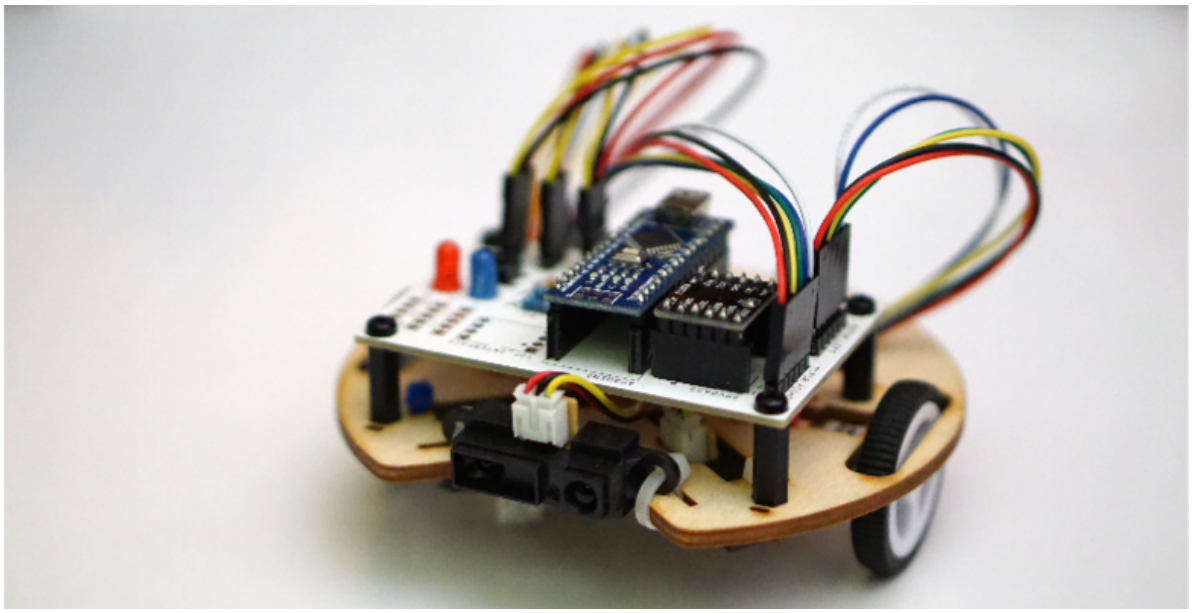
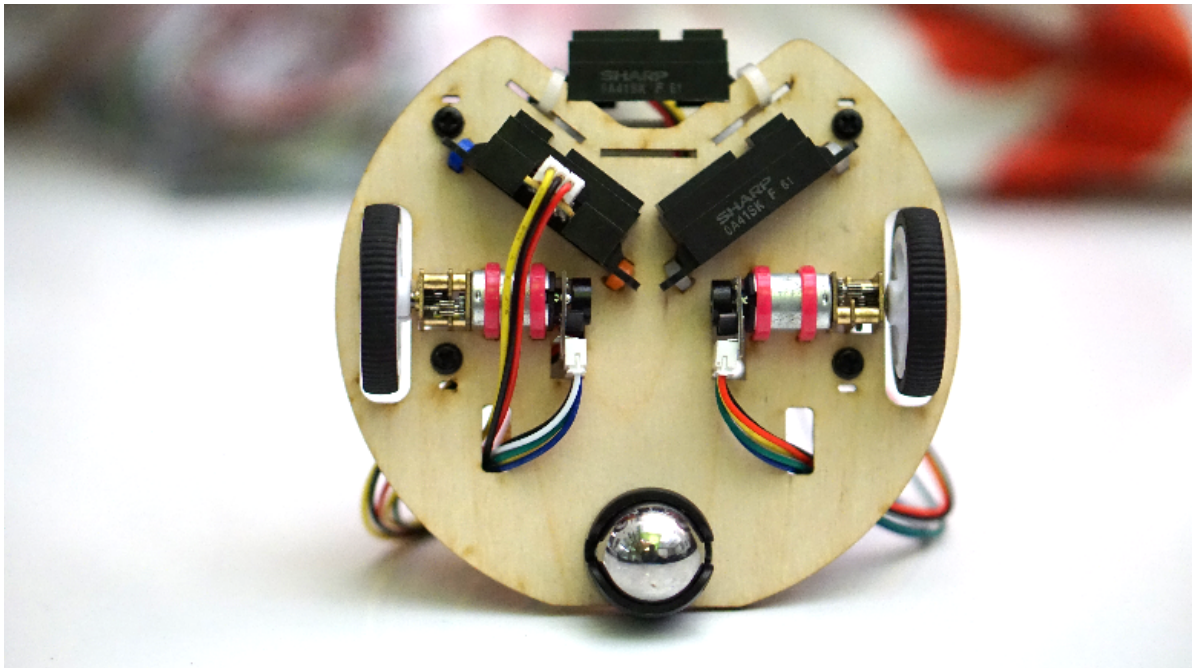
Lab 3 - Build day and Encoders

