1. Use today to finish your code for obtaining angles from the accelerometer and gyroscope. Plug all 5 accelerometer and gyroscope inputs into the Arduino. I have xAcc, yAcc and zAcc plugged in to ports 0,1,2, and xRate and yRate in 3 and 4. Power up the sensor, and **also connect the pin labelled AZ to ground.** To help you along if you're getting stuck at any point, here is my code for obtaining angles from the sensors.

```
// Code to get pitch and roll using ACCELEROMETER

float xZero = 1.41;   //zero the x voltage
float yZero = 1.42;   //zero the y voltage
float zZero = 1.46;   //zero the z voltage
float xVoltage, yVoltage, zVoltage;
float xAcc, yAcc, zAcc;
float accScale = 0.27;

void setup(){
  Serial.begin(9600);
}

void loop(){

  //convert sensor values to voltages
  xVoltage = 5*(analogRead(0)/1023.0);
  yVoltage = 5*(analogRead(1)/1023.0);
  zVoltage = 5*(analogRead(2)/1023.0);

  // convert voltages into accelerations
  // (in units of g)
  xAcc = (xVoltage-xZero)/accScale;
  yAcc = (yVoltage-yZero)/accScale;
  zAcc = (zVoltage-zZero)/accScale;

  //convert acceleration into angle using trigonometry
  float pitch = atan(xAcc/sqrt(pow(yAcc,2) +
pow(zAcc,2)));
  float roll = atan(yAcc/sqrt(pow(xAcc,2) +
pow(zAcc,2)));

  //convert radians into degrees
  pitch = pitch * (180.0/PI);
  roll = roll * (180.0/PI) ;

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


  delay(100);

}
```

```
//Code to get pitch and roll using GYROSCOPE

float currentAngle = 0;      //Keep track of our current angle

float xZeroGyro = 1.33;  //zero the x voltage
float yZeroGyro = 1.35;  //zero the y voltage

int delaytime = 10;         // a tiny delay time
float pitch = 0;
float roll = 0;

float gyroScale = .002;  // used to convert volts into
// degrees/second (don't change this value)

float rotationThreshold = 5; // A value in degrees / second
// values lower than this will be ignored

void setup() {
  Serial.begin(9600);
}

void loop() {
  // this next bit converts gyro sensor values to voltages,
  // subtracts away the zero value,
  // and converts voltage into degrees / second
  float xGyro = (5*(analogRead(3)/1023.0) - xZeroGyro)/
gyroScale;
  float yGyro = (5*(analogRead(4)/1023.0) - yZeroGyro)/
gyroScale;

  delay(delaytime);

// If the angular velocity exceeds a threshold
// Then update the angle (this helps with the drift problem)
  if (xGyro >= rotationThreshold || xGyro <=
-rotationThreshold){
    roll = roll - xGyro*delaytime/1000.0;
  }
  if (yGyro >= rotationThreshold || yGyro <=
-rotationThreshold){
    pitch = pitch + yGyro*delaytime/1000.0;
  }

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

}
```

Check that your programs work for some test angles using the Serial Monitor or the Processing program.

2. Now we'd like to combine input from the gyroscope and the accelerometer. The reason for this is that accelerometers are noisy, and gyros tend to drift. So we will use each one to compensate for the shortcomings of the other.

One way to do this is simply to combine your two programs together. Let's say we have a variable that stores the pitch angle from the last time your program went through the loop. Call this variable **pitch**.

We can update the angle using the gyroscope:
pitch_gyro = pitch + yGyro*delaytime/1000.0;

or we can just use the accelerometer to find the new angle
pitch_acc

These will give you two different answers. The final pitch, then is a weighted sum of the two:

pitch = 0.95*pitch_gyro + 0.05*pitch_acc

$$\begin{cases} \theta_{est}\,|_{t_2} = \alpha(\theta_{est}\,|_{t_1} + G\Delta t) + \beta\theta_{A_x} \\ \alpha + \beta = 1 \end{cases}$$

What this says is to mostly trust the gyroscope's angle, but to use the accelerometer a little bit to correct for the drift. You can play with the two coefficients (0.95 and 0.05) to see what works best, but remember that they must add to 1.

Check that this works properly, using the Serial Monitor or the Processing code.

Here is the final code that I used to do this (in case you're getting stuck):

```
//ACCEL AND GYRO COMBINED CODE
//ACCEL variables
float xZero = 1.41;
float yZero = 1.42;
float zZero = 1.46;
float xVoltage, yVoltage, zVoltage;
float xAcc, yAcc, zAcc;
float accScale = 0.27;
float pitch_a, roll_a;

//GYRO variables
float gyroScale = .002;
float rotationThreshold = 5;   //Minimum deg/sec to keep track of - helps with gyro drifting
float currentAngle = 0;        //Keep track of our current angle
float xZeroGyro = 1.33;
float yZeroGyro = 1.35;
int delaytime = 10;
float pitch_g, roll_g;

//initialize angles
float pitch = 0;
```

```
//PROCESSING CODE FOR GRAPHING

/*
Accelerometer Tilt
Context: Processing
Takes the values in serially from an accelerometer
attached to a microcontroller and uses them to set
the angle 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

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.
```

```
float roll = 0;

void setup() {
  Serial.begin(9600);
}

void loop(){

  //convert acceleration sensor values into voltages
  xVoltage = 5*(analogRead(0)/1023.0);
  yVoltage = 5*(analogRead(1)/1023.0);
  zVoltage = 5*(analogRead(2)/1023.0);

  //convert voltages into accelerations (in units of
g)
  xAcc = (xVoltage-xZero)/accScale;
  yAcc = (yVoltage-yZero)/accScale;
  zAcc = (zVoltage-zZero)/accScale;

  //convert acceleration into angle using
trigonometry
  pitch_a = atan(xAcc/sqrt(pow(yAcc,2) +
pow(zAcc,2)));
  roll_a = atan(yAcc/sqrt(pow(xAcc,2) +
pow(zAcc,2)));
  //convert radians into degrees
  pitch_a = pitch_a * (180.0/PI);
  roll_a = roll_a * (180.0/PI) ;

  //convert gyroscope sensor values into voltages
  float xGyro = (5*(analogRead(3)/1023.0) -
xZeroGyro)/gyroScale;
  float yGyro = (5*(analogRead(4)/1023.0) -
yZeroGyro)/gyroScale;

  delay(delaytime);

  //if the angular velocity exceeds a threshold
  //update your angle using the angular velocity
  if (xGyro >= rotationThreshold || xGyro <=
-rotationThreshold){
    roll_g = roll - xGyro*delaytime/1000.0;
  }
  if (yGyro >= rotationThreshold || yGyro <=
-rotationThreshold){
    pitch_g = pitch + yGyro*delaytime/1000.0;
  }

  //combine gyroscope and acceleromter angles
  pitch = 0.95*pitch_g + 0.05*pitch_a;
  roll = 0.95*roll_g + 0.05*roll_a;

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

}
```

```
myPort = new Serial(this, Serial.list()[0], 9600);
// only generate a serial event when you get a
newline:
myPort.bufferUntil('\n');
// enable smoothing for 3D:
//hint(ENABLE_OPENGL_4X_SMOOTH);
}


void draw () {
background(#20542E);
fill(#79BF3D);
// draw the disc:
tilt();
}


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);
}


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]);
    }
  }
}
```