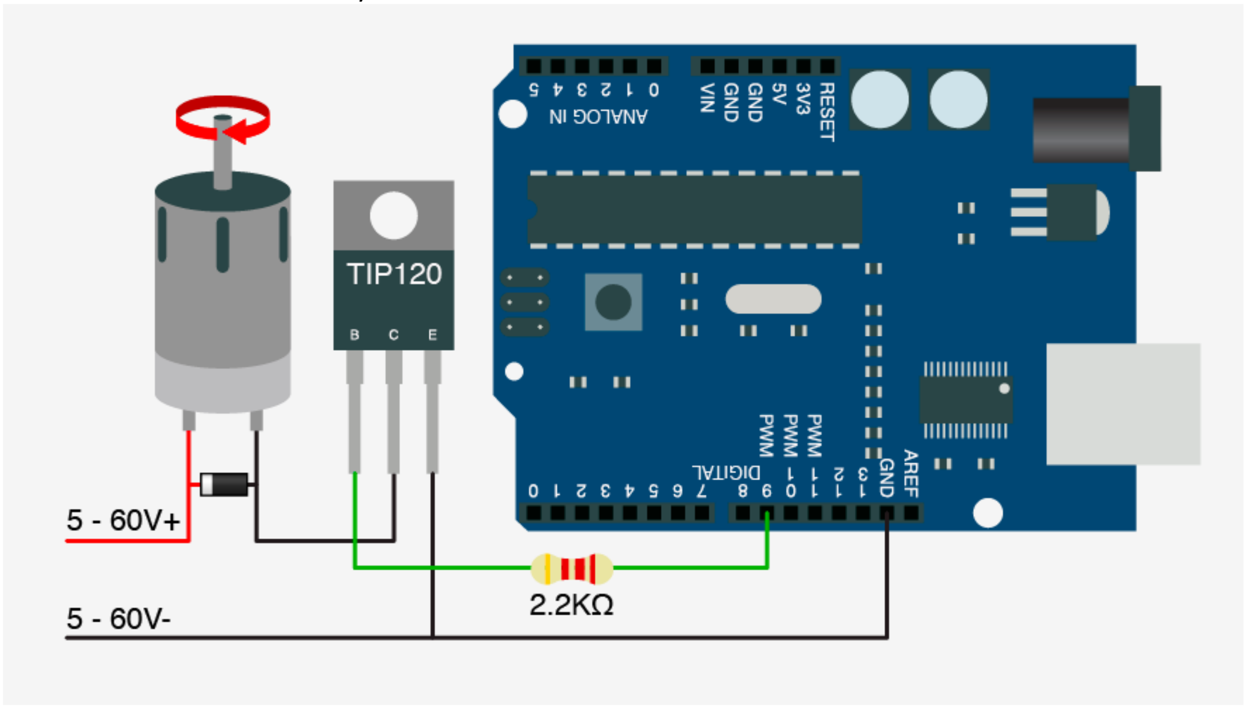
DC MOTOR PWM CONTROL by Arduino



What's the diode used for?
This circuit is pretty simple. This type of **transistor** is switched by **current** and not **voltage**, so we need to make sure to supply the correct current to the base to switch it, so a **resistor** is connected from the Arduino to the base to limit the **current** to the proper amount.
You can see that in 2 of the 3 illustrations, there is a **diode** parallel to the device we are powering. Any time you are powering a device with a **coil**, such as a**relay**, **solenoid**, or **motor**, you need this guy, and don't leave home without it. What happens is when you stop powering the coil, a reverse **voltage**, up to several hundred volts, spikes back. This only lasts a few microseconds, but it is enough to kill our transistor. So this **diode** (only allows current to pass one way) is normally facing the wrong direction and does nothing. But when that voltage spikes comes flowing the opposite direction, the diode allows it to flow back to the coil and not the **transistor**. We will need a diode fast enough to react to the kickback, and strong enough to take the load. A rectifier diode like the **1N4001** or **SB560** should do the job. If you are looking for extra protection you could use an **optoisolator** between the Arduino and the transistor. An optoisolator optically isolates both sides (high and low power) of the circuit so the high-voltage can not possibly come back to the microcontroller.
Just make sure that protection **diode** is facing the correct way (stripe facing the V+ of device). If it is facing the wrong direction, the device you are trying to power will not work as the **diode** will just allow the current to bypass it.

Limitations
Transistors like the **TIP120** are really great for controlling high-power devices from your **microcontroller**, but they do have some limitations. This current configuration is only useful for switching **DC** current, so don't try this with an **AC** source, also transistors have both a **voltage** and an**amperage**/current limitation. The TIP120 can handle switching up to 60V, and the amperage is limited to 5A, or up to 8A pulses of 300μs. I have managed to blow out one of these with a 5A load because of heat. Actually anything over a few amps, especially when the current is constant (like in a motor) and not short pulses, I would recommend using a heat-sink. I usually just solder a bent pice of metal to the back, just something to help dissipate the heat. Just note, if you are using more than one of the TIP120s, you can not solder them to the same heat-sink as the back is connected to the base of the transistor, not the emitter. If you need to switch more than 5A or AC, I would look at using a **relay** instead.

Fade it!
You know the **PWM** outputs on your Arduino? Yeah, the thing that allows you to **analogWrite(pin, value)**. Well, **PWM** is not actually an **analog**output. The Arduino is actually pulsing (very quickly) between 0 and 5v so that the average **voltage** is somewhere in between 0 and 5. Because of this, the**PWM** can be extended through the **transistor** (the transistor can only turn on or off, but can do so very quickly) allowing us to fade lights or control the speed of a motor just like if they were connected directly to the Arduino. All you need to do in order to take advantage of this is make sure the **TIP120**'s base is connected to a **PWM** pin.

```
//Simple code to output a PWM sine wave signal on pin 9
#define fadePin 9

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

void loop(){
  for(int i = 0; i<360; i++){
    //convert 0-360 angle to radian (needed for sin function)
    float rad = DEG_TO_RAD * i;
    //calculate sin of angle as number between 0 and 255
    int sinOut = constrain((sin(rad) * 128) + 128, 0, 255);
    analogWrite(fadePin, sinOut);
    delay(15);
  }
}
```