- Robot Assembly
- Encoders
    - Polling
    - Interrupts

Robot Chassis Assembly

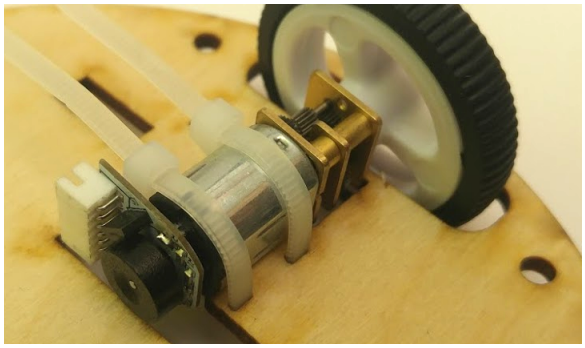
The first thing we'll be doing today is assembling the chassis of your micromouse!
This is what it should look like at the end:





A couple notes to keep in mind while you're doing this:

- **Infrared sensors:** there are three of them attached by zip ties; one on top and two below. Note the orientations of each in the picture.
- **Metal ball caster:** screw in the shell first, then insert the ball.
- **Motors:** these are also attached with zip ties. Again, watch the orientation! The connectors should face horizontally out the back of the robot.



Checkoff #1

1. Show your mentor your assembled chassis!

Measuring Wheel Velocities

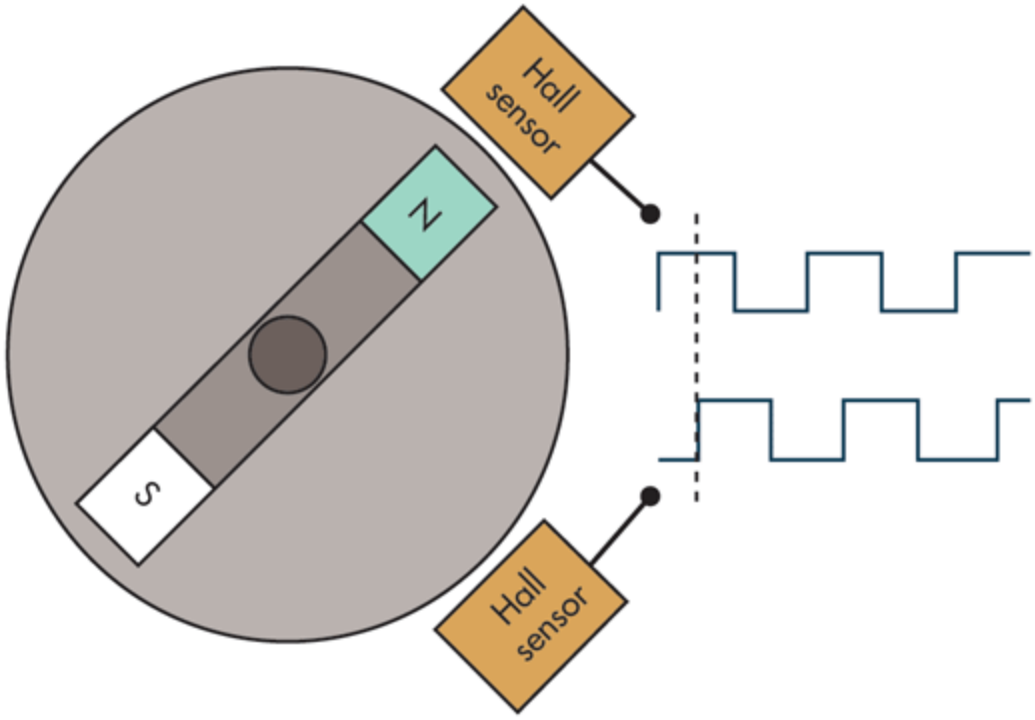
Last week, we went over how to control the two motors on our micromouse.
To effectively navigate a maze, we'd also like to answer a few key questions:

- How quickly is our robot driving?
- How far has it driven?
- Is it driving straight, or moving at an angle?

