# Things at ThingM

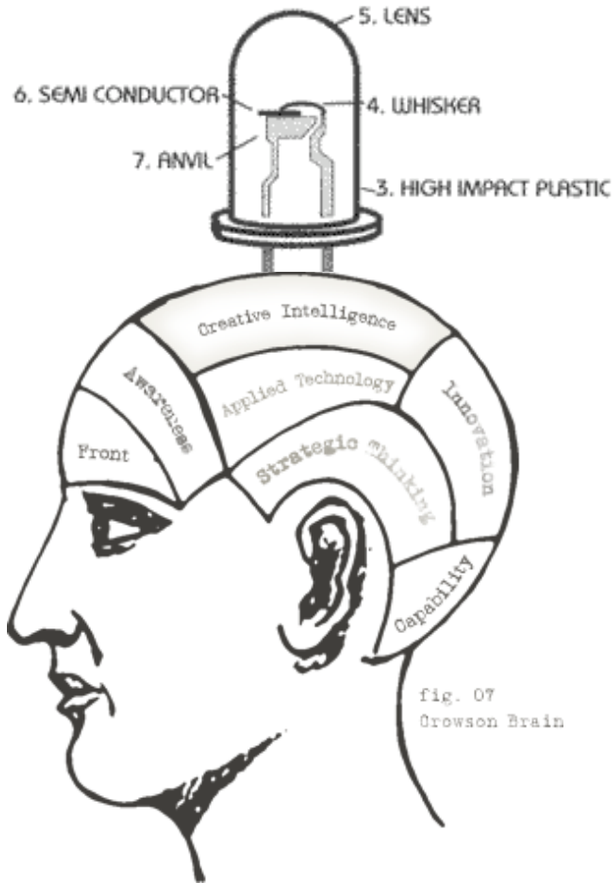


fig. 07
Crowson Brain

8 Nov 2007 - UC Berkeley iSchool INFO290 - Tod E. Kurt

# Random Walk through ThingM's Brain

Or, what I'm talking about today

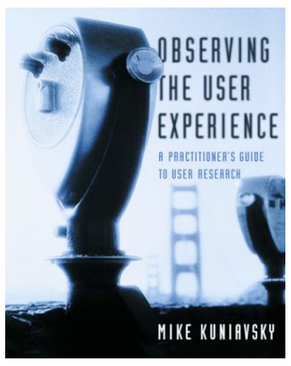
- **Computation as Material**
- **Smart Interface Components**
- **Reversible Hacking**
- **Technology Sketches**
- **Informational Objects**

Hi I'm Tod.
I think you've been using some my Arduino stuff.
Here's the main topics I'll be talking about today.
Some are kind of abstract and theoretical, others are extremely practical.
Some will probably be pretty obvious to you in this class.

# ThingM Humans

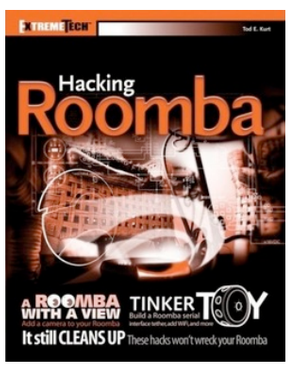
## Mike Kuniavsky

– user experience & design expert
– founder of Adaptive Path
– author “Observing the User Experience”




## Tod E. Kurt

– ex- bespoke space probe engineer
– founding developer of GoTo.com / Overture
– author “Hacking Roomba”




**and a large on-demand pool of designers, fabricators, assemblers**

First a little bit about ourselves.

My business partner is Mike. We’ve been working together since 1994, when we both worked creating one of the first ecommerce sites on the Web. It beat Amazon by several months, it was the hot sauce shop HotHotHot.

Mike went on to work with HotWired and designed HotBot. He then founded Adaptive Path, the pre-eminent consulting firm for web user experience and IT design. Adaptive Path is hired by Fortune 50 corps., non-profits, and web startups.

I became a founding developer of GoTo.com, which later became Overture, and sold to Yahoo!. Before the web (and a little during), I worked on a few hardware engineering projects for robotic space cameras. One of these was the ill-fated BlastOff project with Jim Cameron. Others were cameras that actually went to Mars.

ThingM is purposefully very light on in-house staff, utilizing the Net to hire the capabilities we need, when we need them.

# Companies we've worked with

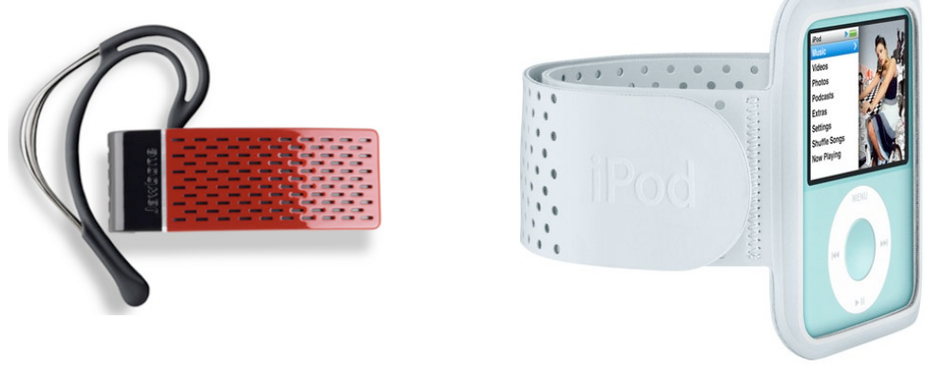

We've been fortunate to work with several pretty cool companies over the years. These are some of them.

# ThingM Concepts

- **Ubiquitous computing device studio**
- **Domestic & Wearable technology**
- **Intersection of the physical and informational**
- **Rapid prototyping & technology sketches**

household robots


wearable computers




ThingM is a “ubiquitous computing device studio”. Our goal is to re-invent how people approach objects in their everyday world.

Eventually all objects will have embedded intelligence. And talk to each other. Future domestic & personal devices may look nothing like current ones, like the coffee maker, vacuum cleaner and telephone shown here. If a coffee maker and vacuum cleaner could talk to each other, what would they say?

Currently we’re exploring how physical objects are being reflected in informational spaces. And we’re this doing via various types of rapid prototyping.

---

Wearable tech is already here: people carry about 4-5 computers on their person already:
RFID transponder in car key, bluetooth headset, cell phone (counts as two: baseband radio, smartfone functions), iPod, PDA. Why does the "net connection" baseband radio need a UI. It's just a brick, stick it in your bag, pocket, shoe, etc.

For example:
- a bracelet that lets you see Wi-Fi
- a chocolate box that plays back shared video memories
- a bed that provides an RSS feed of sleeping habits
- a bookshelf that knows your books

# Ubiquitous Computing


**1989: $900**

Quantities of 1000
33 MHz, 20 MIPs


**2007: $0.53**

Quantities of 1
20 MHz, 20 MIPs

*Moore's Law works inversely on price*

*computation and networking are becoming so cheap as to be effectively free*

Let's talk a little about what I mean by ubiquitous computing. We think about it in terms of Moore's Law flipped. Instead of how processing power increases, we're interested in how a given amount of computation is constantly getting cheaper. These tiny chips don't get as much press as the big ones, but they sell more units and are in more devices.

They can't play the latest games, but they can process a lot of information and you can stick them anywhere. Embedded computation has become a cost-effective competitive advantage among manufacturers.

---

"There's another way to think about computers. It's called ubiquitous computing. Driven by the same forces that make today's laptops, it has all the many GigaHertz that we all hear about but it's on the less glamorous end of the semiconductor industry. Low-end chips may not get as much play in the media, but they get CHEAP at the same rate as high-end ones get powerful. So today a chip with the same power as a top-end processor in 1989 costs less than a dollar, runs on a fraction of the electricity and is significantly smaller." --mikek

"A processor like this won't run the latest games, but that's not the point. It can still process a lot of information and you can stick it inside everyday objects. This means is that embedded information has become a cost-effective competitive advantage among manufacturers.
It's like discovering a new kind of material. Now, adding information processing to a thing is akin to choosing whether to make it out of rubber or plastic." --mikek

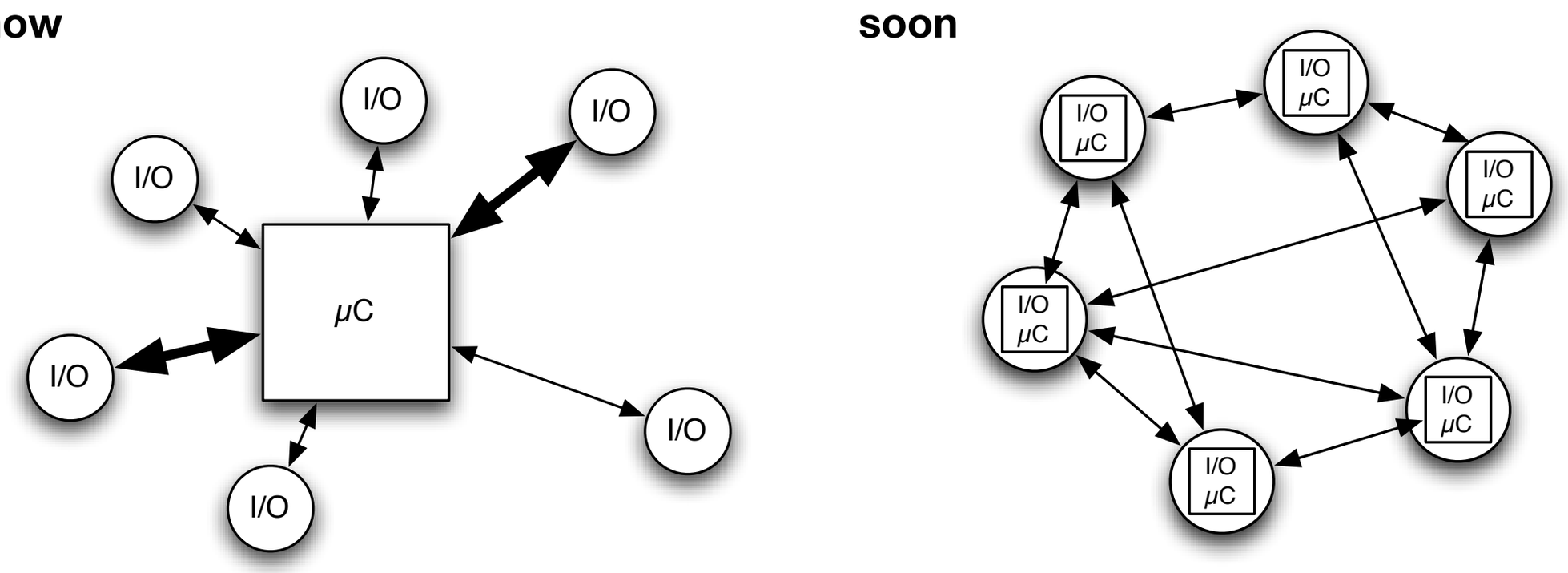
# Computation as Material


*by grewlike on flickr*

It's like discovering a new kind of material.  Now, adding information processing to a thing is akin to choosing whether to make it out of rubber or plastic.

