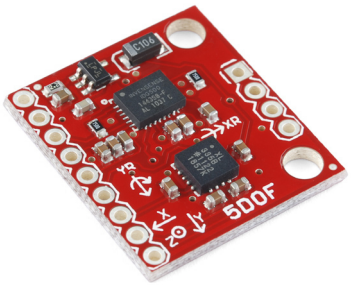
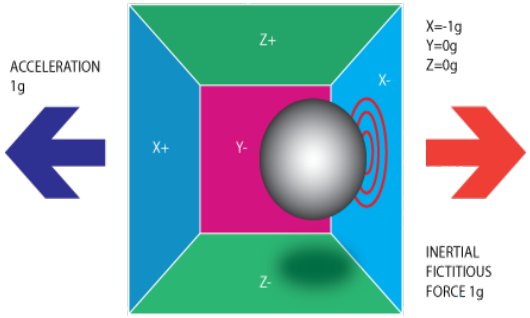
1. Sensing your orientation: how to use an accelerometer

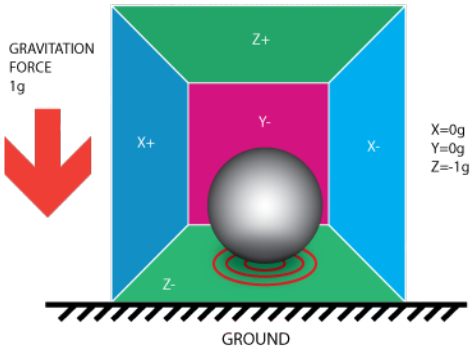


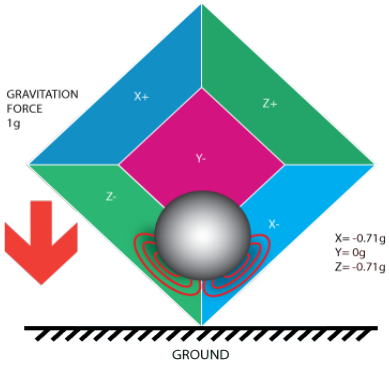
You've learned how to take a sensor, connect it to the Arduino, and use it in a circuit. To move on to the goal of stability, we need to understand how to use an accelerometer.

The sensor we'll work is a 5 degree of freedom Inertial Measurement Unit, available from http://www.sparkfun.com/products/11072. It combines a three axis ADXL335 accelerometer with a two axis IDG500 gyroscope. For now we'll just use the accelerometer.

A three axis accelerometer detects linear accelerations in three perpendicular directions. If it helps, picture a ball inside a box with pressure sensitive walls. As you shake the box around, the ball presses against different walls, which tells you the direction of acceleration. If the accelerometer is not moving, the ball will still push against the walls simply due to gravity. By comparing the readings on the x, y and z axis, you can work out the orientation of a stationary object.







a. Tape the accelerometer to the breadboard and connect it to the Arduino. Work out the minimum and maximum acceleration readings in the x, y and z direction.

b. Use Processing to graph the acceleration in the x direction. Do the same for the y and z direction. How do the x, y and z readings correspond to the physical orientation of the breadboard? How would you move the breadboard to vary the x, y or z reading?

c. Set up 4 LEDs in the shape of a cross. Build a tilt sensitive platform where each LED lights up when you tilt the breadboard in that direction. The brightness of the LED should correspond to the degree of tilt - the steeper the angle, the brighter the light.

