A SUN SPOT APPLICATION NOTE

Using the LIS3L02AQ Accelerometer

Ron Goldman

The Sun SPOT demo sensor board includes an LIS3L02AQ accelerometer that can be used to measure the orientation or motion of the SPOT. This application note describes how to use and calibrate the LIS3L02AQ accelerometer.

The LIS3L02AQ is a low-power, three-axis linear accelerometer that is mounted on the demo sensor board of the Sun SPOT. The accelerometer can be used to measure the motion of the SPOT. It can also measure the SPOT's orientation with respect to gravity. The Z-axis is perpendicular to the Sun SPOT boards. The X-axis is parallel to the row of LEDs on the sensor board. The Y-axis is parallel to the long edge of the sensor board.

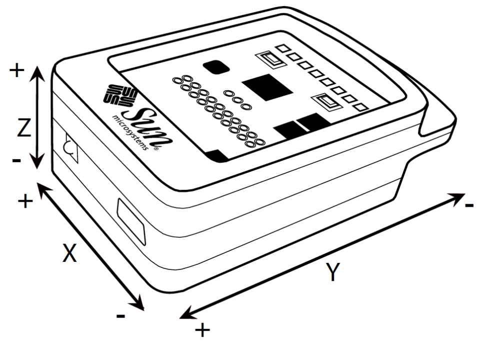

Figure 1. Accelerometer X, Y and Z axes

In Figure 1, the plus (+) on the end of an axis indicates that when the device's acceleration vector increases in that direction, the associated accelerometer readings will grow larger. If the SPOT is sitting flat on a table then the acceleration due to the Earth's gravity will be 1g along the positive Z-axis, and 0g along the X and Y axes. Note that while gravity is pointing down (along the negative Z-axis) this is equivalent to a uniform upwards acceleration of 1g according to the Einstein equivalence principle even though the SPOT is not moving.

The LIS3L02AQ accelerometer consists of a Micro-Electro-Mechanical System (MEMS) sensor element that is displaced from its nominal position when a linear acceleration is applied, causing an electrical imbalance that is read via the demo sensor board's analog-to-digital converter. The raw voltage value is then converted to g-force units. The accelerometer can be set to measure accelerations over a scale of either ± 2g or ± 6g. For a full description of the technical specifications of the LIS3L02AQ accelerometer please refer to the STMicroelectronics documentation available at http://www.st.com/stonline/products/literature/od/9321.pdf.

The SPOT library includes both the `IAccelerometer3D` interface that defines the basic methods that any three-axis accelerometer should support, and the `LIS3L02AQAccelerometer` class that implements that interface along with several other methods specific to the LIS3L02AQ.

The Basic `IAccelerometer3D` API

The basic methods used to read the current acceleration in G's along each axis are `getAccelX()`, `getAccelY()` and `getAccelZ()`. A fourth method, `getAccel()`, returns the magnitude of the current total acceleration, $|\bar{a}|$. This is the vector sum of the acceleration along the X, Y & Z axes, $|\bar{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$.

Here is a code fragment that will loop until the acceleration along the X-axis exceeds ¼ g:

```
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.IAccelerometer3d;
import com.sun.spot.util.Utils;

    IAccelerometer3D acc = EdemoBoard.getInstance().getAccelerometer();

    while (true) {
        double ax = acc.getAccelX();
        if (ax >= 0.25) {
            System.out.println("X acceleration above threshold: " + ax);
            break;
        }
        Util.sleep(250);  // check every 1/4 second
    }
```

Here's another fragment to detect when the SPOT is in motion by checking for the total acceleration to deviate from the 1g of gravity:

```
    public boolean isMoving() throws IOException {
        double mag = acc.getAccel();
        return Math.abs(mag – 1.0) >= 0.1;
    }
```

Measuring relative acceleration

Another set of methods, `getRelativeAccelX()`, `getRelativeAccelY()`, `getRelativeAccelZ()` and `getRelativeAccel()`, return the current acceleration relative to a previously measured acceleration. This allows one to zero out the force of gravity on the SPOT and measure the relative acceleration only. The method, `setRestOffsets()`, computes the current acceleration along each axis and saves it, establishing a new zero reading to be used by the above methods. Here is the above code example rewritten using the relative acceleration routines:

```
    acc.setRestOffsets();    // zero out the current forces on the SPOT

    public boolean isMoving() throws IOException {
        return acc.getRelativeAccel() >= 0.1;
    }
```

Measuring tilt

A third set of methods, `getTiltX()`, `getTiltY()` and `getTiltZ()`, use the acceleration along an axis in order to compute the inclination, or tilt, of that axis with respect to the total acceleration the SPOT is experiencing. If the SPOT is at rest then the tilt is measured with respect to gravity.