Computation becomes simply another potential attribute of a product. A product differentiator. Fabric or leather? nylon or lexan? smart or not?

# Computation is Suffusive




And it won't simply be that objects will become smart.

For those objects currently with some intelligence, we'll see that intelligence spread out and move towards the boundary of an object. No more "central" processing units, hidden away in a protective core. Instead a more distributed, network-based model.

That is, computation will suffuse itself throughout an object.

Cell phones are a heading down this path: the baseband radio and graphic display both have their own processors and talk to the main processor over a bus.

Desktop computers also have an inkling of this. With high-level bus protocols like USB and WiFi being an integral part of even the simplest peripherals, even something as simple as a mouse today has 12 MHz RISC CPU.

(See Cypress Semiconductor's Low-speed peripheral controllers like the CY7C63221A)

# Smart Interface Components

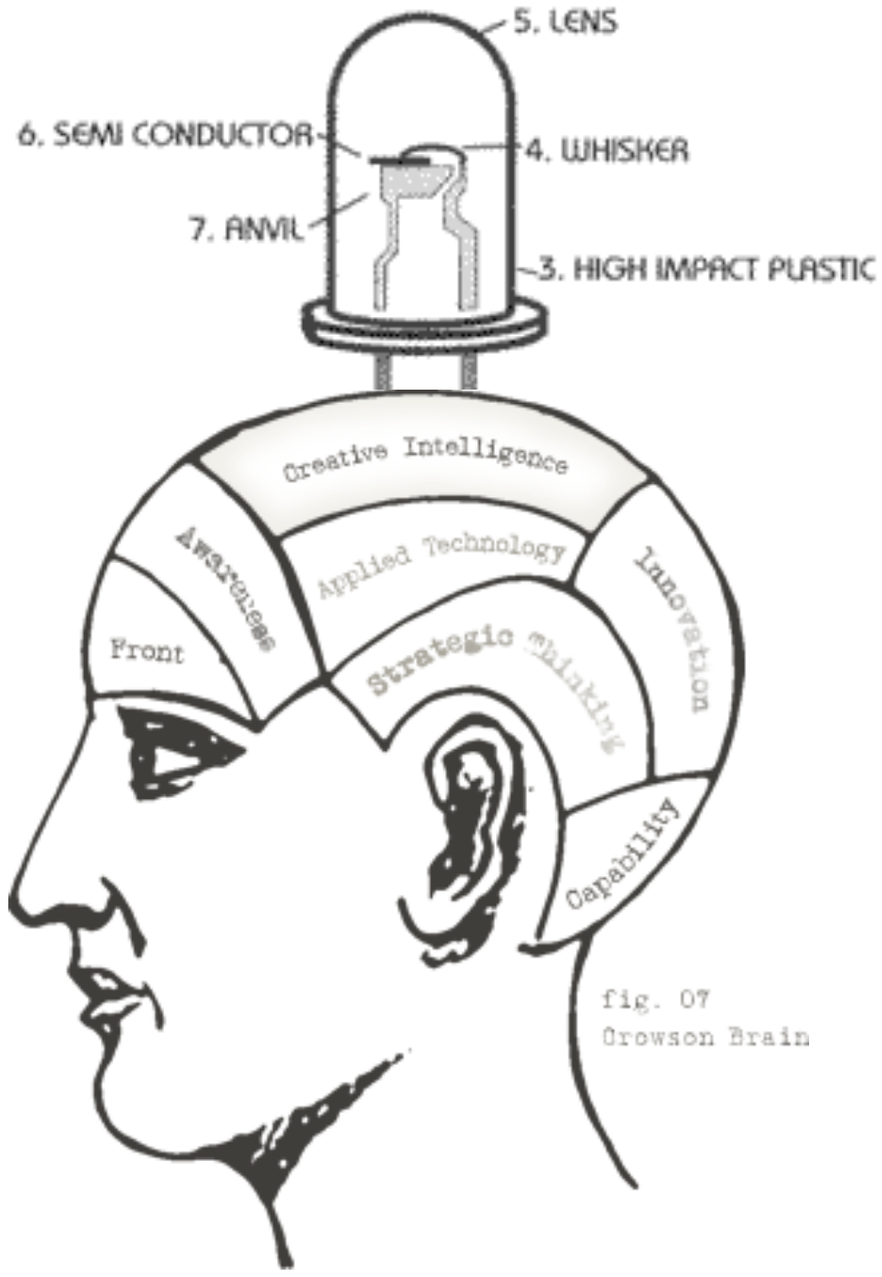



fig. 07
Crowson Brain

One avenue we're exploring regarding suffusive computation are what I call "smart interface components".

# Interface Components?

- Sensors
  - buttons / knobs
  - light
  - sound
  - force
  - proximity, location
  - etc.
- Actuators
  - motion / vibration
  - lights
  - sound

Interface components are the various input and output transducers and mechanisms that comprise the physical interfaces of an electronic device.  You know them.

# Interface Components?

- **Sensor Examples**
  - **switches**
  - **photodiodes**
  - **piezos**
  - **potentiometers**
  - **rotary encoders**
  - **accelerometers**
- **Actuator Examples**
  - **motors**
  - **LEDs**
  - **piezos / speakers**
  - **relays**

**...you know, the standard parts used in gadgets everywhere**

You've been learning out to hook them up to Arduino in this class.

# The Idea: Make them smart

- **Sensors and Actuators w/ embedded intelligence**
  **(and possibly additional sensors/actuators)**

- **Can then have meaningful conversations with them:**

  - **knob: “angle turned=42º”**
  - **button: “double-clicked”, “held for 5 seconds”**
  - **speaker: “play 440Hz”, “play Eb3”, “play quieter”**
  - **light: “#FF33CC”, “dim 30%”**
  - **motor: “spin at 29.97 RPM”, “torque is >20 lb-ft”**

Give each interface component it’s own processor.
Then you can have high-level conversations with them.

Do you *really* want to figure out how to make a stepper motor move?
Do you *really* want to create yet-another-key-debounce function?
Do you *really* want to remember what kind of back EMF diode & transistor to add to your motor circuit?

Push the boundary of intelligence to the interface.  boundary...interface...interface...boundary... hmm

# Why?

- **Create new affordances, richer interfaces**
- **Reduce implementation time**
- **Embed engineering knowledge in the device**

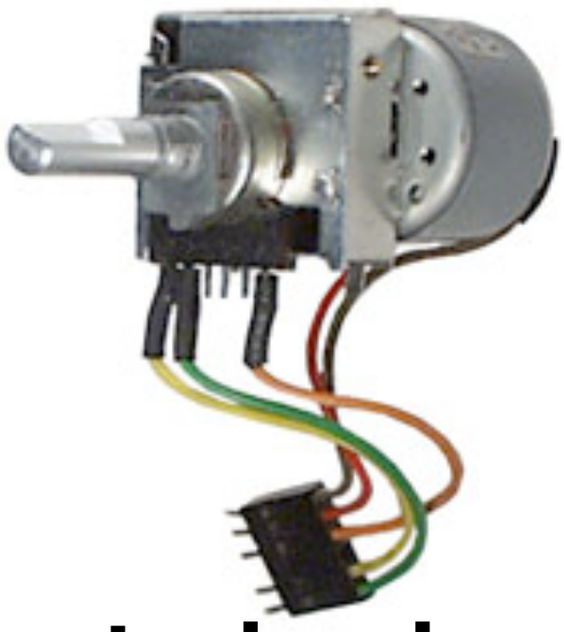
**doesn't this get expensive fast?**
**yes and no.**

When you no longer have to worry about the low-level details of an interface, you can worry about if the interface makes sense and is usable.
As an added benefit you get a working prototype faster.
And since these smart components encapsulate the interface-specific engineering needed, you can be assured you're getting the most out of a component.

