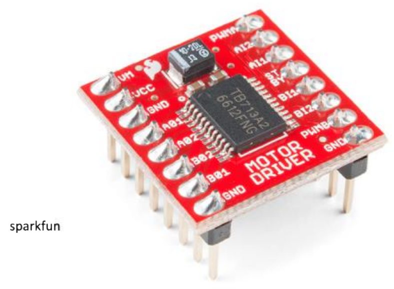
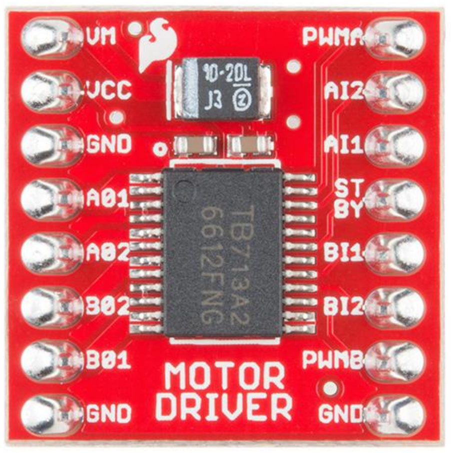
ECE 1001/1002 Introduction to Robotics
Lab #8: Motor Robot

Objectives

Introduction to the motors and H-Bridge

Your kit has two dc geared-motors (with wheels) and a 2-motor switching motor-control (driver) board known as an H-Bridge. These look like:



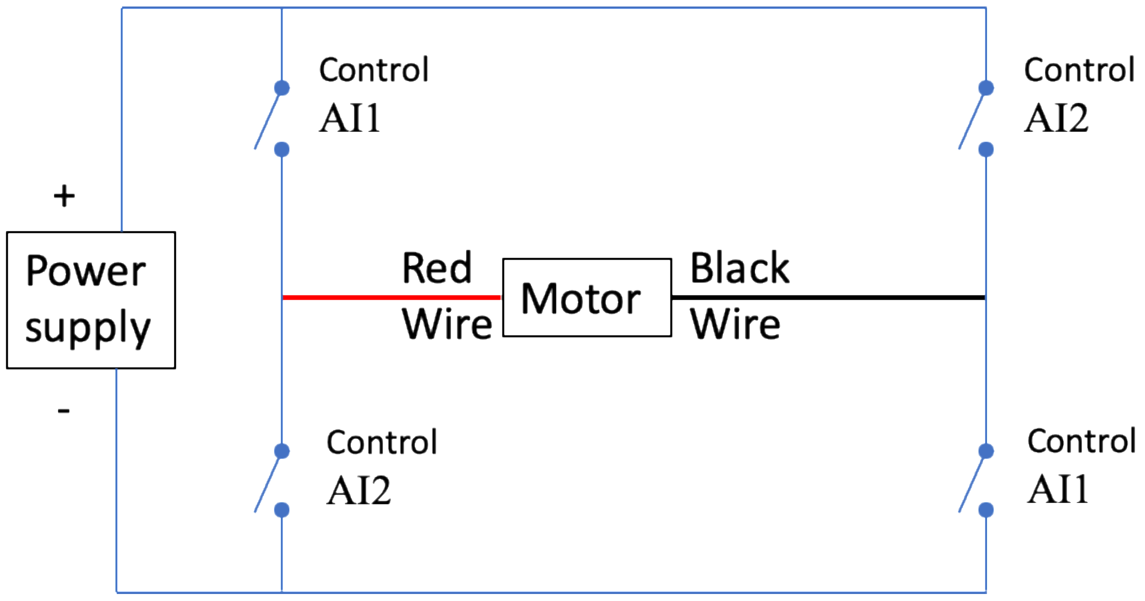


At 4.5V input without a load, the motors will spin at 6720 rpm. The 48:1 transmission lowers that rotation speed to 140 rpm. That would be a maximum speed; under load or with lower voltage the rotational speed is lower. Each motor has two wires, a red and black. The motors are polarized, but confusingly the meaning of red and black is +/- and you can provide a high voltage to – and low voltage to + without hurting the motor, it will simply spin in the opposite direction. The motor speed can be controlled by Pulse Width Modulation (PWM). So by controlling the polarity of the connections to the motor and the PWM level, the motor can be made to turn at a variety of speeds, forward and reverse.

