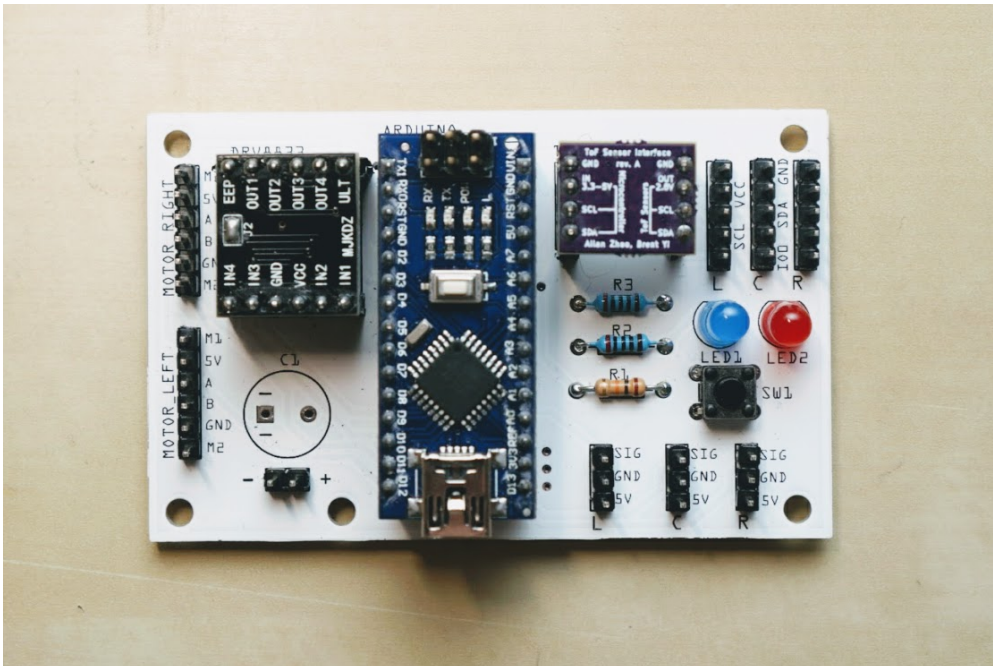
IEEE Micromouse Spring 2018

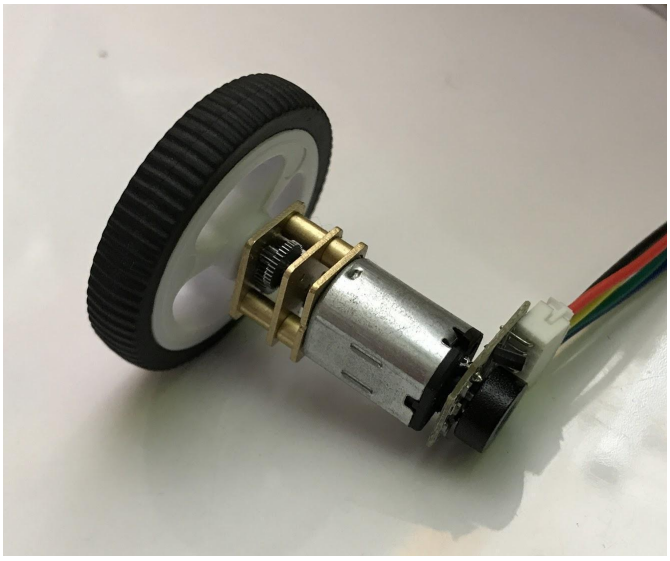
# Lab 2: Brushed Motors & Encoders

# Introduction



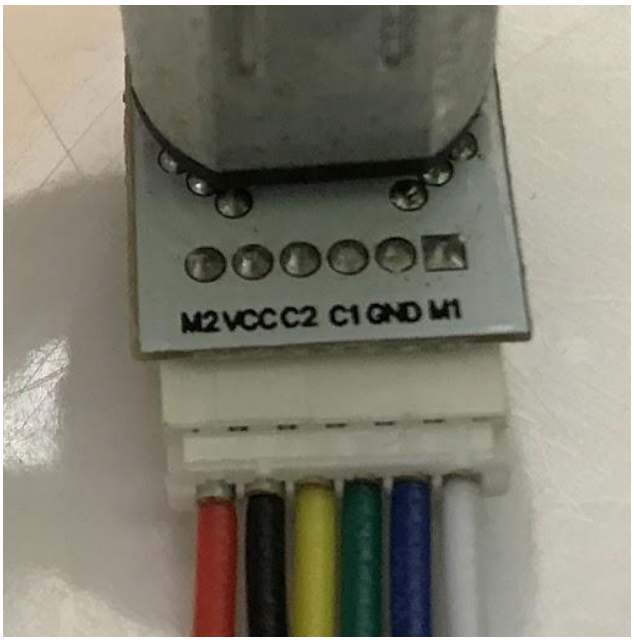

After last week's lab, your PCB should look like the picture above. Hopefully you were able to get the LEDs to blink. This week, you will learn how to make your motors spin.

# Brushed Motor Control



Motor with wheel and cable

You should have a motor, wheel, and a motor cable.

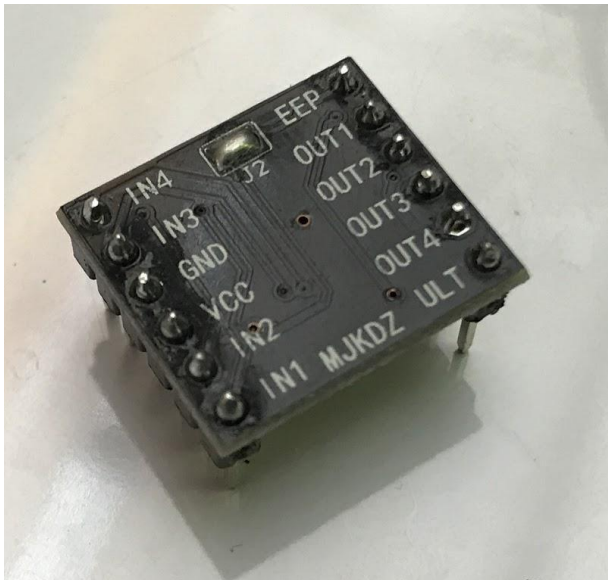



Motor wiring

Two of the wires in the cable connect to the actual motor: M1 and M2. Applying voltage in one direction (M1 > M2) makes the motor spin in one direction, and applying voltage in the other

direction (M1 < M2) makes the motor spin in the other direction. The rest of the wires are connected to the **encoder**, which is a sensor that tells you how quickly the motor is spinning.