# Some devices are close

*“composite interface components”*

**potentiometer**



**motorized pot**

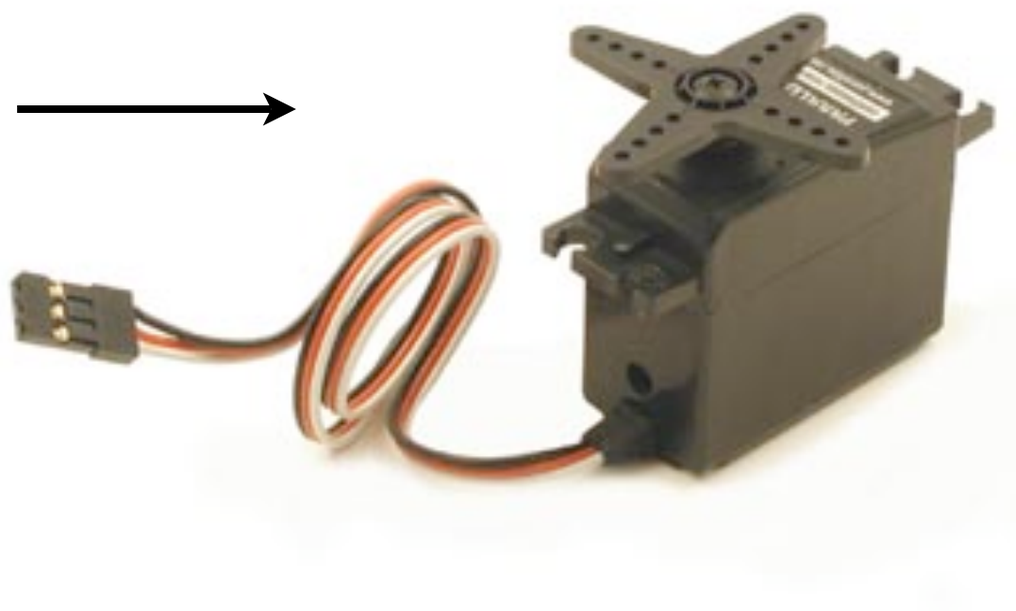
**provides haptic feedback**

**allows memorized presets**

**no logic though... yet**

Used to be if you wanted a motorized pot you had to build it yourself:
assemble the pot, motor, gearing, mechanicals to hold it all together. a real pain.

# Almost there



**DC motor** **hobby servo**

**Turns analog problem into a digital one**

A servo is just a DC motor with some electronics, a pot, & gears
DC motors are hard to design & program for, especially if you want to move it a given angle.
But servos are relatively easy.
(or use a stepper motor, even harder)

# This is more like it

**HD4470 text LCD**
**parallel control**


**graphic LCD**
**serial control**

HD4470 is easiest of LCDs to control, but physically wiring it up is a pain.
And it's only a *text* LCD.  Driving a graphic LCD by hand is impossible.

# Even some CE vendors are trying

**but currently very expensive (~1000x)**

Optimus Maximus OLED keyboard: http://www.artlebedev.com/everything/optimus/

# Design & Cost

- **Think production use, not prototyping.**

  **Design for: high-volume, low parts count, cost**

- **Needs to be <5x cost of equivalent dumb component**

- **Really more like <2x cost**

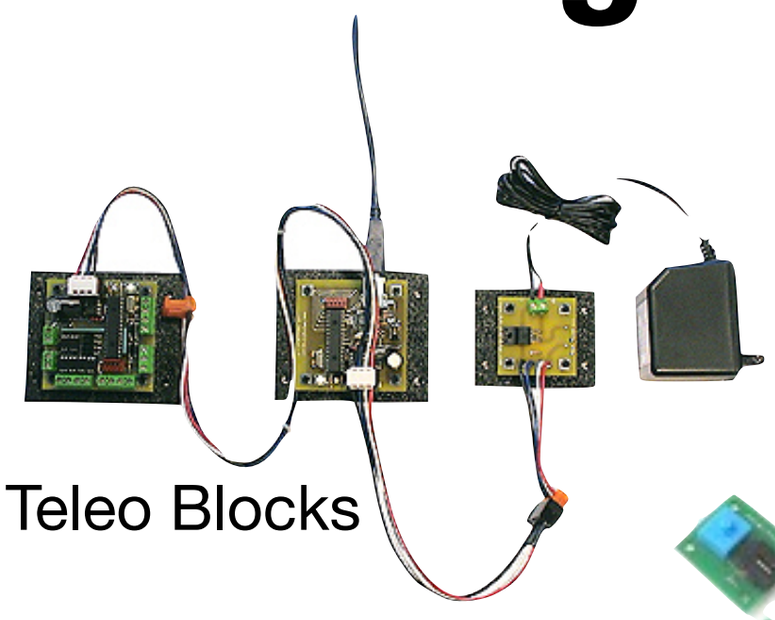
- **Benefits the prototyper too**

Let's think about implementing smart interface components as real products, at a fairly high-volume, not a home-made experiment.

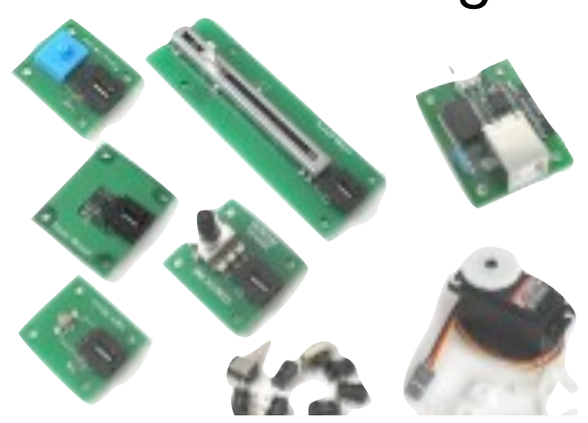
The cost of smart versions of current dumb interface devices needs to be sane, or no one will use them.

# Already exists in the hardware sketching world

- **Somewhat**
- **But bulky**
- **And Expensive**
- **Not made for production use**
- **Often not “smart”, just standardized connection interface**



Teleo Blocks



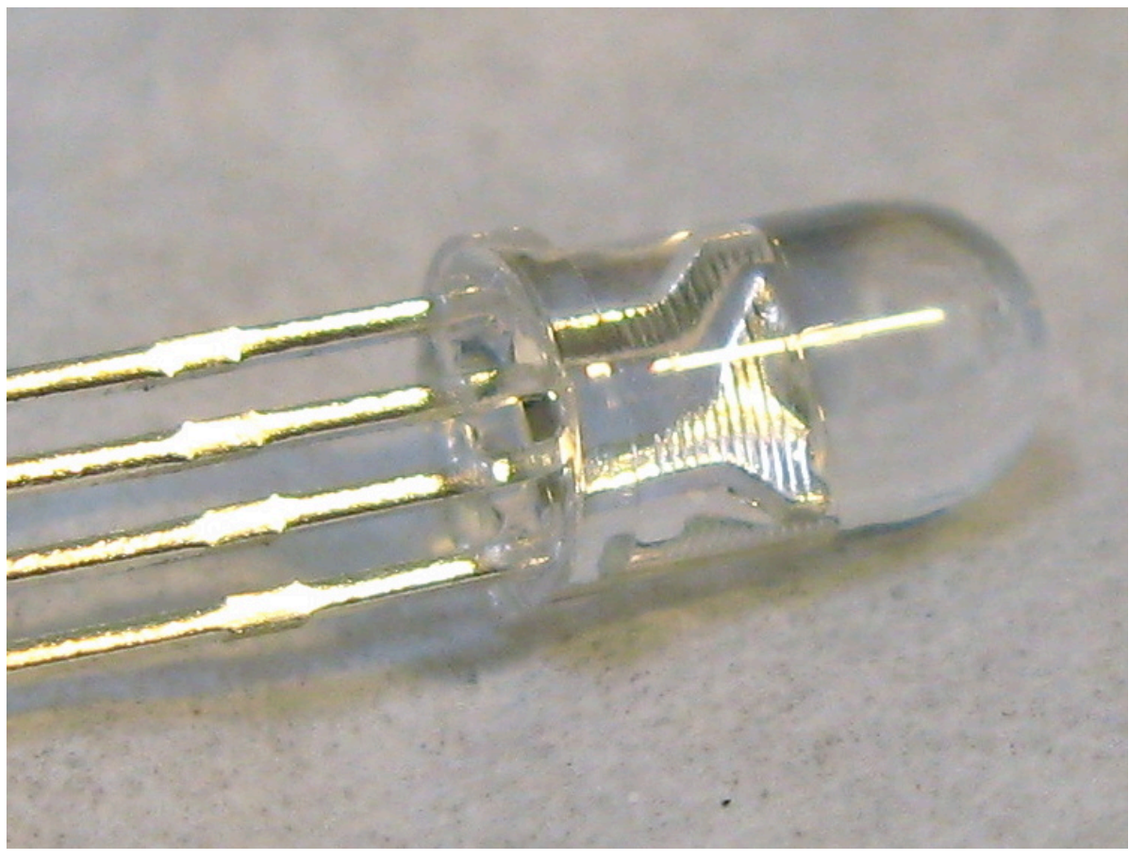
Phidgets

I-CubeX

Most all are meant as alternative input devices to a computer, usually running Max/MSP. Not meant for someone who is exploring creating a stand-alone device.