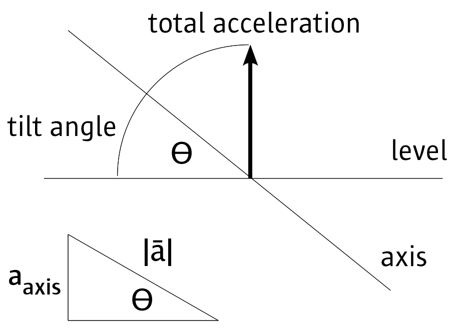

Figure 2. Computing the tilt

As shown in Figure 2 the tilt angle can be computed as $\Theta = \arcsin\left(\frac{a_{axis}}{|\bar{a}|}\right)$ where Θ is the tilt angle measured in radians (with a range of $-\frac{\pi}{2}$ to $+\frac{\pi}{2}$), a_{axis} is the acceleration measured along the given axis, and $|\bar{a}|$ is the total acceleration.

Here is a code example to measure the tilt of the SPOT and display the tilt in the LEDs like a bubble in a level:

```
    public void demoBubbleLevel() {
        for (int i = 0; i < 8; i++) {
            leds[i].setOff();                    // turn off all LEDs
            leds[i].setColor(LEDColor.BLUE);     // set them to be blue when lit
        }

        while (true) {
            try {
                int tiltX = (int)Math.toDegrees(acc.getTiltX()); // returns [-90, +90]
                int offset = -tiltX / 15;        // so bubble goes to higher side  [6, -6]
                if (offset < -3) offset = -3;    // clip angle to range [3, -3]
                if (offset > 3)  offset =  3;
                leds[3 + offset].setOn();        // use 2 LEDs to display "bubble""
                leds[4 + offset].setOn();
                Utils.sleep(50);                 // update 20 times per second
                leds[3 + offset].setOff();       // clear display
                leds[4 + offset].setOff();
            } catch (IOException ex) {
                System.out.println("Error reading accelerometer: " + ex);
            }
        }
    }
```

The above example is installed with the Sun SPOT SDK and can be found in:
`Demos/CodeSamples/AccelerometerSampleCode`

For most SPOT programs the functionality defined by `IAccelerometer3D` will be all that is needed.

The `LIS3L02AQAccelerometer` API

The LIS3L02AQ accelerometer can be set to measure accelerations over a scale of either ± 2g or ± 6g. The method `setScale(int)` is used to select the desired scale. it takes one argument, either the constant `SCALE_2G` or `SCALE_6G`.

```
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.LIS3L02AQAccelerometer;

    LIS3L02AQAccelerometer acc =
             (LIS3L02AQAccelerometer)EdemoBoard.getInstance().getAccelerometer();

    acc.setScale(LIS3L02AQAccelerometer.SCALE_2G);          // use the 2G scale
```

The method `getCurrentScale()` returns an integer (either `SCALE_2G` or `SCALE_6G`) indicating which scale is currently selected. The method `getScales()` returns an array of integers, { 2, 6 }, that list the scales the accelerometer supports. Note that the constants, `SCALE_2G` and `SCALE_6G`, can be used as indices into this array. So after the code snippet above to select the 2G scale, the statement:

```
    int scale = acc.getScales()[acc.getCurrentScale()];
```

would set the variable `scale` to a value of 2.

The methods `getRawX()`, `getRawY()` and `getRawZ()` are used to get the raw voltage values read from each axis of the accelerometer. They return a value in the range 0 to 1023. To convert this value to G's requires two steps: subtracting off the *zero offset* value for that axis which will shift the range to −*offset* to +*offset*, and then dividing by the *gain* for the axis to scale the value to G's.

$$a = \frac{\text{raw} - \text{zeroOffset}}{\text{gain}}$$

The nominal zero offset value is 465.5 and the nominal gain for the 2G scale is 186.2, while for the 6G scale it is 62.

The routines `getZeroOffsets()` and `getGains()` each return a two-dimensional array, [2][3], giving the values for each axis at each scale. The routine `getRestOffsets()` returns a similar array containing the offsets for computing the relative acceleration. So the code to convert from a raw value into G's looks like:

```
int scale = acc.getCurrentScale();
double[][] gains = acc.getGains();
double[][] zeroOffsets = acc.getZeroOffsets();
double[][] restOffsets = acc.getRestOffsets();

double accX = (acc.getRawX() - zeroOffsets[scale][0]) / gains[scale][0];
double relX = (acc.getRawX() - restOffsets[scale][0]) / gains[scale][0];
```

The zero offset and gain of each axis can differ by ±10% from one accelerometer to the next, or between axes of the same accelerometer. To get the greatest accuracy it is important to calibrate each SPOT's accelerometer as described below. Once the proper gains and zero offset values have been determined they can be passed to the accelerometer code with the methods `setZeroOffsets(`*`double[2][3]`*`)` and `setGains(`*`double[2][3]`*`)`. There is also the routine `setRestOffsets(`*`double[2][3]`*`)` that can be used to specify the offsets for calculating the relative acceleration. The method `saveCalibration()` takes the current arrays of gains, zero offsets and rest offsets and saves them to the flash memory on the demo board. When the accelerometer code is initialized it tries to read the calibration values back in—if they do not exist it will default to using the nominal values.