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 it's closer to the north pole of a magnet
- LOW when it's 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
#define PIN_ENCODER_RIGHT_A 12
#define PIN_ENCODER_RIGHT_B 3

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

  Serial.begin(9600);
}

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

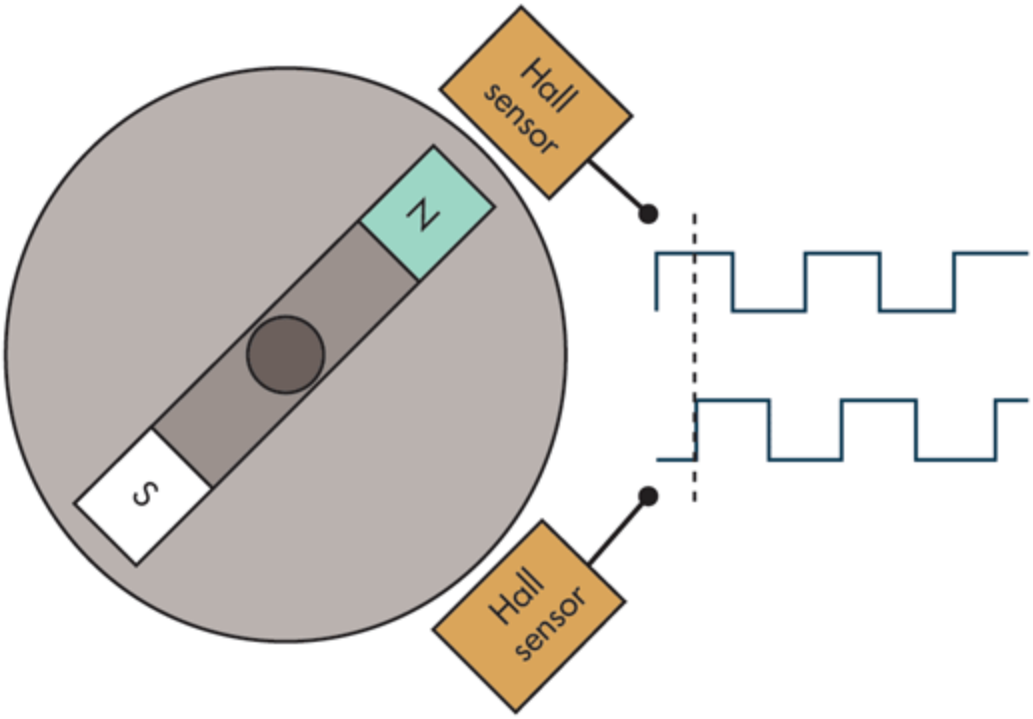
Spin the motor by hand, at various speeds. Note that by reading just one input, it's impossible distinguish between clockwise and counterclockwise rotations.

Checkoff #2

1. What happens to the Serial Plotter output when you spin the motor faster? Why?
2. Why do we need two hall effect sensors per motor? What would be wrong if we only had one?
3. 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.print(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!

Interrupts

One issue that you might have noticed already with polling is that it requires us to quickly and consistently check the state of the encoder outputs. This is simple and functional, but gets complicated when we want to do anything more with our code.

Think about all of the other things we want to do with our micromouse -- motor control, distance sensing, pathfinding -- we'll have inaccurate readings if any of these take longer than the length of an encoder pulse.

What about just polling more often?

Luckily there is a better way...

The microcontroller has specialized hardware that can listen for signals on a digital input pin. The moment it sees an edge, it can tell the program to stop whatever it's doing to count the tick, and then resume where it left off. These are called **interrupts**.

In Arduino, an interrupt can be made with the `attachInterrupt` function:

```
// Call function() when pin goes from LOW to HIGH
attachInterrupt(digitalPinToInterrupt(pin), function, RISING);
```

Try tracking both the left and right wheel positions in the Serial Plotter, via interrupts. Again, here's some starter code:

```
#define PIN_ENCODER_LEFT_A 11
#define PIN_ENCODER_LEFT_B 2
#define PIN_ENCODER_RIGHT_A 12
#define PIN_ENCODER_RIGHT_B 3

int left_position = 0;
int right_position = 0;
```

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

  // Attach an interrupts to an encoder pin, so that
leftEncoderRisingEdge() is automatically called on each rising edge
  // Our hardware only support this for pins #2 (left b) and #3
(right b)
  attachInterrupt(digitalPinToInterrupt(PIN_ENCODER_LEFT_B),
leftEncoderRisingEdge, RISING);

  // TODO: setup the encoder for the right wheel as well

  Serial.begin(9600);
}

void loop() {
  Serial.print(left_position);
  Serial.print(" ");
  Serial.println(right_position);
  // We can do whatever we want here!
  delay(100);
}

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

Checkoff #3

Use your Serial Plotter to simultaneously view the positions of the left and right wheels.

1. How many ticks correspond to a revolution? Where do you think this number comes from?
2. What do interrupts do? Why would we use them instead of polling?
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?
    b. (Optional) Try plotting each wheel's velocity!