# Let's explore: Smart LEDs

- **An LED that can be any color, any brightness, at any time**
- **How: Take normal RGB LED, make it smart**
- **RGB LEDs are capable of full-spectrum color rendition**
- **Already composite devices: 3 LED dies in one epoxy package.**

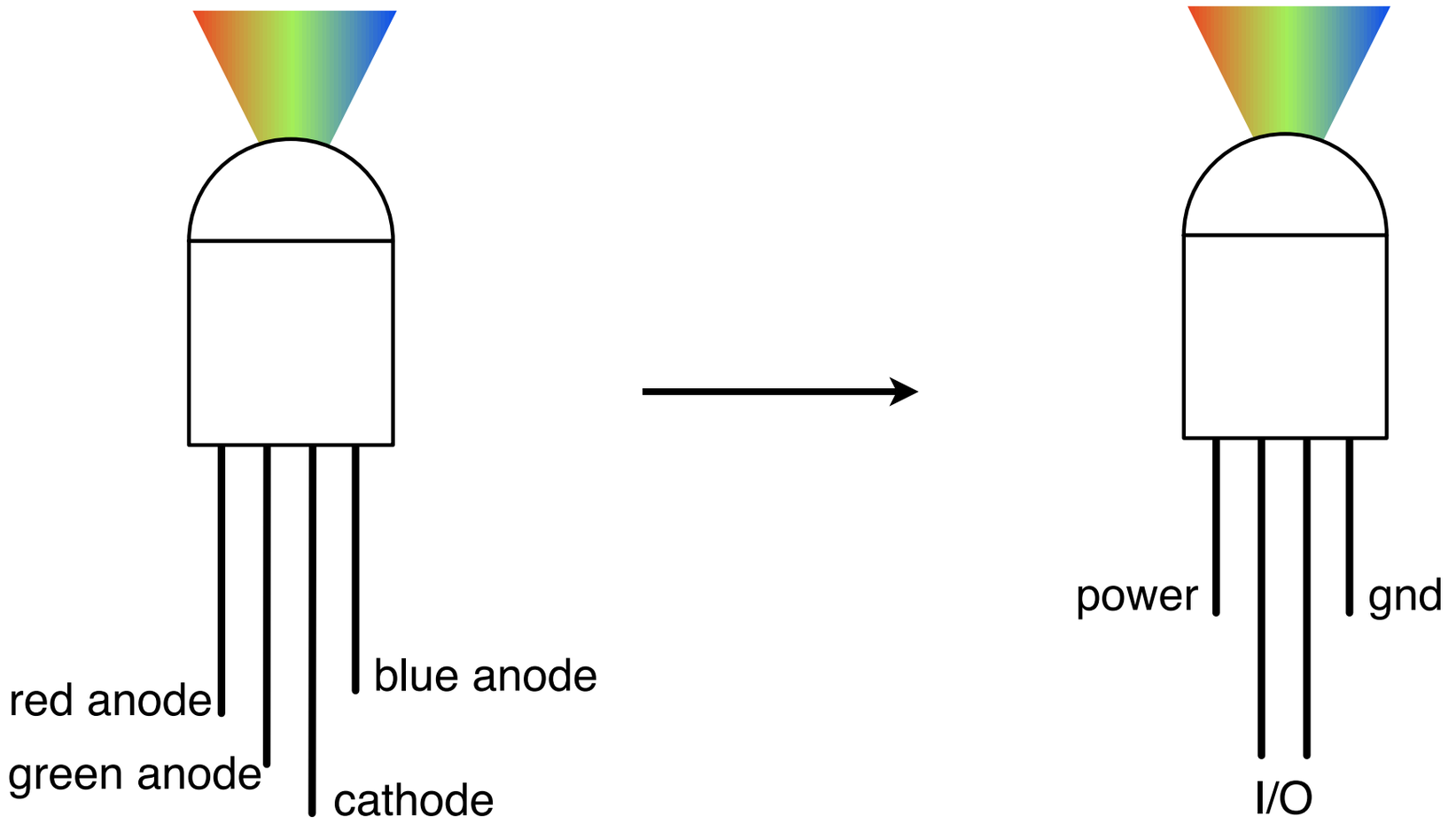


*"I can has brains?"*

Can something as simple and inexpensive as an LED be made smart? Imagine a drop-in replacement for a regular LED that can be any color, any brightness, and has built-in functions for light patterns, and so on.

Don't just add brains, add sensors. and actuators. :-)

# A Design for Smart LEDs




*The number of pins remains the same, but the semantics change.*

RGB LEDs have 4 pins.  A Smart LED would have an equivalent number of pins, but have more advanced meanings

Can we add intelligence to the standard “T 1-3/4” LED package?
Remember, think towards high-volume production uses.

# Smart LEDs are Good

- Normally, need 3 I/O lines & an interrupt to deal with time-critical RGB PWM calculations

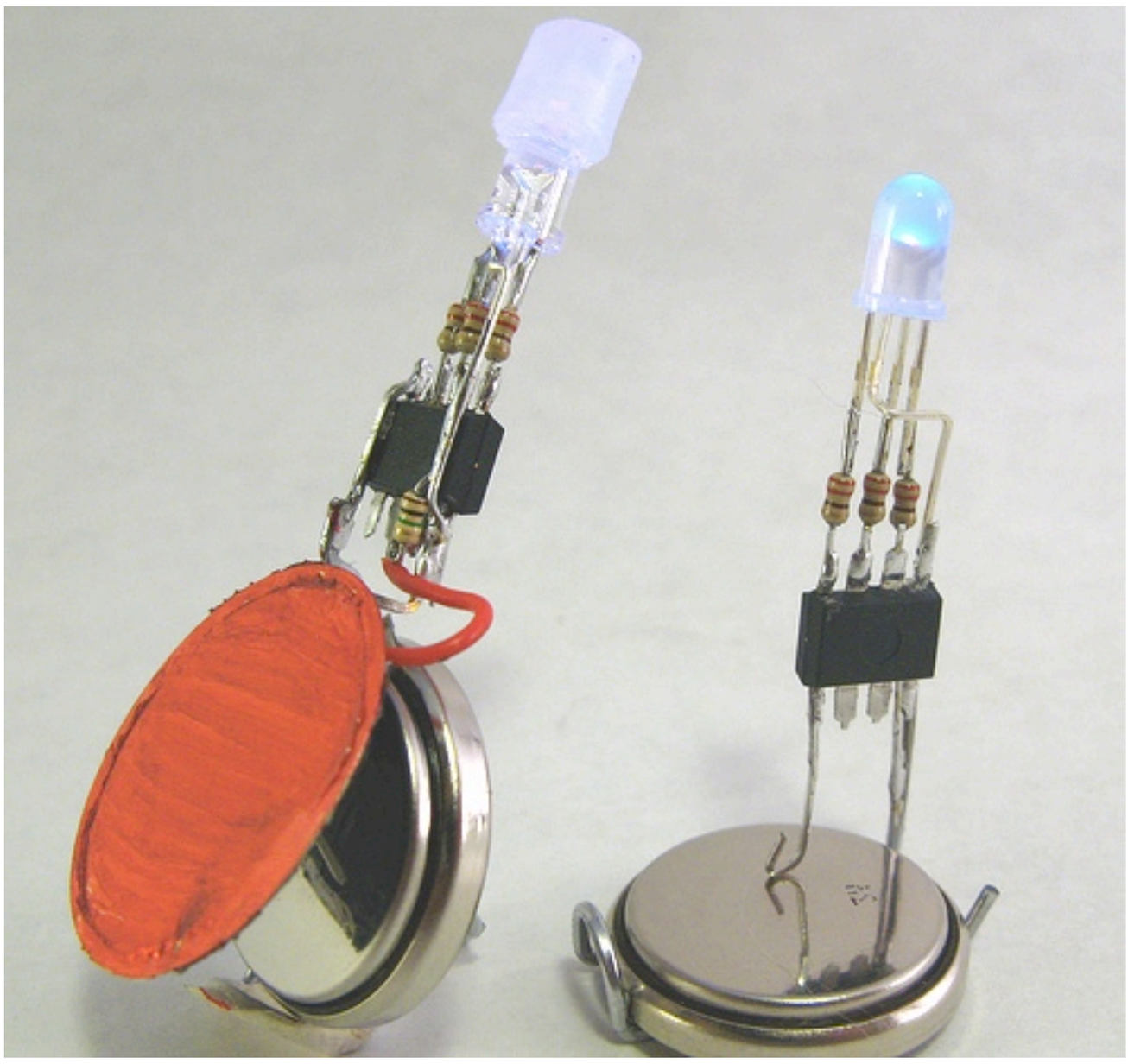
- No!  Let the LED do the work.

- No need to worry about current limiting, matching resistors, PWM, timing, etc.

- Embed a bit of color theory into the device: specify HSB or RGB values via input lines. Even as text: “#FF3366”

# Smart LED Prototypes

***hand-wired, thru-hole parts***



**with sensor** **without**

Here are some weekend design studies for a Smart LED.
What if we just add a microcontroller to an RGB LED?