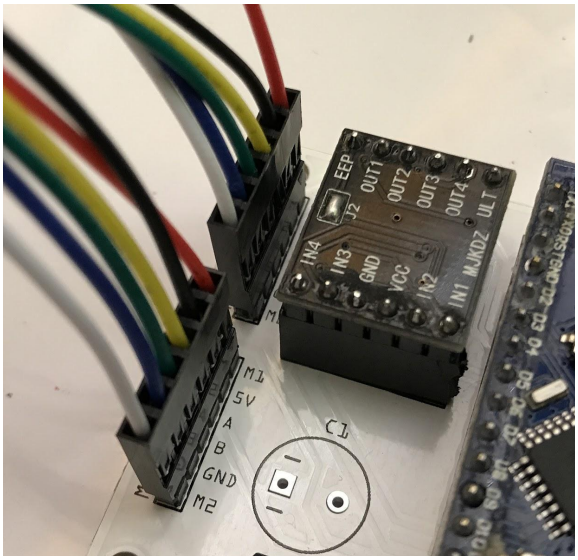
Motors need large amounts of current, which our Arduino can't provide directly. To solve this problem, we use the DRV8833 motor driver chip. The DRV8833 can power two motors.




DRV8833 module

Plug in the DRV8833 module and a motor as shown below. One of the motors is connected to the OUT1 and OUT2 pins on the DRV8833, and the other is connected to the OUT3 and OUT4 pins. The Arduino is connected to the INx pins, allowing us to control the corresponding OUTx pins.

The DRV8833 is powered separately from the Arduino, so you will have to plug in a battery too. Make sure the red wire on the battery cable is connected to the "+" pin.




Motor and DRV8833 connections

You should see a green LED on the DRV8833 module light up, which means it's powered. Now we're ready to spin some motors!