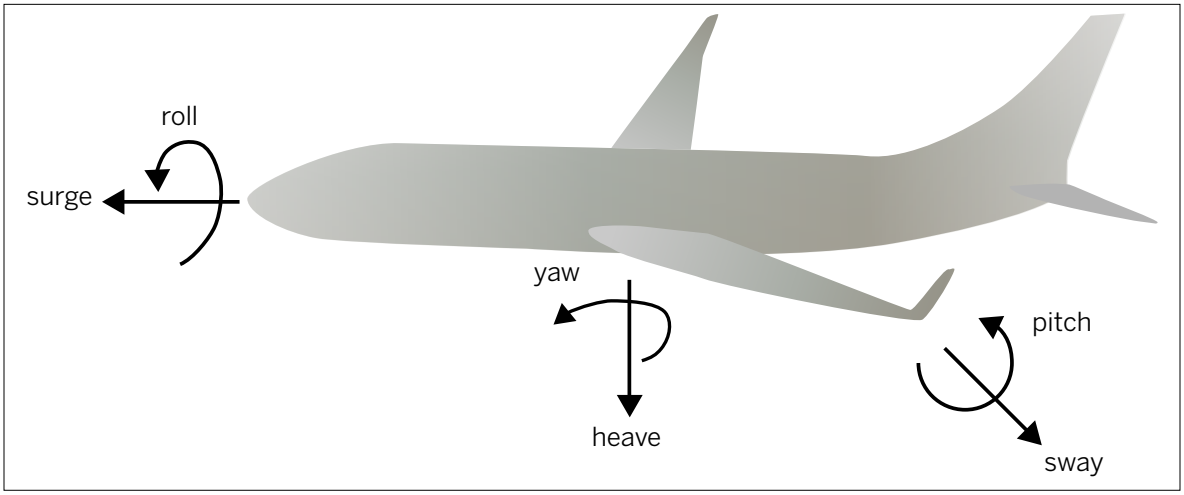


Figure 8-14
Rotations and translations of a body in three dimensions.

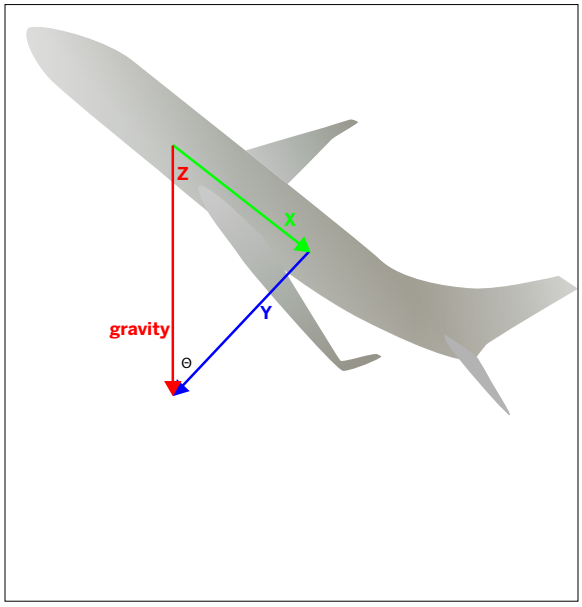
Determining Pitch and Roll from an Accelerometer

Three-axis accelerometers like the one you're using measure the linear acceleration of a body on each axis—in other words, the surge, sway, or heave of a body. They don't give you the roll, pitch, or yaw. However, you can calculate the roll and pitch when you know the acceleration along each axis. That calculation takes some tricky trigonometry. For a full explanation, see Freescale Semiconductor's application note on accelerometers at http://cache.freescale.com/files/sensors/doc/app_note/AN3461.pdf. Here are the highlights:

The force of gravity always acts perpendicular to the earth's surface. So when an object is tilted at an angle (called theta, or θ), part of that force acts along the X axis of the object, and part acts along the Y axis (see Figure 8-15). The X-axis acceleration and the Y-axis acceleration add up to the total force of gravity using the Pythagorean Theorem: $x^2 + y^2 = z^2$.

Since you know that, you can calculate the portions of the acceleration using sines and cosines. The X-axis portion of the acceleration is gravity * sinθ, and the Y-axis portion is gravity * cosθ (remember, sine = opposite/hypotenuse, and cosine = adjacent/hypotenuse).

Figure 8-15
Calculating the portion of the force of gravity based on the angle of tilt.