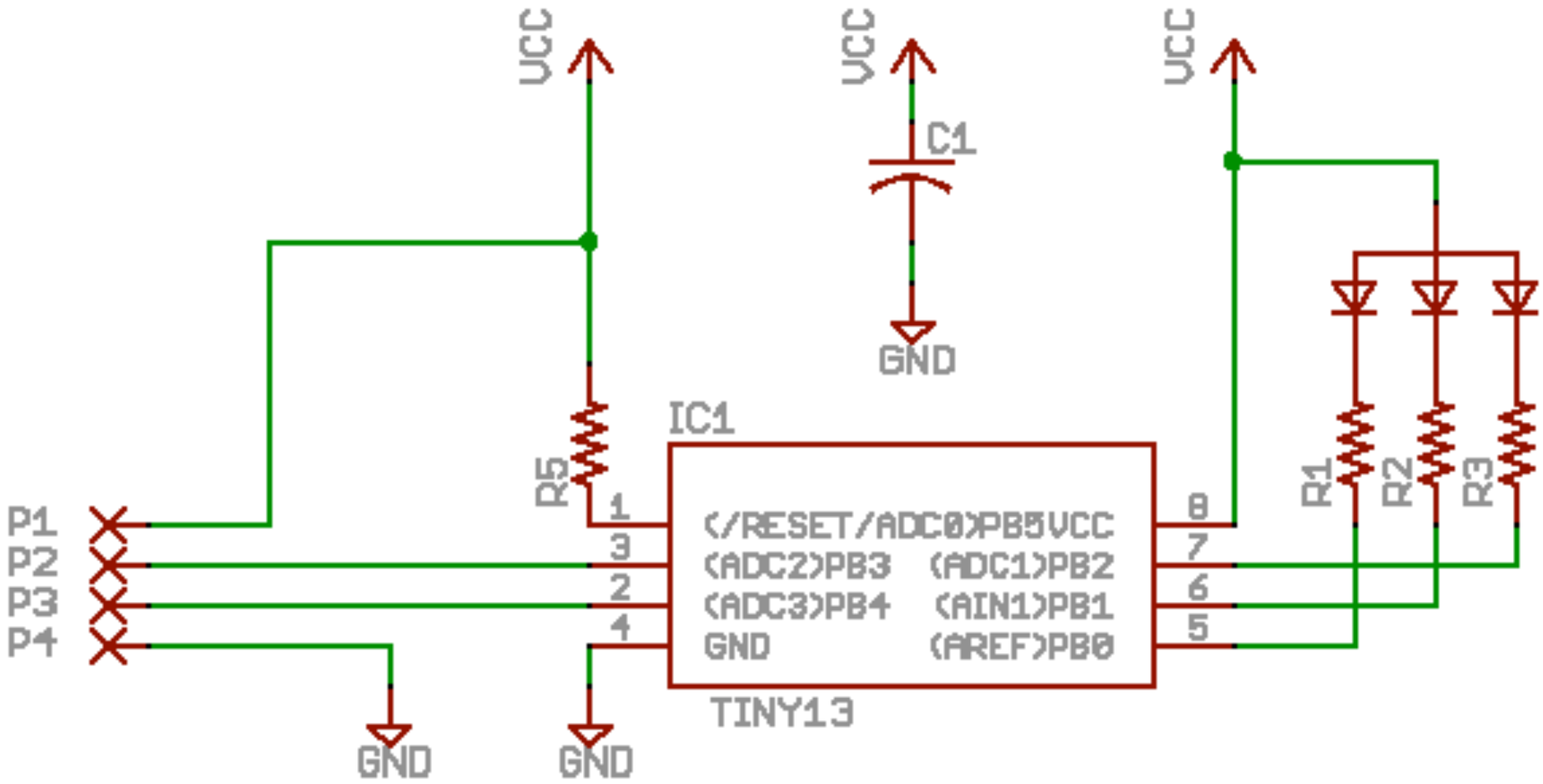
Originally from: http://todbot.com/blog/2007/03/25/smart-led-prototypes/
Also see Alex Weber's "Programmable LED" work: http://www.instructables.com/id/ELJXZZVX6JEYVZCV7K/

# Not new

- Ambient Orb
- Color Kinetics
- Triklets / Big Round Cubatron
- Every beginning Arduino sketch

There are some fairly elaborate and expensive precursors to the Smart LED idea.
And just about anyone who plays with embedded systems creates an RGB color fader.  But why devote your processor's resources to a task so thoroughly solved?

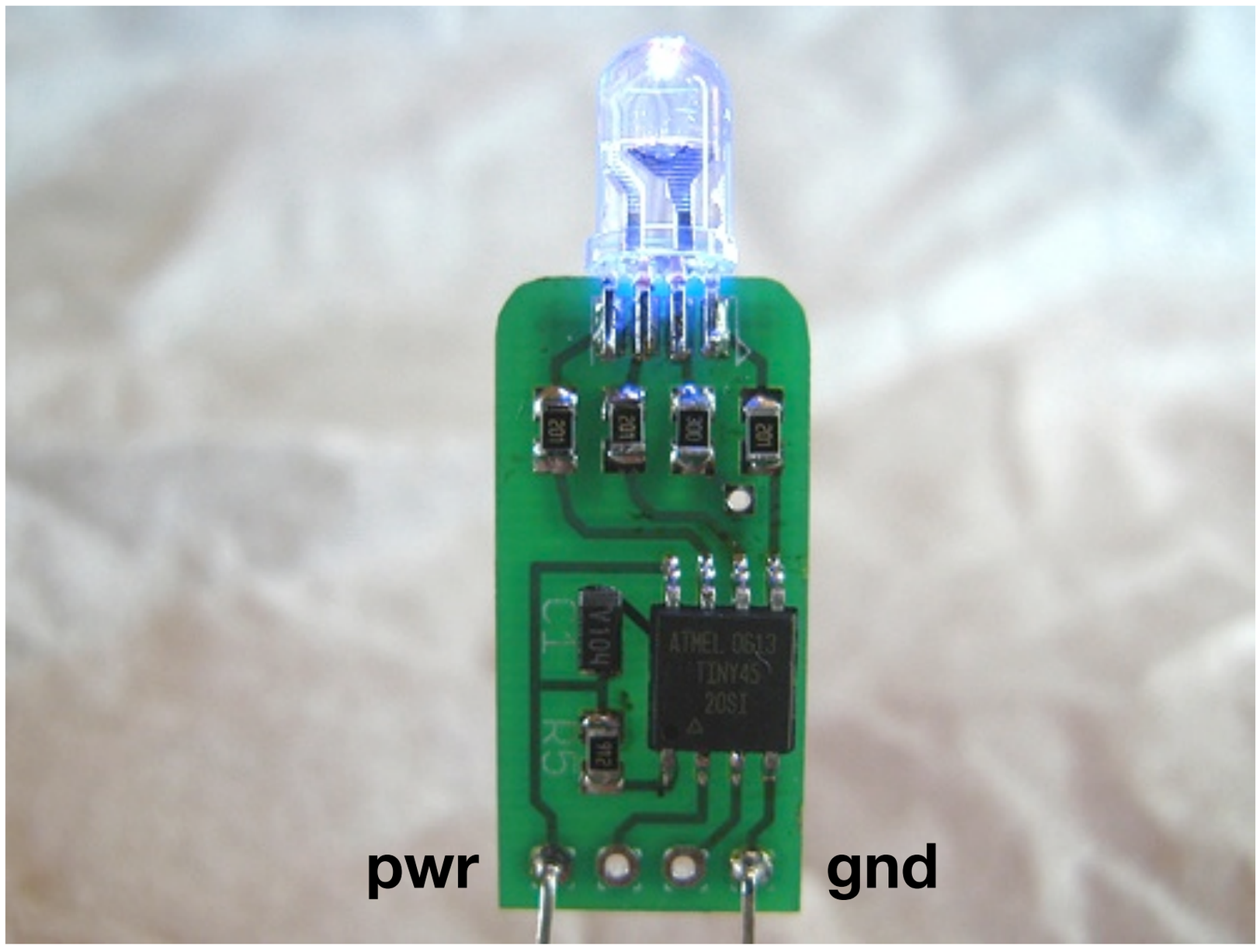
# Schematic is as expected




Nice thing about most embedded microcontrollers nowadays like the AVR is they need basically zero parts to operate.  This circuit operates at 4.8 MHz.

# Smart LED Prototypes

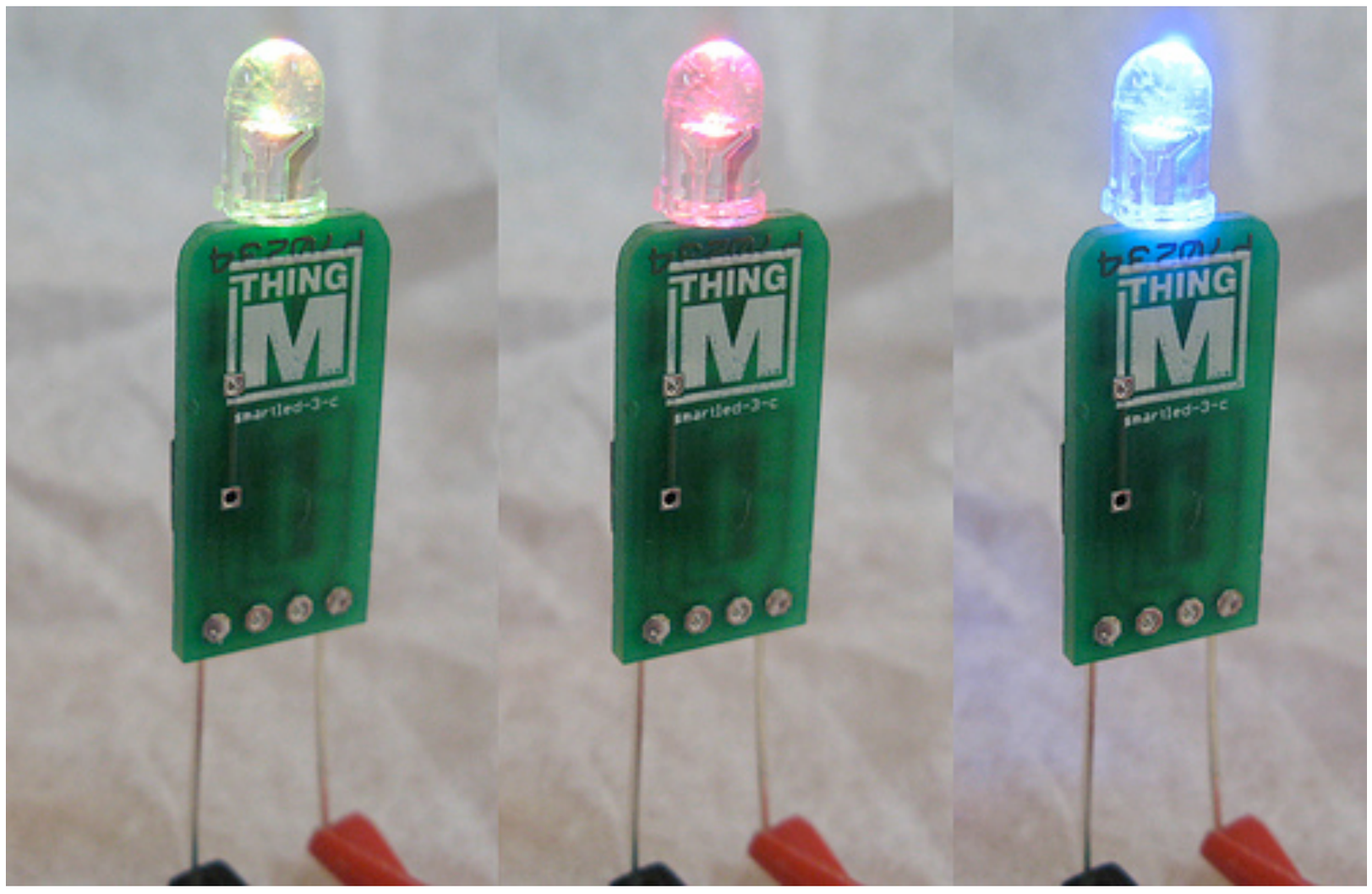
***mostly SMD, vertical orientation***




Inputs can be digital or 10-bit analog.
They could also be outputs.
Uses standard thru-hole RGB LEDs (common cathode)
This form factor allows high horizontal density, at expense of needing an inch of clearance in the back.

