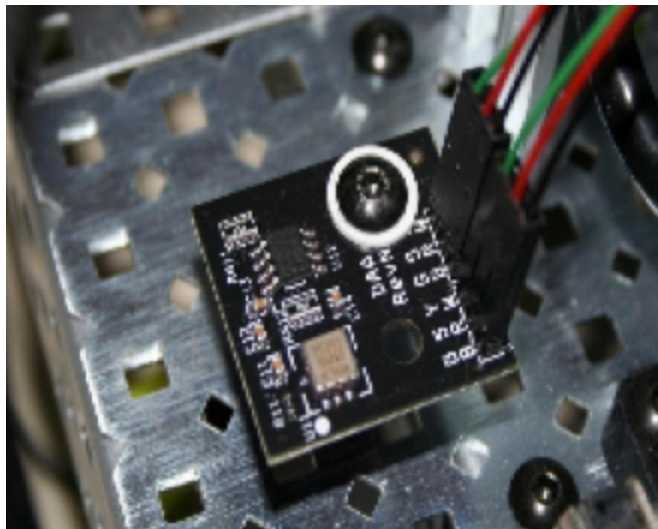


Accelerometers - measuring acceleration and tilt

Accelerometers measure acceleration in one or more axis. One typical usage is to measure robot acceleration. Another common usage is to measure robot tilt, in this case it measures the acceleration due to gravity.

Two-axis analog accelerometer



A commonly used part (shown in the picture above) is a two-axis accelerometer. This device can provide acceleration data in the X and Y-axes relative to the circuit board. The WPI Robotics Library you treats it as two separate devices, one for the X- axis and the other for the Y-axis. The accelerometer can be used as a tilt sensor – by measuring the acceleration of gravity. In this case, turning the device on the side would indicate 1000 milliGs or one G. Shown is a 2-axis accelerometer board connected to two analog inputs on the robot. **Note that this is not the accelerometer provided in the 2014 KOP.**

Analog Accelerometer code example

```
C++
class AccelerometerSample: public SampleRobot {
    AnalogAccelerometer *accel;
    double acceleration;

    AccelerometerSample()
```

Accelerometers - measuring acceleration and tilt

```
{
    accel = new AnalogAccelerometer(0); //create accelerometer on analog input
0
    accel->SetSensitivity(.018); // Set sensitivity to 18mV/g (ADXL193)
    accel->SetZero(2.5); //Set zero to 2.5V (actual value should be determined
experimentally)
}

public void OperatorControl() {
    while(IsOperatorControl() && IsEnabled())
    {
        acceleration = accel->GetAcceleration();
    }
}
}
```

Java

```
public class AccelerometerSample extends SampleRobot {
    AnalogAccelerometer accel;
    double acceleration;

    AccelerometerSample()
    {
        accel = new AnalogAccelerometer(0); //create accelerometer on analog input
0
        accel.setSensitivity(.018); // Set sensitivity to 18mV/g (ADXL193)
        accel.setZero(2.5); //Set zero to 2.5V (actual value should be determined
experimentally)
    }

    public void operatorControl() {
        while(isOperatorControl() && isEnabled())
        {
            acceleration = accel.getAcceleration();
        }
    }
}
```

A brief code example is shown above which illustrates how to set up an analog accelerometer connected to analog channel 1. The sensitivity and zero voltages were set according to the

Accelerometers - measuring acceleration and tilt

[datasheet](#) (assumed part is ADXL193, zero voltage set to ideal. Would need to determine actual offset of specific part being used).

