10.h bridge

H-bridges are integrated circuits (IC), components that hold large circuits in a tiny package. In the “Motor” Project you got a motor to spin in one direction. If you were to take power and ground on the motor and flip their orientation, the motor would spin in the opposite direction. It’s not very practical to do that everytime you want to spin something in a different direction, so you’ll be using an H-bridge to reverse the polarity of the motor.
When looking at an IC, the part with a dimple is referred to as the top. You can identify pin numbers by counting from the top-left in a “U” direction like in Fig. 1.

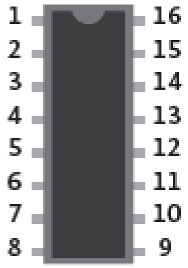
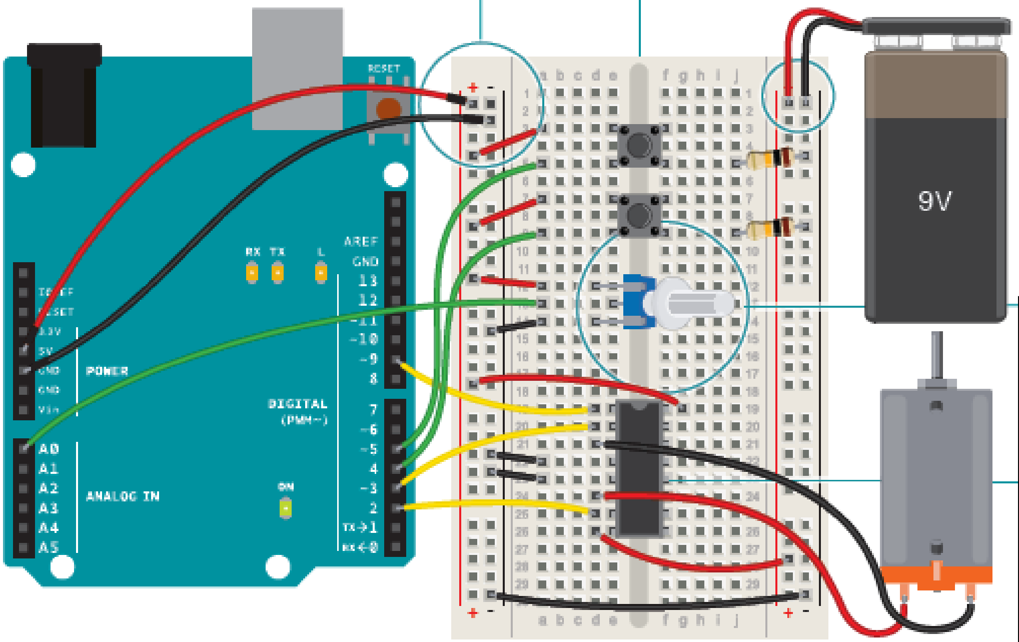


Fig. 1

provide a common ground

Add 2 momentary switches to the breadboard, connecting one side of each to power. Add a 10 kohm pull-down resistor in series with ground on the output pin of both switches. The switch on pin 4 will control direction, the switch on pin 5 will turn the motor on and off.