Paste the following code into the Arduino IDE:

```
#define PIN_MOTOR_LEFT_1 9
#define PIN_MOTOR_LEFT_2 10
#define PIN_MOTOR_RIGHT_1 5
#define PIN_MOTOR_RIGHT_2 6

void setup() {
  pinMode(PIN_MOTOR_LEFT_1, OUTPUT);
  pinMode(PIN_MOTOR_LEFT_2, OUTPUT);
  pinMode(PIN_MOTOR_RIGHT_1, OUTPUT);
  pinMode(PIN_MOTOR_RIGHT_2, OUTPUT);
}

void loop() {
  // TODO
}
```

To make the left motor spin, we want to set one of its pins to HIGH and the other pin to LOW. You can use the following code in `loop()`:

```
digitalWrite(PIN_MOTOR_LEFT_1, HIGH);
digitalWrite(PIN_MOTOR_LEFT_2, LOW);
```

Try spinning it in the other direction too. How do we make the motor stop? We just need to set both pins to the same value. It seems like there's two different ways to do it:

```
digitalWrite(PIN_MOTOR_LEFT_1, LOW);
digitalWrite(PIN_MOTOR_LEFT_2, LOW);
```

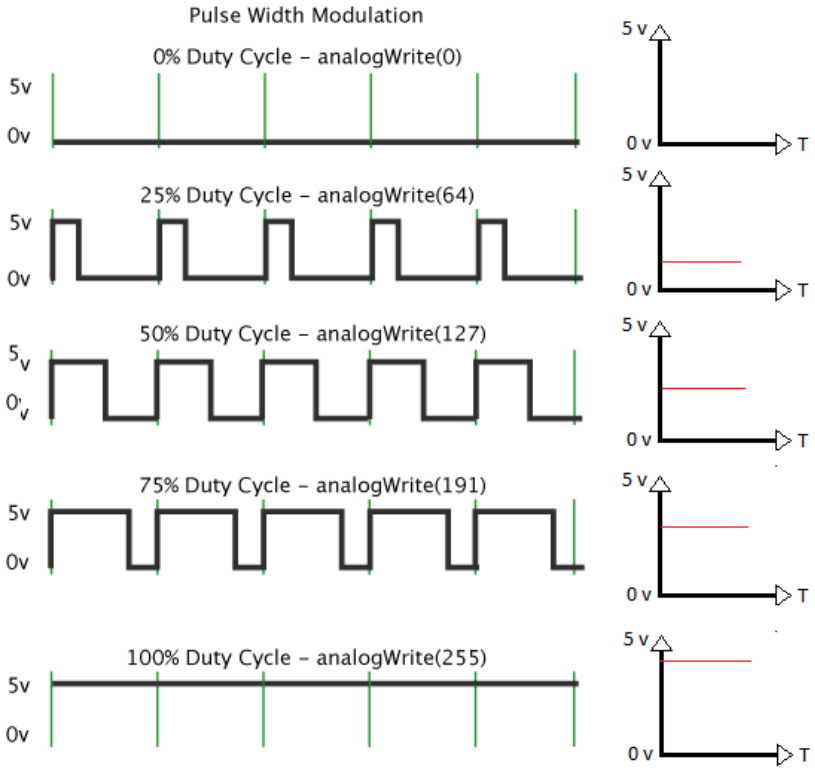
and

```
digitalWrite(PIN_MOTOR_LEFT_1, HIGH);
digitalWrite(PIN_MOTOR_LEFT_2, HIGH);
```

Try both ways. Is there a difference? [1]

So far, we have only been able to make the motor go full speed or not at all. **How can we make the motor spin at a different speed?**
We could try switching the motor on and off really fast, so that the voltage applied to the motor "averages out." This is the basic idea behind pulse width modulation (PWM).

Arduino has a built-in function for doing this: `analogWrite(pin, value)`




[1] One of them "brakes" the motor and the other causes it to "coast." You should be able to feel the difference if you spin the motor by hand.

Try the following code, which should make the motor spin slowly:

```
analogWrite(PIN_MOTOR_LEFT_1, 64); // analogWrite takes a value from 0-255
analogWrite(PIN_MOTOR_LEFT_2, 0);
```

**Checkoff #1:**

1. What's the difference between setting both motor pins to `HIGH` and setting both of them to `LOW`?
2. Make your motor change speed or direction when the button is pressed.[2]
    a. How do you change the speed of the motor?
    b. How do you flip the direction of the motor?
    c. What happens when you analogWrite the same value to both motor driver pins? Why?