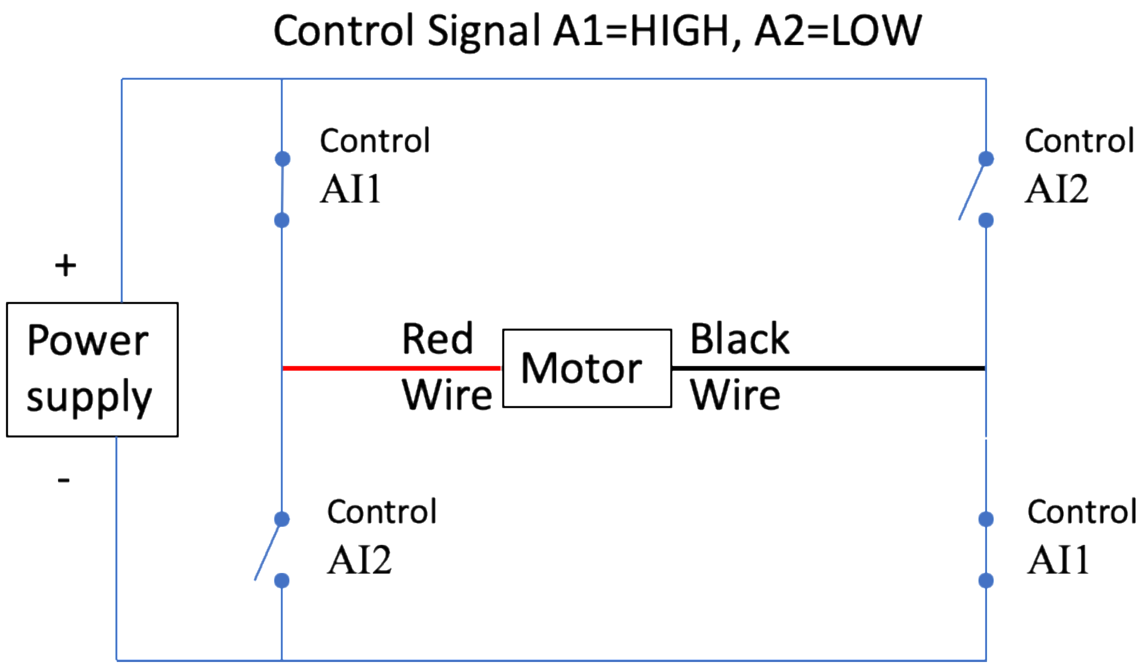
Unfortunately, motors take lots of power, and the Arduino doesn't have the high-power capability to drive the motors directly. The motor-control board, also known as an H-Bridge is built to supply the power the motors need (actually, it's built to supply the current needed, which translates to power since power=voltage*current).

A further issue with motor control is PWM allows control of the power to the wheels, but it can't control the polarity of the voltage supplied—the PWM outputs on the Arduino vary between 0V and 5V. We can switch the polarity of the applied voltage by swapping the red and black wires on the motor, but that doesn't give your robot programmatic control of direction. The H-Bridge is a set of electronic switches which are able to swap the applied voltage between the red and black wires. By changing control voltages in the proper sequence on the H-Bridge, the robot can be made to change direction.

