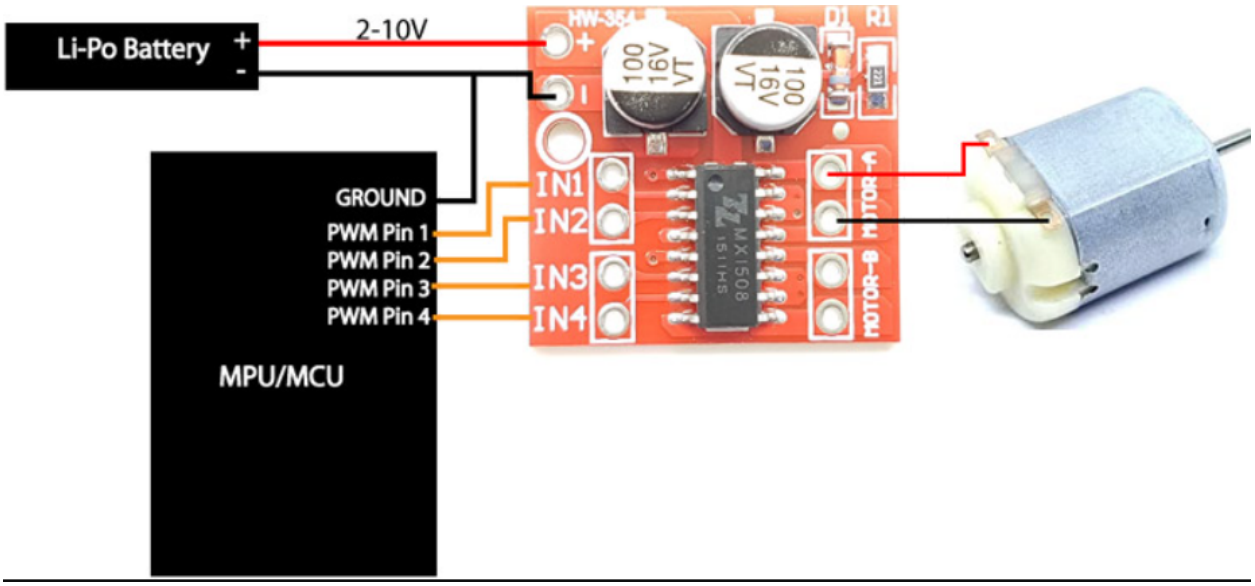
Connecting your motor directly to an arduino is a sweet recipe for disaster. Motors usually draw or request a lot more current than Arduinos can safely, for this reason never directly connect your motor to your Arduino to avoid damage to either or both of them.

To remedy this, motor drivers were developed which interface Arduinos and motors. Think of it like you asking your parents for rent. Your bank account will almost definitely break if you try to pay such an amount to your landlord, but if you tell your parents how much to pay they can safely take the heavy amount of money from wherever (hopefully legally) and pay it to your landlord. That's how the system works, you being the Arduino, your parents being the motor driver, and your landlord being the motor.

Below, is a schematic diagram that shows how to connect a motor through the motor driver to an arduino. We will be using this diagram as a reference.




In our case, the MCU or black box shown above is the Arduino and we are going to connect the motor driver input pins to the Arduino as follows.

| Input pin on the motor driver | Digital Pin on the Arduino |
|---|---|
| IN1 | D6 |
| IN2 | D9 |
| IN3 | D10 |
| IN4 | D11 |
| - | GND |

| | |
|---|---|
| + | VIN |

On the diagram above, we can see a battery is connected directly to the motor driver and not the MCU (Microcontroller which in our case is the Arduino). Always make sure to connect the GND or Ground pin of the motor driver to any available GND pin on the Arduino. **This is called common ground and is necessary for this system to work!** You would want to connect the positive rail of the Battery to the pin on the Arduino that says “VIN” to power the Arduino and the motor driver using the same battery. If not as long as the Arduino is connected to your laptop via USB, and both the motor driver GND and Arduino GND are connected, the system will still work.

Code

The complete code used in this presentation is available on our github at (github.com/McMasterSumobot/Motors).

In this tutorial, we will go over each block of code to explain what each part does.

```
//left motor or wheel
#define IN1 6 //in1 at pin D6
#define IN2 9 //in2 at pin D9

//right motor or wheel
#define IN3 10 //in3 at pin D10
#define IN4 11 //in4 at pin D11
```

Firstly we want to tell the Arduino what pins we are going to be using, what their function is (inputs or outputs, as well as give them a nickname to make programming easier)

By using the “define” function, we tell the Arduino that we are using its pin 6 (written as D6 on the Arduino Nano), and we want to call that pin “IN1” to make our programming easier. We do this because it's easier when writing long lines of code to define a pin using a name that is easy to remember than to write using their actual names or number. So it is easier to write calling this pin IN1 rather than remembering all the time that ‘IN1’ from the motor driver is connected to pin 6 on the Arduino. IN1 is easier to remember than 6 when writing code.

```
void setup() {
  // put your setup code here, to run once:
  pinMode(IN1, OUTPUT);   //here we tell the Arduino IN1 is an OUTPUT pin
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}
```

In this function, we tell the Arduin that these pins are going to be acting as OUTPUT pins. Code in this function runs only once. **Be careful when writing programs, missing a semicolon or bracket could cause your whole program not to work.**

Controlling the motor:
When using the motor driver, we can control the speed and direction of the motor it is connected to, this is done by changing the states of the input pins either HIGH or LOW. The table below shows how this can be done. **Note that this is the same for pins IN3 and IN4.

| Motor Driver Pin | State | Resulting motor movement |
|---|---|---|
| IN1 | HIGH | Clockwise |
| IN2 | LOW | |
| IN1 | LOW | Counter clockwise |
| IN2 | HIGH | |
| IN1 | LOW | No movement / stop |
| IN2 | LOW | |

```
void loop() {
  // put your main code here, to run repeatedly:

  //moves wheel 1 forward
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);

  //moves wheel 2 forward
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);

  delay(5000); //this tells the arduino to wait for 5000 milliseconds or 5 seconds
```

First thing, any piece of code placed in the loop function runs continually without an end.
The function “digitalWrite” is our way of telling the Arduino to change the state of one of its pins. Arduino digital pins have two states, “LOW” and “HIGH”. In these block If codes, we are telling the Arduino to make IN1 & IN3 HIGH, while making IN2 & IN4 LOW. This is case sensitive so be careful.

Using the table above, you have an idea of what direction the motor is going to turn in. The next line includes a delay function. This function accepts any number in its brackets and makes the Arduino wait in whatever state it is in for the specific number of milliseconds told. Example, if you say “Delay(6000);” you are telling the Arduino to wait for six thousand milliseconds which is equivalent to six seconds.

After we tell the Arduino to move forward, we ask the Arduino to wait 5 seconds. In the real world this means that the Arduino will ask the motor driver to make the motors drive forward for 5 seconds.

The rest of the program takes a look at other permutations shown in the table such as stopping the motor and reversing or moving backwards.

For turning left or right we basically want to make one motor move in the opposite direction of the other motor.