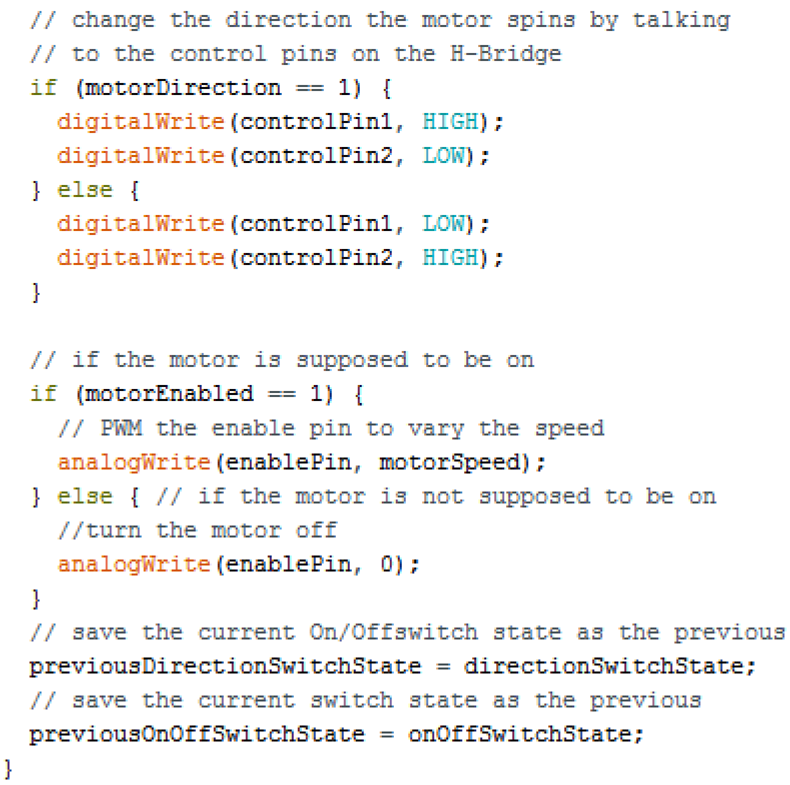


Connect the potentiometer to the breadboard. Wire 5V to one side and ground to the other. Attach the center pin to analog input 0 on the Arduino. This will be used to control the speed of the motor.

Connect pin 1 of the H-bridge to digital pin 9 on the Arduino. Pin 1 is the enable pin. When it receives 5V, it turns the motor on, when it receives 0V, it turns the motor off.
Connect pin 2 on the H-bridge to digital pin 3. Connect pin 7 to digital pin 2. These are the pins you will use to communicate with the H-bridge, telling it which direction to spin. If pin 3 is LOW and pin 2 is HIGH, the motor will spin in one direction. If both the pins are HIGH or LOW at the same time, the motor will stop spinning.
The H-bridge get its power from pin 16, plug that into 5V. Pins 4 and 5 both go to ground. Attach your motor to pins 3 and 6 on the H-bridge. These two pins will switch on and off depending on the signals you send to pins 2 and 7.of the motor.

Plug the battery connector (without the battery attached!) to the other power rails on your breadboard. Connect ground from your Arduino to the battery’s ground. Connect pin 8 from the H-bridge to the battery power. This is the pin that the H-bridge powers the motor from. Make sure you do not have your 9V and 5V power lines connected. They must be separate, only ground should be connected between the two.

```
const int controlPin1 = 2; // connected to pin 7 on the H-bridge
const int controlPin2 = 3; // connected to pin 2 on the H-bridge
const int enablePin = 9;   // connected to pin 1 on the H-bridge
const int directionSwitchPin = 4;  // connected to the switch for direction
const int onOffSwitchStateSwitchPin = 5; // connected to the switch for
                                        //turning the motor on and off
const int potPin = A0;  // connected to the potentiometer's output
// create some variables to hold values from your inputs
int onOffSwitchState = 0;  // current state of the On/Off switch
int previousOnOffSwitchState = 0; // previous position of the on/off switch
int directionSwitchState = 0;  // current state of the direction switch
int previousDirectionSwitchState = 0;  // previous state of the direction switch
int motorEnabled = 0; // Turns the motor on/off
int motorSpeed = 0; // speed of the motor
int motorDirection = 1; // current direction of the motor

void setup() {
  // intialize the inputs and outputs
  pinMode(directionSwitchPin, INPUT);
  pinMode(onOffSwitchStateSwitchPin, INPUT);
  pinMode(controlPin1, OUTPUT);
  pinMode(controlPin2, OUTPUT);
  pinMode(enablePin, OUTPUT);

  // pull the enable pin LOW to start
  digitalWrite(enablePin, LOW);
}
void loop() {
  // read the value of the on/off switch
  onOffSwitchState = digitalRead(onOffSwitchStateSwitchPin);
  delay(1);

  // read the value of the direction switch
  directionSwitchState = digitalRead(directionSwitchPin);

  // read the value of the pot and divide by 4 to get
  // a value that can be used for PWM
  motorSpeed = analogRead(potPin) / 4;

  // if the on/off button changed state since the last loop()
  if (onOffSwitchState != previousOnOffSwitchState) {
    // change the value of motorEnabled if pressed
    if (onOffSwitchState == HIGH) {
      motorEnabled = !motorEnabled;
    }
  }

  // if the direction button changed state since the last loop()
  if (directionSwitchState != previousDirectionSwitchState) {
    // change the value of motorDirection if pressed
    if (directionSwitchState == HIGH) {
      motorDirection = !motorDirection;
    }
  }
```



```
  // change the direction the motor spins by talking
  // to the control pins on the H-Bridge
  if (motorDirection == 1) {
    digitalWrite(controlPin1, HIGH);
    digitalWrite(controlPin2, LOW);
  } else {
    digitalWrite(controlPin1, LOW);
    digitalWrite(controlPin2, HIGH);
  }

  // if the motor is supposed to be on
  if (motorEnabled == 1) {
    // PWM the enable pin to vary the speed
    analogWrite(enablePin, motorSpeed);
  } else { // if the motor is not supposed to be on
    //turn the motor off
    analogWrite(enablePin, 0);
  }
  // save the current On/Offswitch state as the previous
  previousDirectionSwitchState = directionSwitchState;
  // save the current switch state as the previous
  previousOnOffSwitchState = onOffSwitchState;
}
```

Create constants for the output and input pins.