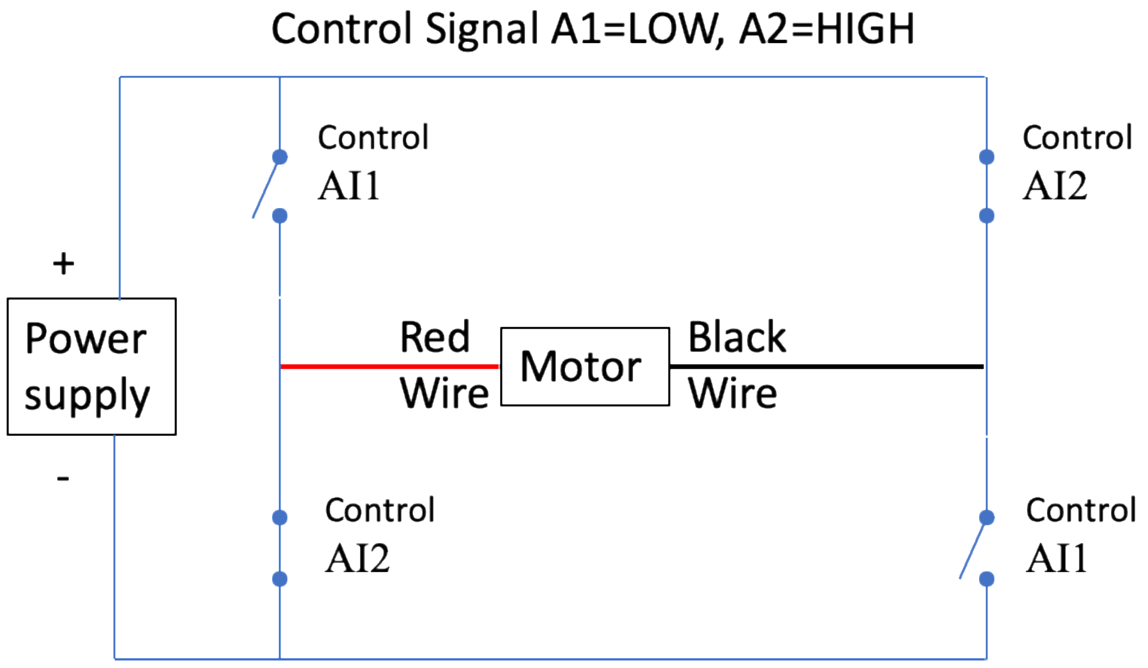
The H-bridge uses transistors as switches to change the voltage polarity, but the H-Bridge control system and equivalent switch position can be understood from mechanical switch diagrams. With all control signals=LOW, all switches would be off (open, disconnected) and no voltage would be applied to the motor, as below. If you ignore the power supply, the arrangement looks like and "H" with the motor in the center.



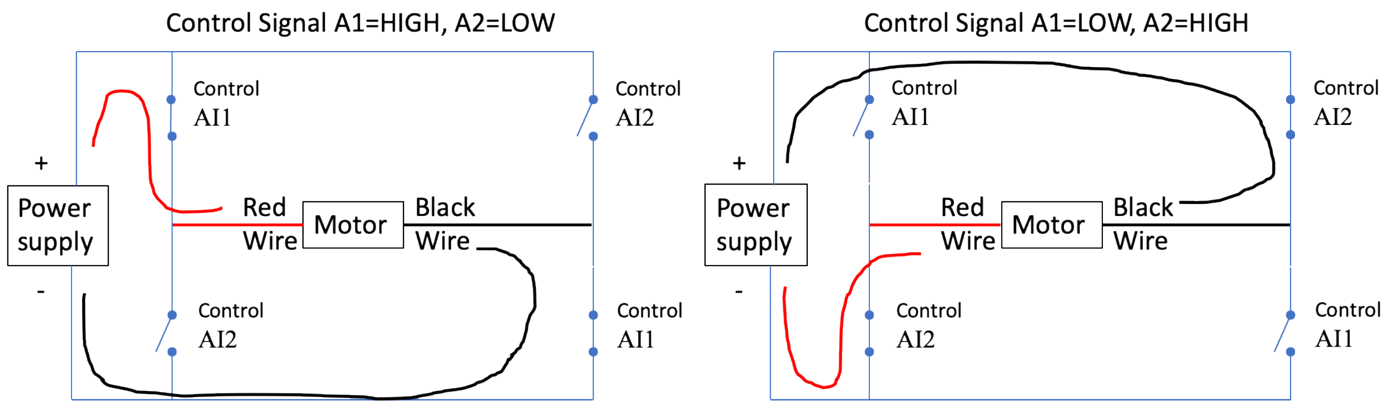
If the control signal A1 is set to HIGH, and A2 set to LOW, the switches would be in connected so that "+" from the power supply is connected to the Red Motor wire, and "-" from the power supply is connected to the Black Motor wire. The motor would be going "forward."