[2] See Lab #1 for notes on reading the button state.

# Tracking Wheel Movement

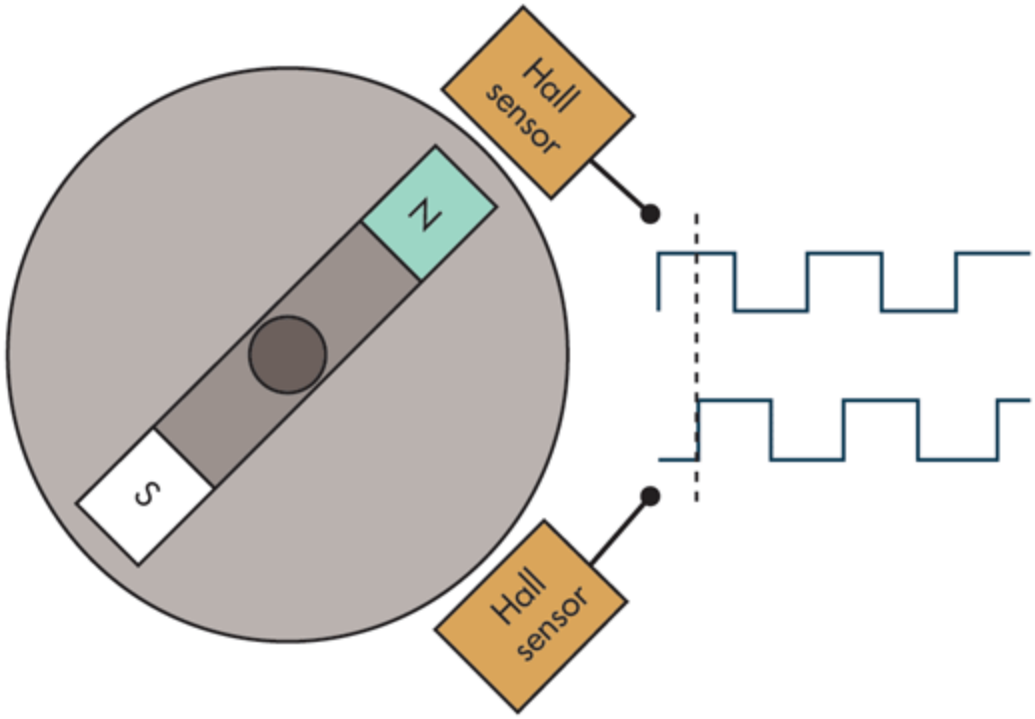
When we're trying to navigate a maze, we'd also like to answer a few motor-related questions:

- How quickly is our robot driving?
- Is it driving straight?
- How far has it driven?

To answer these questions, we'll use the **quadrature encoder** sensors mounted on the back of your motors.

These are comprised of a spinning magnet and two Hall effect sensors.
In the case below, the Hall sensors output:

- **HIGH** when they're closer to the **north** pole of a magnet
- **LOW** when they're closer to the **south** pole of the magnet




The two waveforms next to the picture plot the signal (y-axis) from our Hall sensors with respect to time (x-axis).

Let's try reading from an encoder!

With your left motor plugged in, try viewing the output of following code in the **Serial Plotter**:

```
#define PIN_ENCODER_LEFT_A 11
#define PIN_ENCODER_LEFT_B 2

void setup() {
  pinMode(PIN_ENCODER_LEFT_A, INPUT);
  pinMode(PIN_ENCODER_LEFT_B, INPUT);

  Serial.begin(9600);
}

void loop() {
  Serial.println(digitalRead(PIN_ENCODER_LEFT_A));
}
```