2. Where am I? Work out your angle with an accelerometer

a. Write a program that reads in the x, y and z accelerometer sensor values, and converts these numbers into voltages. (Be careful about rounding errors. If you're dividing two numbers, the Arduino language will round off the result, unless you put in the decimal form by hand. For example, 3/4 = 0, but 3/4.0 = 0.75)

Use the serial port to output all three voltages in the same line, with a code snippet like this:

```
Serial.print(xVoltage);
Serial.print(",");
Serial.print(yVoltage);
Serial.print(",");
Serial.println(zVoltage);
```

b. Next, we have to calibrate the accelerometer. While your program is running, play around with the orientation of the breadboard. Watch the voltage values in the serial monitor. You'll notice that the voltages aren't centered, so an orientation of zero degrees doesn't correspond to zero voltage. You can fix this by subtracting a number from the voltage. Define three different offset values:

```
float xZero = ?;        float yZero = ?;        float zZero = ?;
```

and set them to appropriate values so that **xVoltage - xZero** and **yVoltage - yZero** are both zero when the board is lying flat (x and y axis are zero degrees to the horizontal). Similarly, **zVoltage - zZero** should be zero when the z axis is at 0 degrees to the horizontal.

c. The voltages should now be appropriately zeroed. But these voltages still need to be converted into accelerations. When the accelerometer is sitting flat on the table, it should experience zero gs in the x and y directions, and 1 g in the z direction. Use this to work out the overall scale factor. In the end, you should have three accelerations, measured in g's.

```
float xAcc = (xVoltage - xZero)/Scale;
float yAcc = (yVoltage - yZero)/Scale;
float zAcc = (zVoltage - zZero)/Scale;
```

d. The last step is to go from accelerations to angles. You can work this out with some trigonometry. It's trickier than it sounds, because you need to do this in a way that has no divergences - the angle shouldn't blow up when the denominator goes to zero. One way out is to use an approximate answer, such as in the code below. For the gory details on how this is derived, you can visit this link: http://cache.freescale.com/files/sensors/doc/app_note/AN3461.pdf

New command: pow(x,2) raises x to the second power.

```
// apply trigonometry to get the pitch and roll:
float pitch = atan(xAxis/sqrt(pow(yAxis,2) + pow(zAxis,2)));
float roll = atan(yAxis/sqrt(pow(xAxis,2) + pow(zAxis,2)));
//convert radians into degrees
pitch = pitch * (180.0/PI);
roll = roll * (180.0/PI) ;
```

e. Print the results. Go to your serial monitor and check if the results make sense.

```
Serial.print(pitch);
Serial.print(",");
Serial.println(roll);
```

No matter how good your accelerometer is, you'll discover that angle readings aren't very accurate, and can be quite noisy. The calculations above assume that the only force acting on the sensor is gravity. But in reality, your sensor may be falling, shaking, rattling, or rolling. These forces lead to additional accelerations or decelerations in the three axes. The sensor can't distinguish these new forces from gravity, and so your angle reading can go haywire. In the next class, we'll add in data from gyroscopes in order to help adjust for these forces.

3. Animate your orientation in real-time. Now in 3D!

In this project, we'll use the program that you just wrote. First, make sure that the only thing being written to the serial port is the pitch and roll variables, using the chunk of code right above.

Next, follow the instructions on the next page to write a Processing program that will visualize the orientation of your breadboard, in 3D.

Connect It This Processing sketch reads the incoming data from the microcontroller and uses it to change the attitude of a disc onscreen in three dimensions. It will work with either of the accelerometer sketches above, because they both output the same data in the same format. Make sure the serial port opened by the sketch matches the one to which your microcontroller is connected.

```
/*
 Accelerometer Tilt
 Context: Processing

 Takes the values in serially from an accelerometer
 attached to a microcontroller and uses them to set the
 attitude of a disk on the screen.
 */
import processing.serial.*;      // import the serial lib

float pitch, roll;               // pitch and roll
float position;                  // position to translate to

Serial myPort;                   // the serial port
```

» The setup() method initializes the window, the serial connection, and sets the graphics smoothing.

```
void setup() {
  // draw the window:
  size(400, 400, P3D);
  // calculate translate position for disc:
  position = width/2;

  // List all the available serial ports
  println(Serial.list());

  // Open whatever port is the one you're using.
  myPort = new Serial(this, Serial.list()[2], 9600);
  // only generate a serial event when you get a newline:
  myPort.bufferUntil('\n');
  // enable smoothing for 3D:
  hint(ENABLE_OPENGL_4X_SMOOTH);
}
```

» You will probably need to look at the output of Serial.list() and change this number to match the serial port that corresponds to your microcontroller.

