Using SensorTag as a Low-Cost Sensor Array for AutoCAD

Kean Walmsley – Autodesk

SD5013 In 2012 Texas Instruments Inc. launched the SensorTag, a $25 sensor array that communicates via Bluetooth Smart (also known as Bluetooth 4.0 or Bluetooth LE) with smartphones or PCs. The SensorTag contains temperature, humidity, and pressure sensors, along with an accelerometer, a gyroscope, and a magnetometer. And there are a couple of physical buttons, for good measure. SensorTags are often used as the basis for weather stations and health monitors, but what possibilities exist for the integration of this technology with design tools? This session will look at the data you can get from a SensorTag and how it might be integrated with Autodesk, Inc., software. We will cover the steps used to connect a SensorTag with smartphone and Windows apps, and we will examine use cases, such as using accelerometer and gyroscope data to manipulate the current 3D view in AutoCAD software.

Learning Objectives
At the end of this class, you will be able to:

- Understand the SensorTag development kit’s features and capabilities
- Discover which code samples are available for working with SensorTag
- Learn how to integrate SensorTag with an AutoCAD software product
- Learn how to use noise filtering to improve the quality of data received from the SensorTag

About the Speaker
Kean Walmsley has been with Autodesk, Inc., since 1995, working for most of that time for Autodesk Developer Network in a variety of roles and in a number of different countries. Kean’s current role is software architect for the AutoCAD software family of products, and he continues to write regular posts for his popular development-oriented blog, Through the Interface. Kean currently lives and works in Switzerland.

kean.walmsley@autodesk.com
Introducing SensorTag

SensorTag is a $25 device containing a number of sensors – an accelerometer, a gyroscope, a magnetometer, a thermometer, a hygrometer and a barometer – that communicates to a monitoring system (whether an iOS or Android mobile device or a Windows or Linux PC) via Bluetooth 4.0 (also known as Bluetooth Smart or Bluetooth Low Energy – BLE). Texas Instruments have packaged their CC2541 sensor platform in a consumer-friendly package with the intention of driving adoption from app developers who will use it to create solutions for health monitoring and the Internet of Things (just to name a couple of “hot” areas).

The device is powered by a single CR2032 coin cell battery, which is enough to power the low-power sensor array and Bluetooth communications for several months (this will depend on the number of sensors used and the frequency of communication, of course).

Developers might use the device to monitor the temperature of cooking pans and hot drinks, to track the location of keys using the newer iBeacon capability, or even to test how level a surface is.

At the core of the SensorTag is the CC2541 chip. We'll see later on that this is also used to power other devices, too.
Here’s a block diagram of the SensorTag device, outlining the various components:

Here’s an assembly diagram alongside a close-up of the SensorTag’s internals, showing where each of the sensors resides:
Applications for 3D design apps

While the SensorTag contains some interesting sensors, those that are of most interest for controlling 3D design applications are the spatial sensors. Just like many other devices, SensorTag contains an IMU – or inertial measurement unit – which consists of 3-axis accelerometer, gyroscope and magnetometer components. The IMU provides accurate information on where the SensorTag is located in 3D space.

This creates some interesting with respect to using SensorTag to control model navigation inside AutoCAD, for instance. Movements of the SensorTag can drive view changes in AutoCAD.

The same mechanism could almost certainly be implemented using a smartphone rather than a SensorTag, for instance, but there are certainly advantages to building a cheap, dedicated device for this.

Getting to know your SensorTag

The simplest way to connect with a SensorTag device and get to know its services is to install the SensorTag app for iOS or Android:

This will allow you to experience the data streaming from the device and determine whether it’s of interest to your specific domain.
Using SensorTag as a Low-Cost Sensor Array for AutoCAD

Transitioning to the Desktop

**Hardware**

Connecting SensorTag with a Windows machine is more complicated, however. You need to use a compatible Bluetooth USB dongle, the TI CC2540 USB dongle. This can be bought for $49 from the TI website.

You’ll also need a [CC Debugger](#) to be able to flash the CC2540 device with the required firmware, which is also available for $49.

These two components are bundled together in a couple of different packages. The first possibility was the [CC2541 Mini Development Kit](#), which – for $99 – includes a dongle, a debugger and a “keyfob” device with a few basic components (LEDs, a buzzer, two buttons and an accelerometer).

The second, the [CC2541 Bluetooth Smart Remote Control Kit](#) – for $50 more at $149 – includes a Smart Remote Control which contains a gyroscope, an accelerometer and a many more buttons (it’s a remote control, after all).

**Software**

Besides this hardware, there is some software needed to communicate with the SensorTag device.

[BTool](#) is essentially the driver for the CC2540 dongle: it allows you to map a virtual COM port that will be used for communication between the software and the SensorTag (via the dongle, of course).

[BLE Device Monitor](#) is the desktop equivalent of the tool we saw earlier for iOS/Android. It allows you to discover and understand the SensorTag’s services, as well as to upgrade the firmware (providing it already has appropriate firmware – for earlier versions you’ll need SmartRF Studio).

[SmartRF Studio](#) is a tool you can use in conjunction with the CC Debugger to flash or reflash CC254x devices (e.g. the dongle or the SensorTag).
Coding with SensorTag

The main resource available for developers interested in developing applications based on SensorTag is the SensorTag wiki: http://ti.com/sensortag-wiki

This lists links to a number of samples – including one to my blog – showing how to connect with SensorTag. As you’d expect, there are a number of mobile-oriented samples and toolkits as well as two main samples of interest to Windows developers.

