Starting with sensors

[cross-posted from sensornews.com]

I'm at the iHub Nairobi today with a bunch of sensors (thanks for the loan, Brck team!), because some of the Kenyan ideas for today's Space Apps Challenge projects are sensor-based. Those projects didn't happen, but I've been having some very interesting chats with people about the hardware we have here, about their own use of hardware, and about why coders aren't including hardware in their projects.

Aside from utility (not every project needs sensors, just as not every project needs a web interface), the two big blocks appear to be unfamiliarity and fear. First, the fear: generally that using hardware will be hard to learn, or that you'll break equipment irreparably. And the familiarity: coders are used to software, and hardware can seem very different to software, at first.

The fear: I suffered from these fears too, as I got back into hardware. That combination of "oh grief I'm going to break it" combined with "but what could I possibly do that's useful with this stuff".   The answer is really quite simple. Buy some kits, start putting them together, and learn from that a) what works and doesn't, b) why, how, when hardware is useful , and c) how to make your designs more useful. Yes, things break; no, the outputs aren't always perfect, but the point of using kits is to learn, and fast. If that sounds familiar, it's because it's the same ethos that drives agile software development: build, fail, learn, build better until it fits what you need.

And speaking of familiarity: electronics is really coding with things. You have basic units (components), basic things you can do with them (connect together, run voltage through them, read outputs from them etc), that you design together into a working system. And if you're using microprocessor-based kits (Arduino, RaspberryPi etc), then you really *are* coding, because you're programming the microprocessor to send signals, respond to data inputs etc.

I'm doing this the easy way. I'm starting with components that slot together and have code that's already written for them: the grove sensor series.   The kit you need to start getting sensor results with these is:

- Arduino Uno (or clone) - $30
- USB cable: USB type A to type B- $2
- Grove base shield - $9
- Grove sensor (any type) - $3 upwards

Erm, that's it.   You'll need to download the basic Arduino software, and find the example code for your sensor in somewhere like the Seeed wiki, but for about $45, you too can start experimenting with sensor data. I've just left a stack of Grove sensors and a couple of Arduino Unos at the Brck Nairobi office; I'm using the same stack in New Jersey so we can compare results and ideas. We've already got data from this experiment – proving that the Brck office is dustier than the Ushahidi office nextdoor isn't a great leap forward in knowledge, but taking these sensors out into the field, and getting comparable data from places without good coverage from 'official' air quality monitors is.

The journey from here involves lots of placing sensors and learning how they fail, what they do under stress, and what their limitations are. It's also, ultimately, to start reducing the number, size and price of components needed to produce usable and useful sensor data, to learn from pioneer communities like Public Laboratory and RiverKeeper, and to make it just 'normal' to include sensors in system designs and 'normal' to plug them into existing equipment like Brck and mobile phones (I have a Geiger counter that plugs into my phone's audio port – I'd love to see more of that sort of reuse out there).

And now, back to thinking about questions like "could you build a gas sensor into your clothes". I just happen to have an MQ-5 gas sensor in front of me, and am thinking about what it would take to get from there to an alarm ringing on my phone...