1. Enough with the angles!

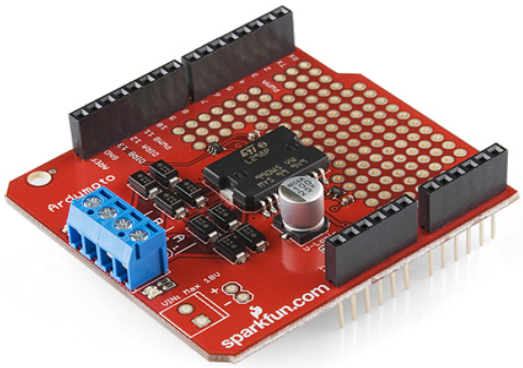
Complete the module from last week, where you combine inputs from the accelerometer and gyroscope to work out your angle. Use the Processing code to test if your code is working well. It needs to be quick and responsive to sudden changes, like the board falling over.

2. Plug in Motors, Build a robot

In order to work power a motor, we need to draw more current than the Arduino is capable of delivering. We can get around this problem using a motor driver 'shield'. Shields are stackable Arduino-compatible boards that sit on top of your Arduino, and can enhance its capabilities. In our case, the shield can power two DC motors and receive power from an external power source (9V battery). We're using the Ardumoto shield whose specifications are available here: http://www.sparkfun.com/products/9815

There are two goals for today's class.

A. Build a basic robot. Tape the motors securely to the bottom of the cardboard box. The Arduino can sit inside the box or be taped on top. The motors you have are the Mini Metal Gear Motor with a 24:1 built in gear ratio (specifications at http://www.sparkfun.com/products/8913). I chose these to give you a reasonable tradeoff between torque (measured in oz-inch or Newton-meters) and speed (measured in rpm).



B. Learn how to control the motors. Make your robot trace out a square. You'll have to work out how to do an on-the-spot turn. How about tracing out an arc of a circle? Once you have this worked out, you can try out more interesting trajectories.

To get you started with motor control, I'm pasting below a section of the Ardumoto Quickstart guide. When finished, your two motors will run sequentially in one direction at two different speeds, then the opposite direction at two speeds.

Explanation of the Sketch

The example sketch has been designed to be as simple as possible. First, we identify what pins will get what signals.

```
int pwm_a = 3;  //PWM control for motor outputs 1 and 2 is on digital pin 3
int pwm_b = 11;  //PWM control for motor outputs 3 and 4 is on digital pin 11
int dir_a = 12;  //dir control for motor outputs 1 and 2 is on digital pin 12
int dir_b = 13;  //dir control for motor outputs 3 and 4 is on digital pin 13

void setup()
{
  pinMode(pwm_a, OUTPUT);  //Set control pins to be outputs
```

```
  pinMode(pwm_b, OUTPUT);
  pinMode(dir_a, OUTPUT);
  pinMode(dir_b, OUTPUT);
}

void loop()
{
  //set both motors to run at (100/255 = 39)% duty cycle (slow)
  analogWrite(pwm_a, 100);
  analogWrite(pwm_b, 100);

  digitalWrite(dir_a, LOW);  //Set motor direction
  digitalWrite(dir_b, LOW);  //Set motor direction

  delay(1000);

  //set both motors to run at 100% duty cycle (fast)
  analogWrite(pwm_a, 255);
  analogWrite(pwm_b, 255);

  delay(1000);

  digitalWrite(dir_a, HIGH);  //Reverse motor direction
  digitalWrite(dir_b, HIGH);  //Reverse motor direction

  delay(1000);

  //set both motors to run at (100/255 = 39)% duty cycle
  analogWrite(pwm_a, 100);
  analogWrite(pwm_b, 100);

  delay(1000);
}
```

The function of each line is likely self explanatory, but the way the jobs get done may not be if you're new to Arduino. The digitalWrite function sets a pin either high or low, and is used here for controlling the direction of the motors (dir_a corresponds to the motor strapped between outputs 1 and 2, and dir_b corresponds to the motor strapped between outputs 3 and 4). The delay functions are in milliseconds and are there to provide a little wait-time so the user can more easily see the operation of the Ardumoto. The analogWrite function is a PWM (Pulse Width Modulation) function, and it's this value that sets the speed of the motors (pwm_a goes with dir_a and outputs 1 and 2, pwm_b goes with dir_b and outputs 3 and 4). This function will accept values from 0 (motor's stopped) to 255 (full throttle). **And...that's it! You're off and running (into walls, chairs, cats...)**

3. Give it a sense of balance.

Securely tape the accelerometer/gyroscope to the bottom of your robot (between the motors). Why is this the best place to put it?

Test whether it's accurately measuring the tilt of your robot (you may need to re-calibrate some of the zero values). Brainstorm with your team members, and take a first stab at writing a program that would make your robot balance on two wheels.