» The draw() method just refreshes the screen in the window, as usual. It calls a method, setAttitude(), to calculate the tilt of the plane. Then it calls a method, tilt(), to actually tilt the plane.

```
void draw () {
  // colors inspired by the Amazon rainforest:
  background(#20542E);
  fill(#79BF3D);
  // draw the disc:
  tilt();
}
```

The 3D system in Processing works on rotations from zero to 2*PI. tilt() maps the accelerometer angles into that range. It uses Processing's translate() and rotate() methods to move and rotate the plane of the disc to correspond with the accelerometer's movement.

```
void tilt() {
  // translate from origin to center:
  translate(position, position, position);

  // X is front-to-back:
  rotateX(radians(roll + 90));
  // Y is left-to-right:
  rotateY(radians(pitch) );

  // set the disc fill color:
  fill(#79BF3D);
  // draw the disc:
  ellipse(0, 0, width/4, width/4);
  // set the text fill color:
  fill(#20542E);
  // Draw some text so you can tell front from back:
  text(pitch + "," + roll, -40, 10, 1);
}
```

The serialEvent() method reads all the incoming serial bytes and parses them as comma-separated ASCII values, just as you did in Project 2, Monski pong in Chapter 2.

```
void serialEvent(Serial myPort) {
  // read the serial buffer:
  String myString = myPort.readStringUntil('\n');

  // if you got any bytes other than the linefeed:
  if (myString != null) {
    myString = trim(myString);
    // split the string at the commas
    String items[] = split(myString, ',');
    if (items.length > 1) {
      pitch = float(items[0]);
      roll = float(items[1]);
    }
  }
}
```

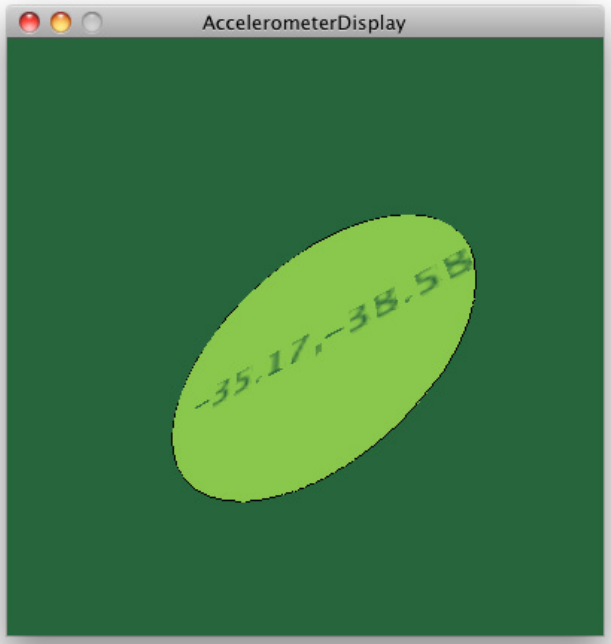


Figure 8-16
The output of the Processing accelerometer sketch.

Though it may seem like a lot of work to go from the raw output of an accelerometer to the visualization shown in Figure 8-16, it's useful to understand the process. You went from the translation of acclerations along three axes into analog voltages, then converted those voltages to digital values in the microcontroller's memory using analogRead(). From there, you converted the digital values into voltage readings, and then converted those to acceleration measurements relative to the acceleration due to gravity. Then, you used some trigonometry to convert the results to angles in degrees.

The advantage of having the results in degrees is that it's a known standard measurement, so you didn't have to do a lot of mapping when you sent the values to Processing. Instead, Processing could take the output from an accelerometer that gave it pitch and roll in degrees.

You don't always need this level of standardization. For many applications, all you care about is that the accelerometer readings are changing. However, if you want to convert those readings into a measurement of attitude relative to the ground, the process you went through is the process you'll use.

X

Address 2007 by Mouna Andraos and Sonali Sridhar
Address shows that location technologies don't have to be purely utilitarian. *Photo by J. Nordberg.*