Although mounting a thru-hole component in a SMD fashion is something most assembly houses won’t do. (without something like 100k unit order)

# Smart LED Prototypes

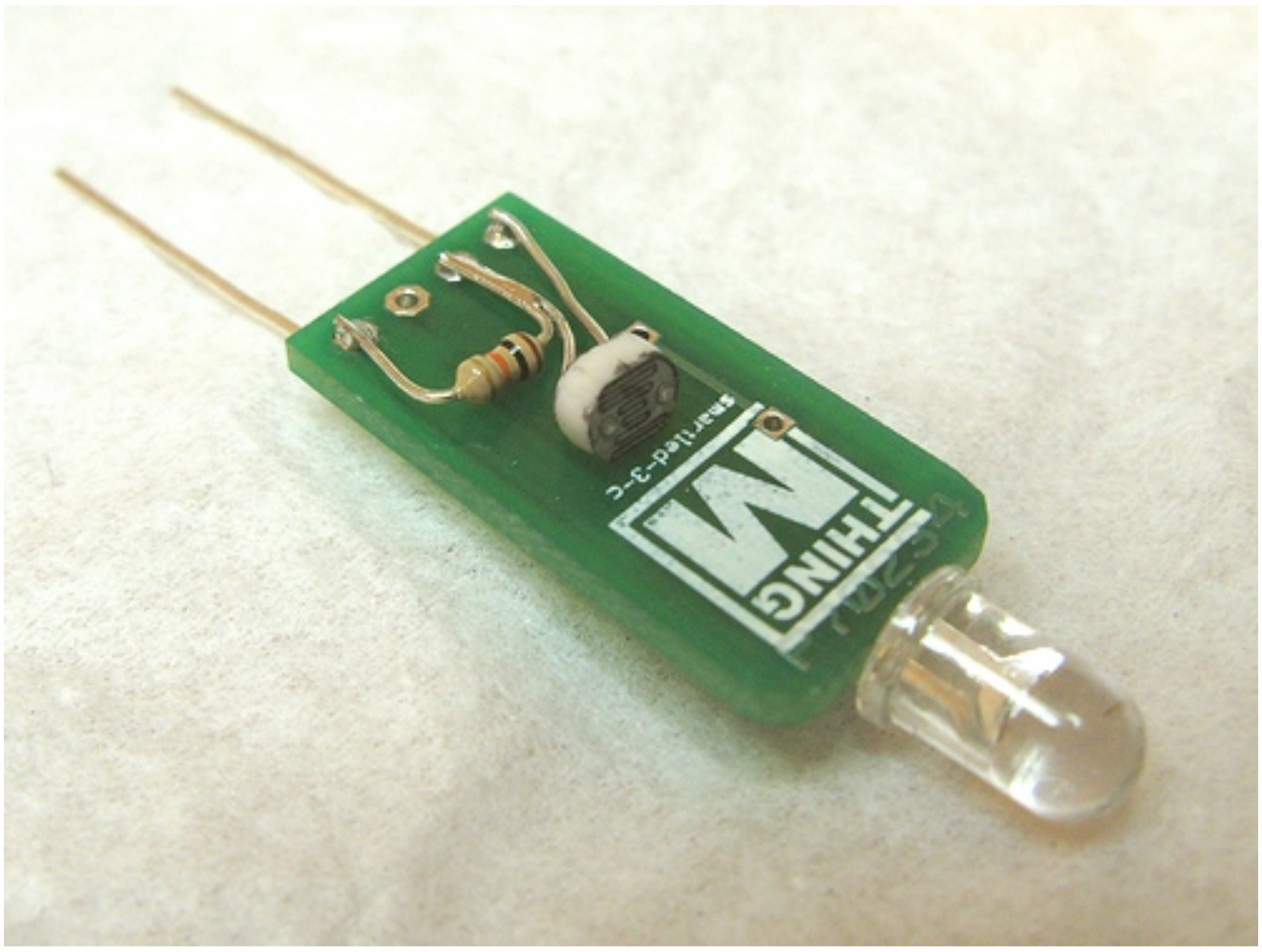



**color selectable by HSB or RGB**
**millions of colors in 24-bit color space**

It the photos above, it’s cycling along the outer rim of the hue circle.

# Sensors on Actuators

***"composite interface components"***



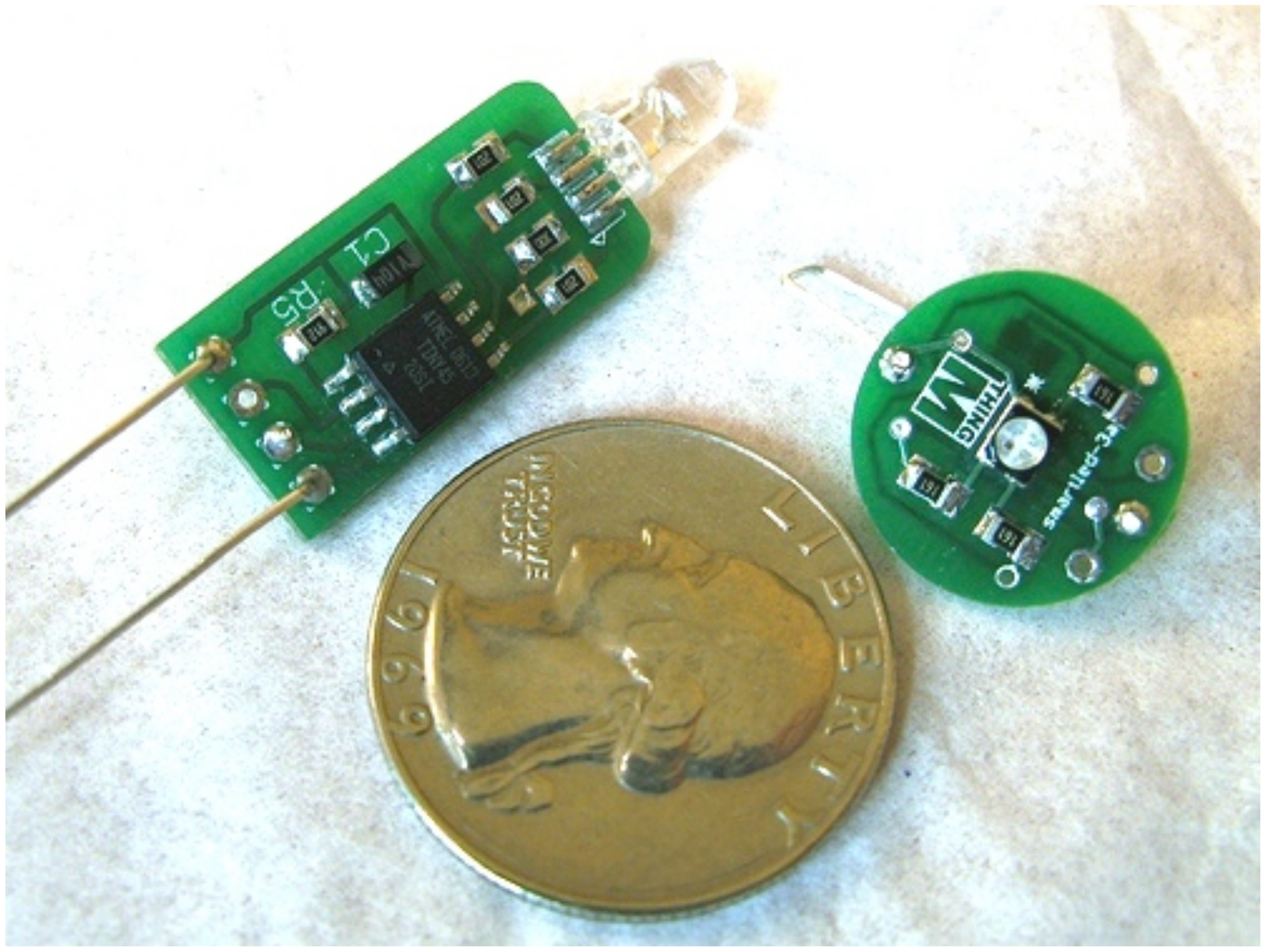

**e.g. emitted color temp. changes with ambient light**

"Composite interface components" is not a new idea either: see servos.
But add a simple sensor, like this photocell, to what we normally think of as a simple light and you can let the actuator partially close the loop.  And it's a sensor now!

If the point of the LED is to track ambient light levels as well as displaying a hue, part of that logic could be implemented in the LED itself.