SensorTag C# Library

This library – posted at http://sensortag.codeplex.com – simplifies the creation of SensorTag apps for the Windows Store. It makes use of APIs available in Windows 8.1, which clearly creates a platform dependency. That said, the APIs are clean and modern, allowing you to use language constructs such as async/await, for instance.

If you’re creating a Windows Store app for Windows 8.1 or higher, this is the way to go.

BLEHealthDemo

This is the project used for the AutoCAD integration prototype. The app was in development on .NET until Beta 10, but development has apparently since shifted across to Android. The existing codebase is probably 50% complete (according to the developer, and based on their milestones), but has some useful core code, despite the need for significant restructuring.

Here are some snapshots from this app. This first one shows the main page with a lot of UI:
It’s this tab that shows some of the potential for our purposes: we have a 3D view representing the orientation of the SensorTag device.

Integrating SensorTag with AutoCAD

The main goal of this sample it to manipulate the current AutoCAD view based on spatial input from an external device (in this case we’re using SensorTag). For this simple scenario we’re going to focus on using the accelerometer data.

This gives us greater simplicity and reduced power consumption, as we’re only enabling input from a single sensor. That said, it doesn’t provide the best accuracy: we’ll see later what might be done to improve the situation.

This application can easily just be command-line based: we will interact with AutoCAD via a jig, but there’s no need to provide a GUI. During the jig we’ll poll SensorTag for input, using the data to modify the current 3D view appropriately and then forcing another cycle by pumping a Windows message. This is a similar approach to the one we saw with the AutoCAD sample integrations for Kinect and Leap Motion.

Understanding 3D input

An Inertial Measurement Unit (IMU) contains three main components:

- Accelerometer
  - This measures linear acceleration – including that due to gravity
Using SensorTag as a Low-Cost Sensor Array for AutoCAD

- Gyroscope
  - Measures relative changes in rotational orientation
- Magnetometer
  - Measures orientation relative to Earth’s magnetic field

The best results are obtained via sensor fusion – combining the results of all three sources of input – but clearly is more complex to implement.

Some accelerometers provide higher level roll, pitch and yaw values, but SensorTag’s does not: we have to calculate this for ourselves.

What we need from the SensorTag

We’re going to subscribe to two main services from the SensorTag: the accelerometer and the “simple keys service”, which tells us when one or other of the buttons is pressed.

For the accelerometer, we need to set the update frequency to the minimum interval, so we get called 10 times per second with sensor data.

The standard orbit mode – which is active when no buttons are pressed – really only needs X and Y values from the accelerometer: we’re going to ignore the Z value, as adding “yaw” makes the experience a lot less predictable. For the “zoom & pan” mode – active when the left button gets pressed – we’re going to use all three axes. We’re going to estimate the distance travelled along X and Y directions to drive panning of the view and along the Z direction for zooming.

The right button will simply reset the view to the original one: this is helpful as some rotation does creep in, over time.

Determining the distance travelled

Our accelerometer provides the acceleration along three axes. To get the distance travelled we need to integrate twice: the first integration provides the velocity at a particular moment which we need to integrate again to get distance.

Thankfully we don’t actually need the data to be particularly accurate: the main thing is to determine the general direction, although some measure of magnitude would be nice. This approach of doubly integrating accelerometer data is notoriously inaccurate: small amounts of noise – and issues such as device tilting – can lead to big issues in the data.
We’re going to use a fairly crude but effective integration technique: the trapezoidal rule:

\[ \text{Area} = (t_2 - t_1) \left[ \frac{f(t_1) + f(t_2)}{2} \right] \]

Given the frequency of our sampling, this will prove accurate results. But if the sensor data is noisy we’ll suffer from GIGO (Garbage In, Garbage Out).

**Improvements and Applications**

Accelerometers – while accurate in the longer term as they don’t suffer from drift – can be quite noisy in the short term. It’s not possible to accurately track spatial position over a multi-second period using accelerometer data alone.

Gyroscopes measure relative rotation and so suffer from drift in the longer term (while being accurate in the short term).

For more “serious” applications, some kind of filtering should be applied to the data:

- Complementary filters
  - The simplest filter type, assigns a higher weighting to the previous data and a smaller weighting to the newly arrived data
- Kalman filters
  - These measure state over time, assessing the quality of the newly arrived data relative to the existing set. Clearly more complicated to implement
• Mahony & Madgwick filters
  o These more advanced filters take 3-axis data from each of the accelerometer, the gyroscope and the magnetometer to get ultimate positioning accuracy

.NET implementations are available of the Mahony & Madgwick filters at:

http://www.x-io.co.uk/open-source-imu-and-ahrs-algorithms

The tricky part will be making sure the various data streams are synchronized in time.

Moving forwards

IMUs and spatial data are everywhere: integrated into all modern smartphones, but also in custom chips that are being placed into all kinds of devices. As an example, you can access the ‘deviceorientation’ event in an HTML page on a smartphone to effectively implement a VR application based on Google Cardboard. This event provides handy roll, pitch & yaw values.

It seems as though every day there are new IMU projects being launched: check FreelMU or IMUduino as examples.

As spatial device input becomes more widespread, the tools to work with it will become easier: filters should be components you can integrate painlessly, for instance.

SensorTag is just one way to get started integrating this kind of input into your apps. It’s worth starting to think about how you might make use of spatial data – or data coming from SensorTag’s other 3 sensors, for that matter. There are definitely interesting use cases for integrating accurate spatial data into a 3D modeling environment.