Now by switching the control signals so that signal A1 is set to LOW, and A2 set to HIGH, the switches would be in connected so that "-" from the power supply is connected to the Red Motor wire, and "+" from the power supply is connected to the Black Motor wire. The motor would be going "reverse."



If you unsure about this, trace the wires from the + on the power supply in the two configurations, in one case you arrive at the red motor wire, in the other case you arrive at the black motor wire:

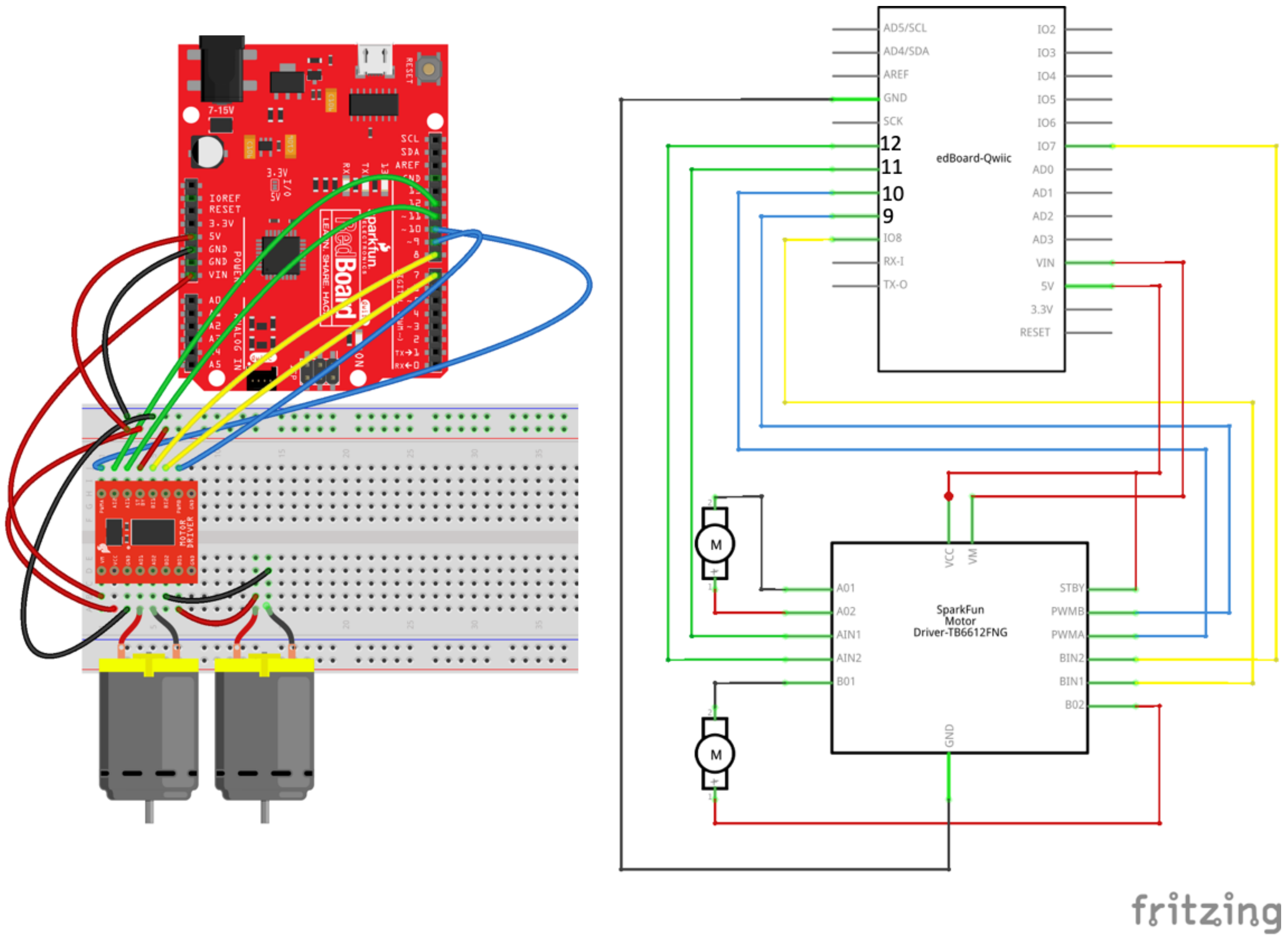


The H-Bridge in your kit can control two motors, so it has two sets of these switches, and control AI1 and AI2 for motor A, and BI1, BI2 for motor B. One of these will be on the left, one on the right, it's up to you.

Requirements and Signoffs

Motor and H-Bridge Installation

Build the circuit as below. Don't attach the wheels to the motors or motors to the robot yet. Double check your wiring



Basic Motor control software

Open the file, Lab8_basic_motor_control.ino, and look at it. At the top, there is. Series of constant integer variable definitions for the pins:

```
const int AIN1 = 11;     // const attribute means program will not allow changes to AIN1
const int AIN2 = 12;
const int PWMA = 10;
const int BIN1 = 8;
const int BIN2 = 7;
const int PWMB = 9;
```

Declaring pins numbers as variables has several advantages:

- It makes it easier to change wiring, and use the program again with a different wiring configuration. Look further down in the loop, and you can see many times each of these variables is used. If these pin assignments are not variables, in order to change the wiring it would be necessary to change all of those pin numbers. This way, it's only necessary to change the pin assignment once at the top (where it is also easy to find, in the case of long complicated programs that is also significant!)