Sorta like what Javascript form checking does for web apps. </webnerd>

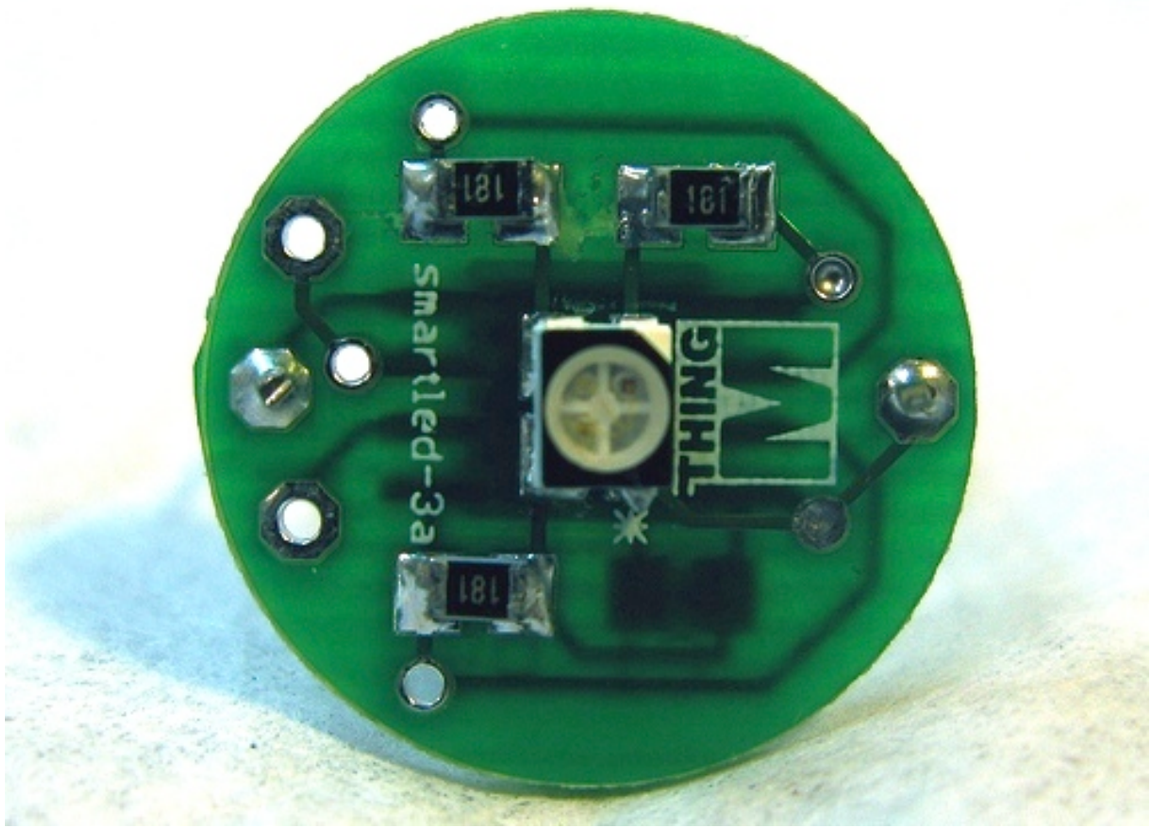
# Can you make it smaller?