Use variables to hold the values from your inputs. You'll be doing state change detection for both switches, comparing the state from one loop to the next, similar to the Hourglass Project. So, in addition to storing the current state, you'll need to record the previous state of each switch.

motorDirection keeps track of which direction the motor is spinning, and **motorPower** keeps track of whether the motor is spinning or not.

In setup(), set the direction of each input and output pin. Turn the enable pin LOW to start, so the motor isn't spinning right away.

In your loop(), read the state of the On/Off switch and store it in the **onOffSwitchState** variable. If there is a difference between the current switch state and the previous, and the switch is currently HIGH, set the motorPower variable to 1. If it is LOW, set the variable to 0.

Read the values of the direction switch and potentiometer. Store the values in their respective variables.

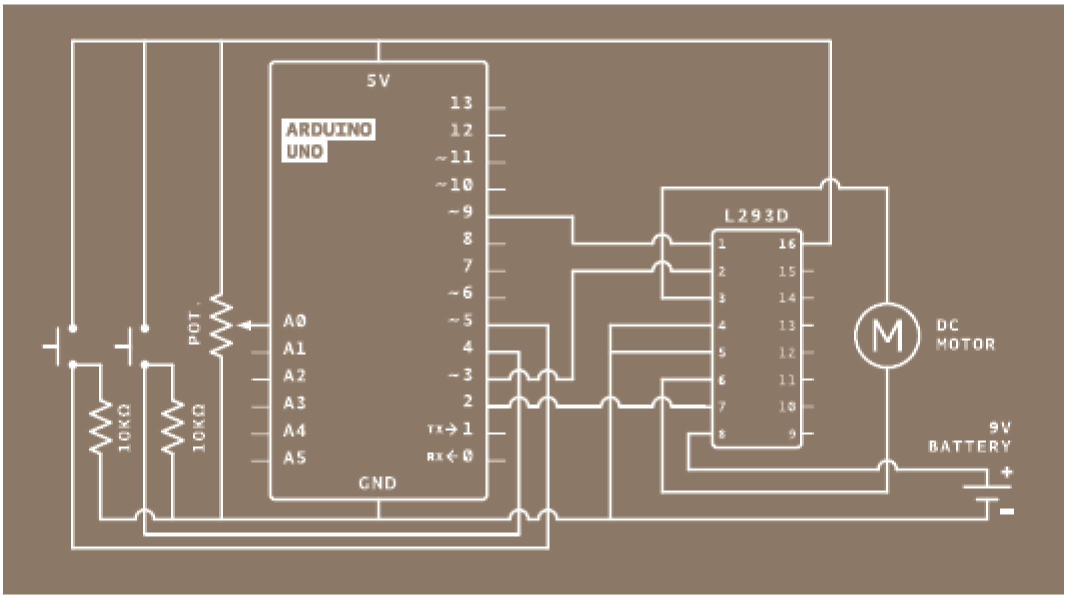
Check to see if the direction switch is currently in a different position than it was previously.If it is different, change the motor direction variable. There are only 2 ways for the motor to spin, so you'll want to alternate the variable between two states. One way to accomplish this is by using the inversion operator like so: **motorDirection =!motorDirection**.

The motorDirection variable determines which direction the motor is turning. To set the direction, you set the control pins setting one HIGH and the other LOW. When motorDirection changes, reverse the states of the control pins.

If the direction switch gets pressed, you'll want to spin the motor in the other direction by reversing the state of the controlPins.

If the motorEnabled variable is 1, set the speed of the motor using analogWrite() to PWM the enable pin. If motorEnabled is 0, then turn the motor off by setting the analogWrite value to 0. Before exiting the loop(), save the current state of the switches as the previous state for the next run through the program.

Plug your Arduino into your computer. Attach the battery to the connector. When you press the On/Off switch, the motor should start spinning. If you turn the potentiometer, it should speed up and slow down. Pressing the On/Off button another time will stop the motor. Try pressing the direction button and verify the motor spins both ways. Also, if you turn the knob on the pot, you should see the motor speed up or slow down depending on the value it is sending.


5V
ARDUINO UNO
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
A0
A1
A2
A3
A4
A5
GND
POT.
10KΩ
10KΩ
L293D
1
2
3
4
5
6
7
8
16
15
14
13
12
11
10
9
M
DC MOTOR
9V BATTERY
+
-