- It makes the code easier to read, as you look down at the main loop, you can see what is happening—it's a kind of self-commented code. It's much easier to understand, trouble-shoot, and reuse.

These integer variables are also made constants, so that they are "read-only" and can't be changed elsewhere in the program. It's a little extra safety measure.

In the setup function, these pin assignments are used to define these motor control pins as outputs.

Now look at the main loop. After a half-second delay, motor A is set to turn "forward" at 30% power.

First it is necessary to set the switches in the H-Bridge, according to the diagram above, AI1 High connects the + to the red motor wire, and the – to the black motor wire. AI2 Low disconnects the other parts of the H-Bridge circuit (of they were not set LOW, there could be damage to some part of the circuit when the power is turned on):

```
digitalWrite(AIN1, HIGH);
 digitalWrite(AIN2, LOW);
```

At this point, the switches are set, but the motor is not turned off. Next the program sets the speed of the motor at 30% power, (PWM setting of 85). That turns on the motor, and it will keep turning until it's turned off. The delay of 500ms lets the motor stay on for half a second:

```
analogWrite(PWMA, 255 / 3);
delay(500);
```

After the motor has been running for half a second, it's turned off by setting the PWM to 0:

```
analogWrite(PWMA, 0);
```

These steps are always needed for running the motors—The H-Bridge switches must be set properly, then a PWM power is set turning on the motor, then it should be left on for some time before it is turned off. In later labs you will learn how to simplify this repetitive programming using functions you can write yourself, but for now it's easier to understand by seeing it all right there in the main loop

Next the other motor is turned on. Then a half-second rest is followed by the motors turning on in the reverse direction at 50% power. The program will repeat and repeat until you unplug the Arduino, of course.