The LIS3L02AQ accelerometer has a self-test mode that is described in the *Sun SPOT Theory of Operation* document. The routine to enter/exit self-test mode is `selfTest(`*`boolean`*`)`. The routine `isInSelfTest()` returns true if the accelerometer is currently in self-test mode.

The final method that needs mention is `reset()`, which ensures that the accelerometer is not in self-test mode and also sets it to use the 2G scale.

Calibrating the accelerometer

A simple way to determine the correct zero offset and gain for a given axis of the accelerometer is to take two readings, one with the axis pointing straight upwards and one with it pointing straight downwards. Since both reading are aligned with gravity the average of the two readings is the zero offset, while one half of the difference between the two readings is the proper gain. This needs to be done for both scales.

The zero offset value can also be computed by taking a reading when an axis is perpendicular to the Earth's gravity—when one axis is pointing up or down the other two axes will read zero acceleration.

An application to do this calibration is part of the Sun SPOT SDK. It can be found in the `Demos` directory in the `CalibrateAccelerometer` folder. If you deploy it to a SPOT and run it you will be able to calibrate the SPOT's accelerometer. The program uses 6 LEDs to indicate which orientations need to be calibrated (red) and which already have (green). When the SPOT's orientation lines up with an uncalibrated orientation the corresponding LED will turn white. To make a reading along the current orientation press switch 1 (just below the LEDs on the left). The SPOT will signal that it is about to take a reading (flashing LEDs) and then for each axis take the average of 50 readings at the 2G scale, and then again using the 6G scale. If the SPOT's print output is being displayed, then the minimum, maximum and average values for each axis and scale will be printed. When readings have been taken along all 6 orientations the computed zero offsets and gains values will be printed.

This calibration process is done twice. The second time uses the values from the first to tighten the constraints on what is an acceptable orientation.

After the calibration process the accelerometer performs a self test (blue LEDs)—push switch 1 to start. If it passes then the calibration values, will be stored as persistent properties in the demo sensor board's Flash memory and used whenever a SPOT application uses the accelerometer in the future.

A Word on Sampling Rates

The rate at which an application needs to read the accelerometer will vary greatly and depends on the type of accelerometer data being measured. For an application concerned with measuring the SPOT's orientation a sample rate of 10-20 readings per second will usually suffice; likewise for gesture recognition. For measuring the motion of the SPOT when mounted in a toy slot car, a rate of 100 readings per second was more than adequate. If one wishes to measure vibrations then the Nyquist-Shannon sampling theorem tells us that the sampling frequency needs to be greater than twice the signal bandwidth. The question then is how high a sample rate is possible.

Figure 3 shows the various components involved in reading accelerometer values.

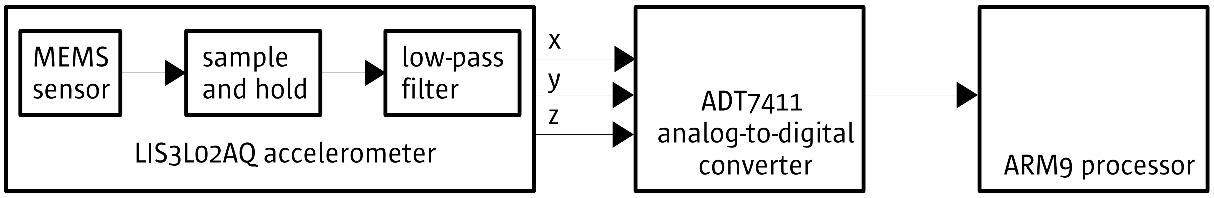

Figure 3. Components involved in reading the accelerometer

The MEMS sensors of the LIS3L02AQ accelerometer are capable of measuring accelerations over a maximum bandwidth of 4.0 KHz for the X and Y axis and 2.5KHz for the Z axis. They are sampled internally at 66KHz. To implement low-pass filtering for antialiasing and noise reduction each output pin from the accelerometer has a capacitor to provide a simple, single-pole low-pass filter. The capacitor used on the SPOT demo sensor board is 0.01µF which results in a cutoff frequency of 160Hz. Replacing this capacitor one can raise the cutoff frequency to 2.5KHz.

The ADT7411 analog-to-digital converter can operate in either a round-robin or single channel mode. Reading a single channel it can perform a conversion every 45 microseconds, a sampling rate of 22.2KHz. In round-robin mode the a-to-d converts each of its 8 analog inputs every 579 microseconds, yielding a sampling rate of 1.7KHz.

The SPOT application running on the ARM9 processor takes about 124 microseconds to read a single raw accelerometer value from the a-to-d, which gives a maximum sampling rate of 8.1KHz. To also convert these to G's takes 151 microseconds per reading for a maximum sampling rate of 6.6KHz. Reading raw values for all three axes requires 370 microseconds, a sampling rate of 2.7KHz. Converting to G's takes 460 microseconds for

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase®
Smarter legal research.