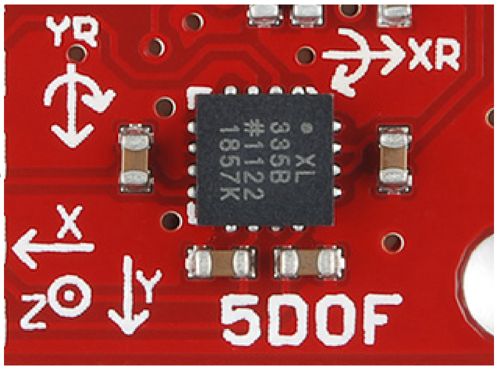
1. Introduction to gyroscopes



After the previous workshop, you should have a feel for what an accelerometer does, and have worked out how to get an angle reading from an accelerometer.

Now, we're going to do the same with a gyroscope. The 5 degree of freedom IMU sensor contains a two axis gyroscope. *A gyroscope measures angular velocity, or the spin, about a particular axis.* In our case, the axes of rotations are labeled as XR and YR in the figure.

Tape the sensor to the breadboard. Write a program that reads in data from the gyroscope along the XR axis. Plot the output using Processing.

What happens if:
- you hold the board at a fixed angle?
- you spin the board about the XR axis?
- you spin the board about the YR axis?
- you turn the board upside down?

Based on your observations, what is the key difference between a gyroscope and an accelerometer?

Think about this: How can you use a gyroscope to work out your angle?

2. Where am I? Find your angle using a gyroscope

We're going to work out how to get a angle reading from a gyroscope. To start off, let's just work with one of the gyroscope axes - the XR axis.

The basic idea is that a gyroscope gives you an angular velocity, or the rate of change of angle (measured in radians/sec). In a sufficiently small time interval (call it **delayTime**), we can get the angular displacement using **deltaAngle = gyroRate * delayTime** and use it to update the angle.

a. Set up some initial variables

```
int gyroPin = 0;                //Gyro is connected to analog pin 0
float MaxVoltage = 5;           //Voltage is being measured on a scale from 0 to 5V
float gyroZeroVoltage = 0;       //The Voltage from the Gyro when it's sitting still
float gyroSensitivity = 0.002;  //A conversion factor from volts to degrees/sec.
                                // For our gyro, 2mV corresponds to 1 deg/sec
float rotationThreshold = 1;    //Minimum deg/sec to keep track of. Helps with gyro drifting
float currentAngle = 0;         //Start of with an angle of 0
int delayTime = 10;             //10 ms delay
```

b. Convert the gyro reading into a Voltage

The Arduino gives you a number between 0 and 1023 representing 0 to 5V. Convert this to a voltage.

c. Zero the voltage.

Keep the gyro sitting absolutely still and note down the voltage reading. This is **gyroZeroVoltage**, and you will need to subtract it from the voltage reading. Once you're done, the calibrated voltage should be 0 when the gyro is at rest.

d. Convert voltage to angular velocity (degrees / sec)

To do this, divide the voltage reading by the conversion factor called **gyroSensitivity**. You should end up with code similar to this:

```
 //This line converts the 0-1023 signal to 0-5V
 gyroVoltage = (analogRead(gyroPin) * MaxVoltage) / 1023;
 //This line calibrates the voltage, and converts it into an angular velocity
 gyroRate = (gyroRate - gyroZeroVoltage)/gyroSensitivity;
```

e. Use the angular velocity to update the angle

If the angular velocity exceeds the threshold **rotationThreshold**, then use it to update the **currentAngle**

```
 if (gyroRate >= rotationThreshold || gyroRate <= -rotationThreshold){
   currentAngle = currentAngle + gyroRate*delaytime/1000.0;
 }
```

NOTE: Don't forget to put in a delay of **delaytime** somewhere between your measurement and the angle update step.

f. Make sure your angle is always between +360 and - 360

If it exceeds 360 degrees, just subtract away 360.

g. Repeat these steps for the other gyroscope axis

You should finally get two angles, one from each axis of the gyroscope. These are the **pitch** and the **roll** of your sensor, the same quantities we got from the accelerometer, but using a very different method.

Finally, remove all Serial print commands except the following:

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

h. See how you did. Open up the Processing visualization from last class and check if the on-screen tilt matches the orientation of your board.

If you find that your gyro is drifting too much, try increasing **rotationThreshold**. Be careful not to set it too high, or you'll lose sensitivity to slow rotations.

You might find that the gyro angle is a third or so of what it should be. This is happening because we also need to connect the pin labeled AZ to ground. I'll solder on a wire to fix this.

We now have two ways of getting angles, each with its pros and cons. Accelerometers give you an absolute way of finding angles, but are sensitive to noise. Gyroscopes are better at responding to fast changes, but tend to drift over time. Try to come up with a way that will combine the two different methods. We'll go over this in the next class, and also start building robots.