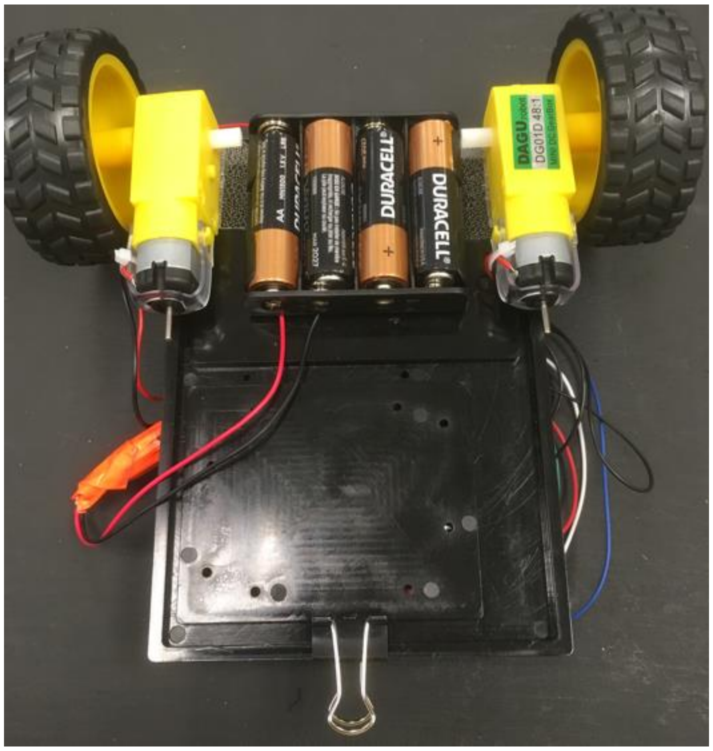
Attach the motors

Now that you have the motors turning, let the robot move. You will need four AA batteries for your battery pack to do this step.

Cut the double-sticky Velcro to attach the battery pack and motors to the bottom of your Arduino system as below. The battery needs to be near the wheels to provide traction. The Wheels can be attached several different ways, but there needs to be sufficient surface area attaching the motors to the baseplate, otherwise you will have trouble making your robot move consistently (and that's a very big problem). Don't attach the wheels so closely that the rub the baseplate, but attach them as closely as possible.

Attach the binder clip to the back side to make your bot a "tail-dragger."

Modelairplanenews.com



Drive the Bot on the Table

With batteries installed and the wheels attached, it's possible for your bot to finally move around on it's own. This opens a dangerous new chapter in robot-building. Be very careful to hold your bot when downloading programs with motors and wheels attached, otherwise it's likely to start moving and drive right off the table, nose-dive to the floor and end up in pieces. Don't let this happen!! You could build also construct a stand to keep the wheels off the table. Eventually there will programmed ways to turn the robot on, but not in this lab.

Open the file, Lab8_Bot_Moves.ino, and look at it. It's very simple and similar to the last program, your bot will move forward for a 0.5 seconds, stop, turn one direction for 0.25 seconds, and then back up for 0.25 seconds. There is a while(TRUE) loop at the end of the program which makes an infinite do-nothing loop, that will cause the main loop to execute only once. If you want it to repeat, push the reset button.

At the table, you will need to plug the batteries in to the Arduino to make it turn on. Don't use a laptop with a USB cable at the table. That's asking for trouble. If your Bot doesn't drive straight, just switch the red and black wires to one of the motors (but not both!)

Give it a try, demonstrate your bot driving around the table for sign-off

Your instructor will either have you print off a sign-off sheet to sign or have a roster to check your name off. To receive credit for this lab, you *must* get the sign-off.