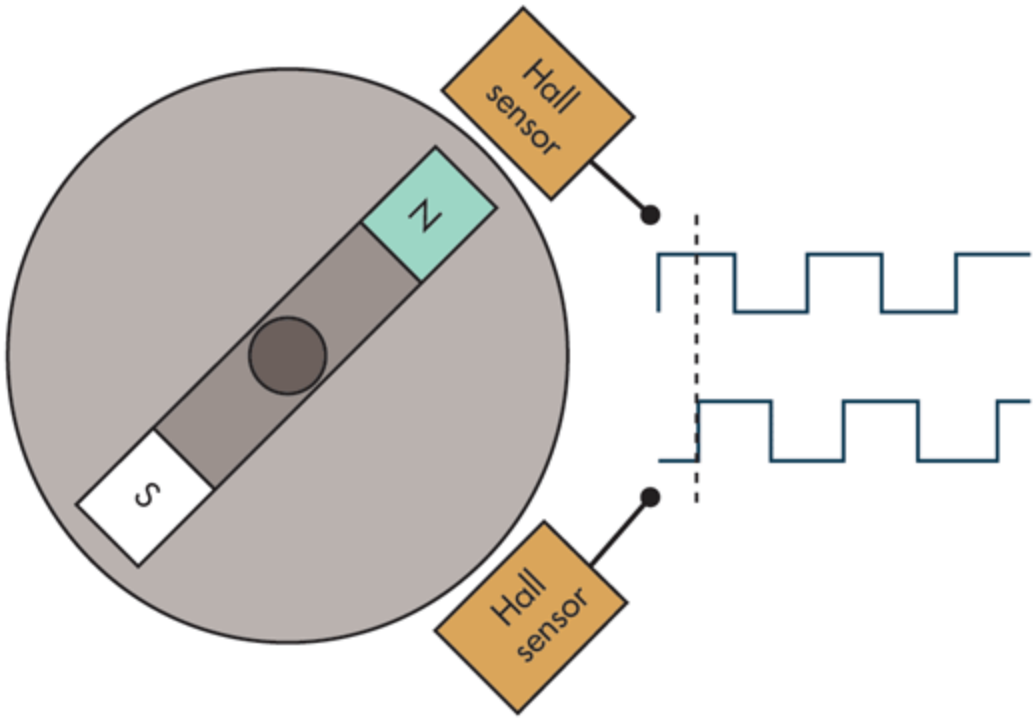
Spin the motor by hand, at various speeds. Note that by reading just one input, it's impossible to distinguish between clockwise and counterclockwise rotations.

**Checkoff #2**

1. What do you see in the Serial monitor when the motor is stationary?
2. What happens to the output when you spin the motor faster? Why?
3. Why do we need two hall effect sensors per motor? What would be wrong if we only had one?
4. Refer to the two waveforms in the graphic above. Which direction is the motor spinning? How do you know?

# Polling

To calculate how far our micromouse has travelled, we can count the number of “rising edges” we see from our encoder output.
One way to accomplish this is by “polling”, or repeatedly reading a digital input that the encoder is attached to and actively counting these edges.




**Try tracking the position of your left wheel.** Here’s some code to get you started!

```
#define PIN_ENCODER_LEFT_A 11
#define PIN_ENCODER_LEFT_B 2

int left_position = 0;

bool prevLeftEncoderVal = 0;

void setup() {
  pinMode(PIN_ENCODER_LEFT_A, INPUT);
  pinMode(PIN_ENCODER_LEFT_B, INPUT);

  Serial.begin(9600);
}
```

```
int count = 0;
void loop() {
  bool leftEncoderVal = digitalRead(PIN_ENCODER_LEFT_B);

  // Detect a rising edge of the left encoder's B channel
  if (leftEncoderVal == HIGH and prevLeftEncoderVal == LOW) {
     leftEncoderRisingEdge();
  }

  prevLeftEncoderVal = leftEncoderVal;

  // Print the position every 1000 loops
  if (count % 1000 == 0) {
     Serial.println(left_position);
  }
  count++;
}

void leftEncoderRisingEdge()
{
  // TODO: increment or decrement left_position,
  //       based on the direction of rotation
}
```

Once you've filled out the `leftEncoderRisingEdge` function, your Arduino should be able to start accurately tracking the position of your left motor!

**Checkoff #3**
Use the **Tools > Serial Plotter** utility to view the position of your motor as you turn it.

1. Roughly how many ticks correspond to a revolution? How do you think this is computed?
2. What issues might the "polling" method of counting encoder ticks have?
3. Now that we've successfully measured the wheel's position, one logical next step might be to measure its velocity.
    a. How might we do this?