Accelerometer interface

C++

```
Accelerometer *accel;  
accel = new BuiltInAccelerometer(Accelerometer:kRange_4G);  
double xVal = accel->GetX();  
double yVal = accel->GetY();  
double zVal = accel->GetZ();
```

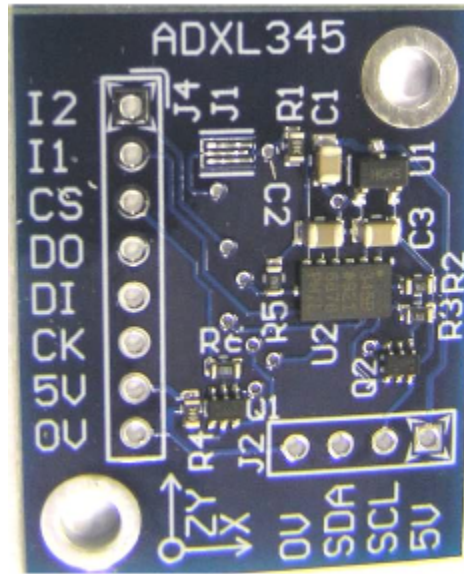
Java

```
Accelerometer accel;  
accel = new BuiltInAccelerometer();  
accel = new BuiltInAccelerometer(Accelerometer.Range.k4G);  
double xVal = accel.getX();  
double yVal = accel.getY();  
double zVal = accel.getZ();
```

Both classes for the ADXL345 and the class for the Built-In accelerometer all inherit/implement a common Accelerometer interface. The plan in the future is to try to get the AnalogAccelerometer class to derive from this interface as well. If you are planning on using one of these sensors it is recommended to write your code against the generic interface. That way you can change between the underlying classes, if desired, with minimal changes to your code. It will also help make your code more compatible with simulation as that capability continues to develop.

Accelerometers - measuring acceleration and tilt

ADXL345 Accelerometer



The ADXL345 is a three axis accelerometer provided as part of the sensor board in the 2012-2014 KOP. The ADXL345 is capable of measuring accelerations up to +/- 16g and communicates over I2C or SPI. Wiring instructions for either protocol can be found in the [FRC component datasheet](#). Additional information can be found in the Analog Devices ADXL345 [datasheet](#). WPILib provides a separate class for each protocol which handles the details of setting up the bus and enabling the sensor.

ADXL345 Code Example

C++

```
class AccelerometerSample: public SampleRobot {
    Accelerometer *accel;
    double accelerationX;
    double accelerationY;
    double accelerationZ;

    AccelerometerSample()
    {
        accel = new ADXL345_I2C(I2C::Port::kOnboard,
Accelerometer::Range::kRange_4G);
    }
}
```

Accelerometers - measuring acceleration and tilt

```
public void OperatorControl() {
    while(IsOperatorControl() && IsEnabled())
    {
        accelerationX = accel->GetX();
        accelerationY = accel->GetY();
        accelerationZ = accel->GetZ();
    }
}

Java
public class AccelerometerSample extends SampleRobot {
    Accelerometer accel;
    double accelerationX;
    double accelerationY;
    double accelerationZ;

    AccelerometerSample()
    {
        accel = new ADXL345_I2C(I2C.Port.kOnboard, Accelerometer.Range.k4G);
    }

    public void operatorControl() {
        while(isOperatorControl() && isEnabled())
        {
            accelerationX = accel.getX();
            accelerationY = accel.getY();
            accelerationZ = accel.getZ();
        }
    }
}
```

A brief code example is shown above illustrating the use of the ADXL345 connected to the on-board I2C bus. The accelerometer has been set to operate in +/- 2g mode. The example illustrates both only the single axis method of getting the sensor values, using the Accelerometer interface. If you need synchronized readings of all 3 axes, you will have to forgo the interface and use the ADXL345 class directly to have access to the GetAccelerations() method. SPI operation is similar, refer to the Javadoc/Doxygen for the ADXL345_SPI class for additional details on using the sensor over SPI.

Accelerometers - measuring acceleration and tilt

Built-In Accelerometer

The roboRIO contains a built-in 3-axis accelerometer with a range of +/- 8g, 12 bit resolution, and a 800 Sample/s sample rate. To use this accelerometer, use the BuiltInAccelerometer class. See the [Accelerometer Interface](#) section above for code illustrating the use of this accelerometer operating in the +/-4g mode using the generic Accelerometer interface (note when using this interface that the built-in accelerometer does not support the +/-16g mode).