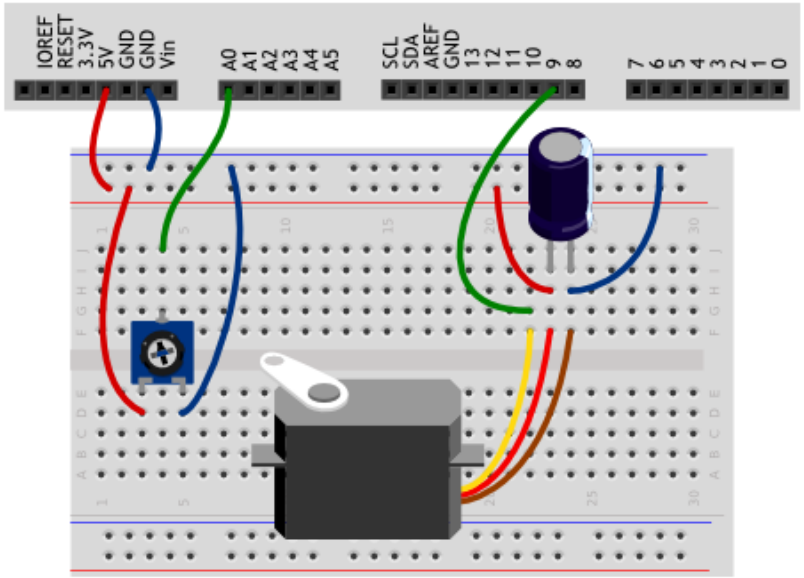


Схема на макетке



В этом эксперименте мы вращаем сервопривод на угол, задаваемый потенциометром.

Пантограф

Скетч:

```
// управлять сервоприводами (англ. servo motor) самостоятельно
// не так то просто, но в стандартной библиотеке уже всё
// заготовлено, что делает задачу тривиальной
#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
Servo myServo;

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

void loop()
{
  int val = analogRead(A0);
  // на основе сигнала понимаем реальный угол поворота движка.
  // Используем вещественные числа в расчётах, но полученный
  // результат округляем обратно до целого числа
  int angle = int(val / 1024.0 * POT_MAX_ANGLE);
  // обычная серва 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);
}
```

Пантограф

Обратите внимание

- Конденсатор в данной схеме нам нужен для того, чтобы при включении сервопривода избежать просадки питания платы.
- Не забывайте про то, что нужно соблюдать полярность элетролитического конденсатора. Короткая ножка (со стороны белой полосы на корпусе) — «минус».
- Вы можете соединить провод сервопривода с макетной платой проводами «папа-папа»: коричневый это земля, красный — питание, оранжевый — сигнал.
- В данном эксперименте мы подключаем питние сервопривода к 5V-выходу Arduino. С одним сервоприводом плата справится, но если в каком-либо проекте вам нужно больше серв, используйте специальные платы-драйвера с отдельным источником питания для серв.

Пантограф

Пояснения к коду

- В данном эксперименте мы также имеем дело с объектом, на этот раз он нужен для простого управления сервоприводом. Как отмечено в комментариях, в отличие от объекта `Serial`, объекты типа `Servo` нам нужно явно создать: `Servo myServo`, предварительно подключив библиотеку `<Servo.h>`.
- Далее мы используем два метода для работы с ним:
  - `myServo.attach(pin)` — сначала «подключаем» серву к порту, с которым физически соединен его сигнальный провод. `pinMode()` не нужна, метод `attach()` займется этим.
  - `myServo.write(angle)` — задаем угол, т.е. позицию, которую должен принять вал сервопривода. Обычно это 0—180°.
- `myServo` здесь это имя объекта, идентификатор, который мы придумываем так же, как названия переменных. Например, если вы хотите управлять двумя захватами, у вас могут быть объекты `leftGrip` и `rightGrip`.
- Мы использовали функцию `int()` для явного преобразования числа с плавающей точкой в целочисленное значение. Она принимает в качестве параметра значение любого типа, а возвращает целое число. Когда в одном выражении мы имеем дело с различными типами данных, нужно позаботиться о том, чтобы не получить непредсказуемый ошибочный результат.