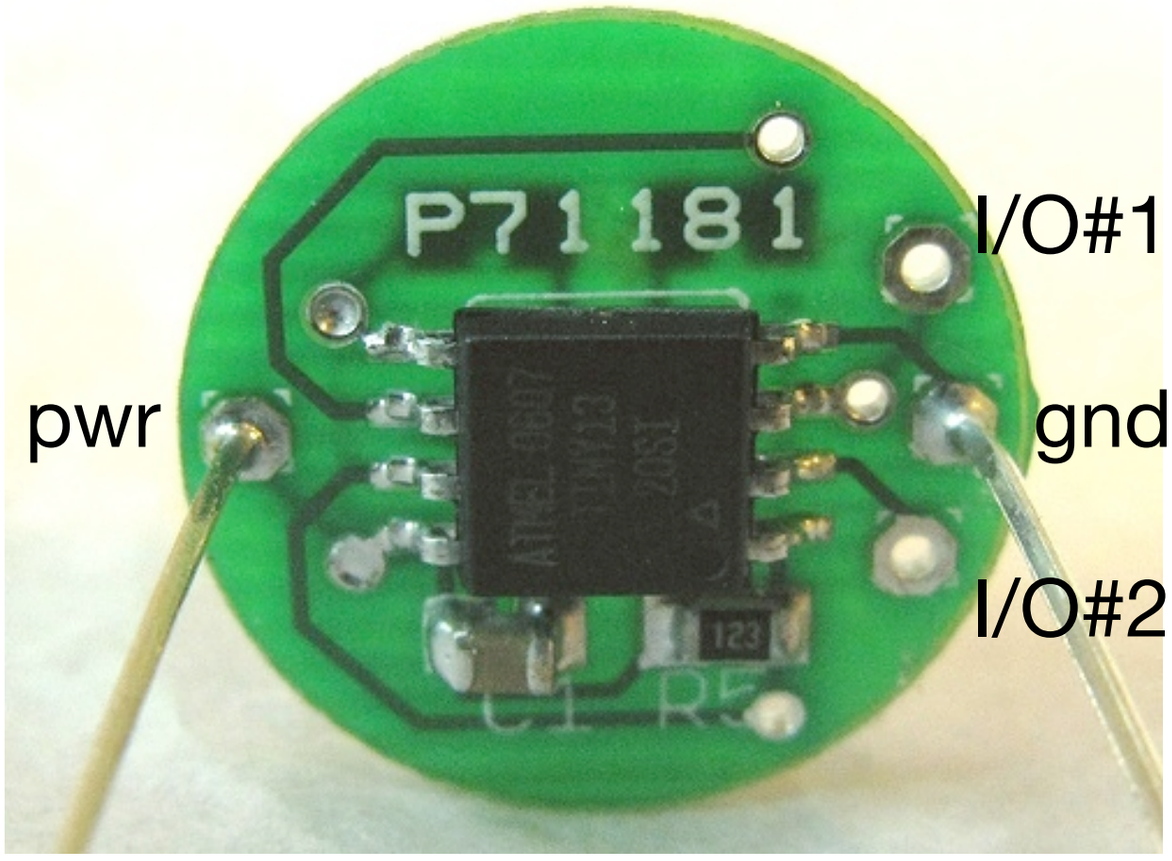



**a little bit, using standard SMD parts**

# Smart LED Prototypes

***all SMD parts, horizontal orientation***

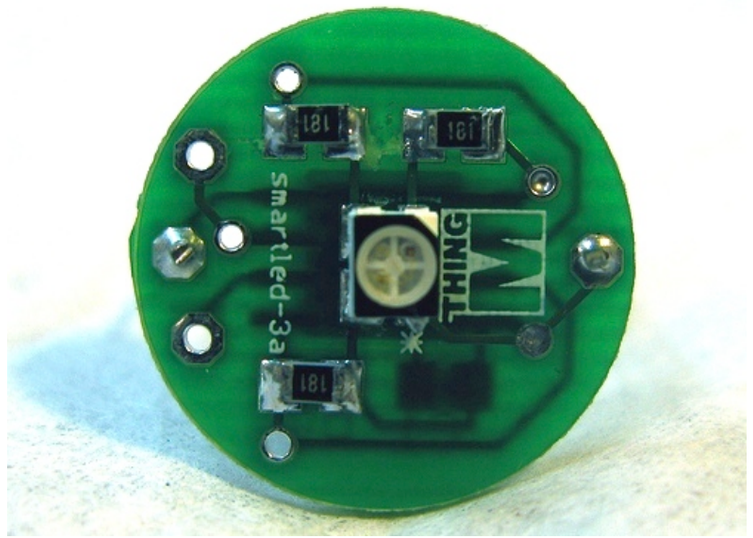


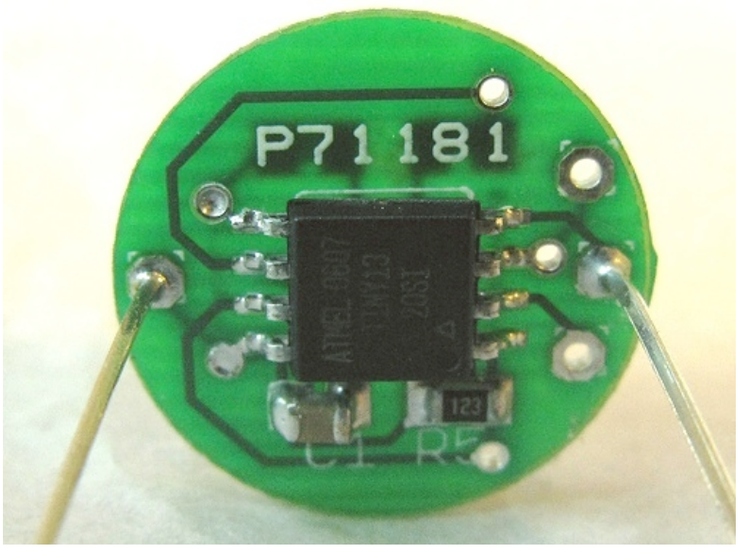




**0.6” diameter**

# Parts Cost @ 1k units

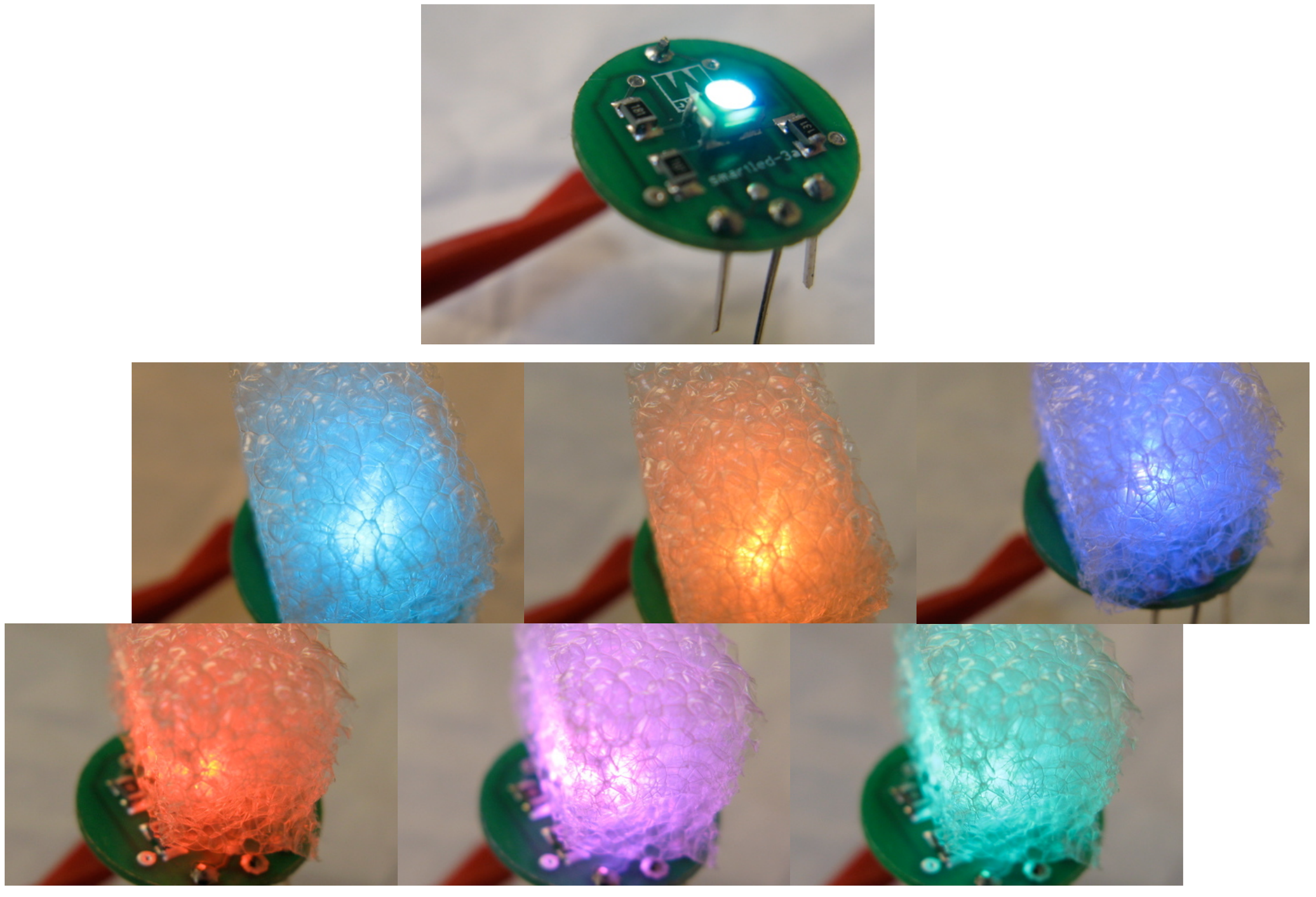
| | |
|---|---|
| **SMD RGB LED** | **$0.31** |
| **ATtiny13 µC** | **$0.88** |
| **passives** | **$0.15** |
| **board** | **$0.45** |
| **TOTAL** | **$1.79** |




*(prices from ebay,digikey,4pcb,goldphoenix)*

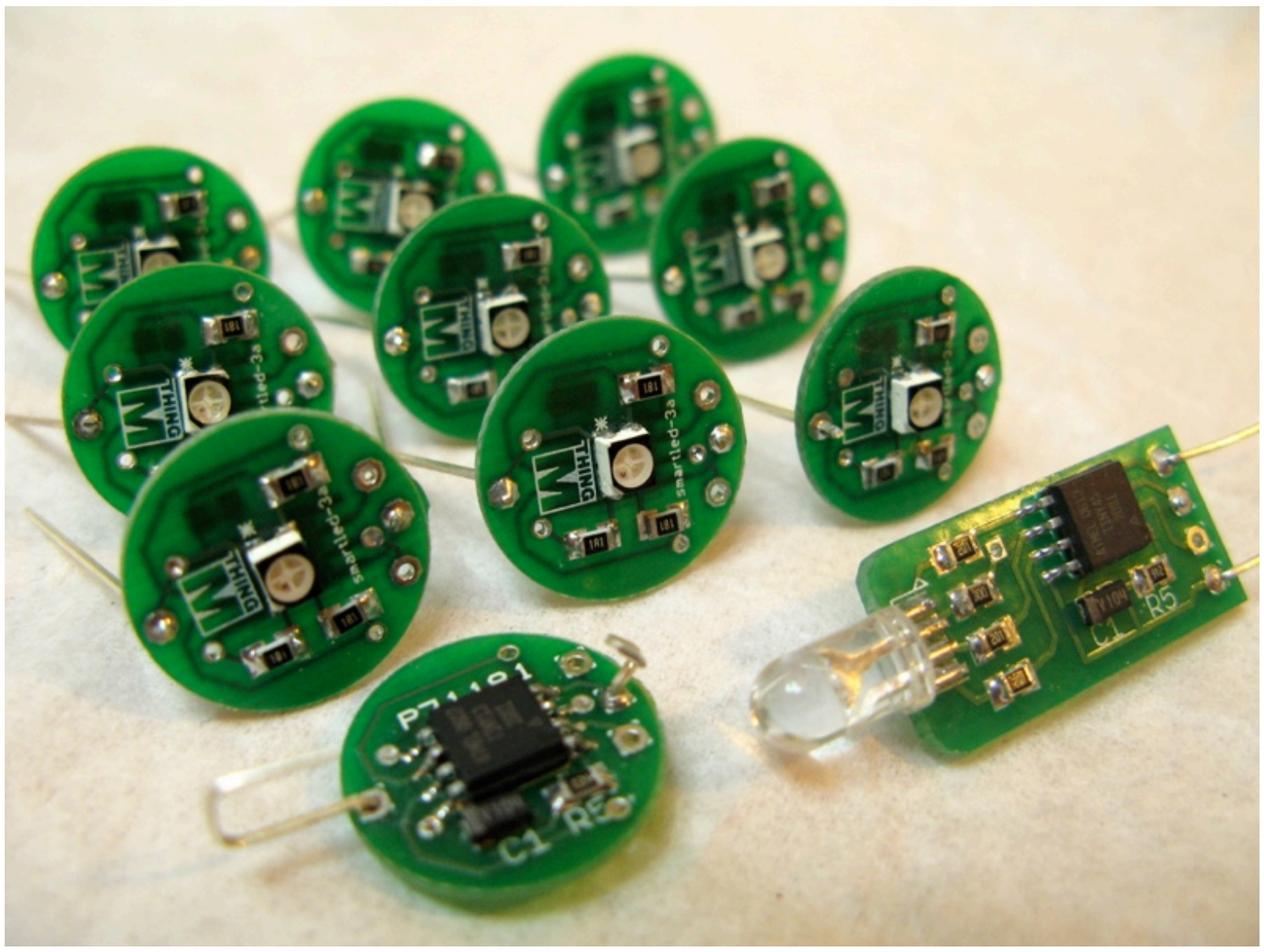
# Pretty colors



okay so I'm an LED nut.
A little diffuser goes a long way.  (in this case, packing foam)

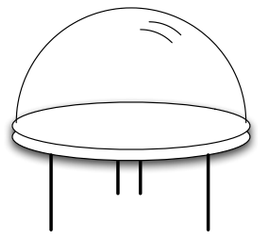
# Smart LED Army

**Functionality:**
- on 1st start: color cycle
- inputs are buttons
- input 1: select hue
- input 2: select brightness
- hue/brightness in EEPROM
- on next power up,
  use hue/bright from saved



# Next steps in Smart LED Research

- **Encase it in epoxy lens?**



- **Use smaller SMD parts & tighter layout rules**
  **— not a lot, maybe 20% smaller**

- **Work with unpackaged chip and LED dies, bonded directly to a substrate**
  **— getting about the right size, but difficult to fabricate**

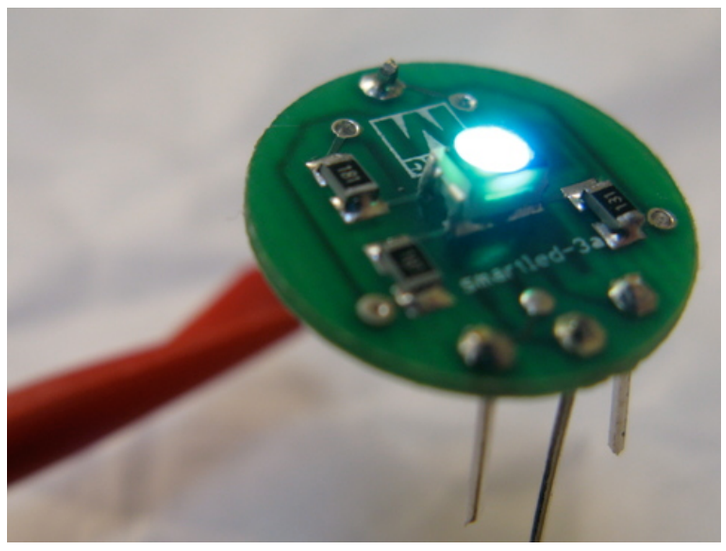
- **For fun: use LEDs as both sensor and actuator**

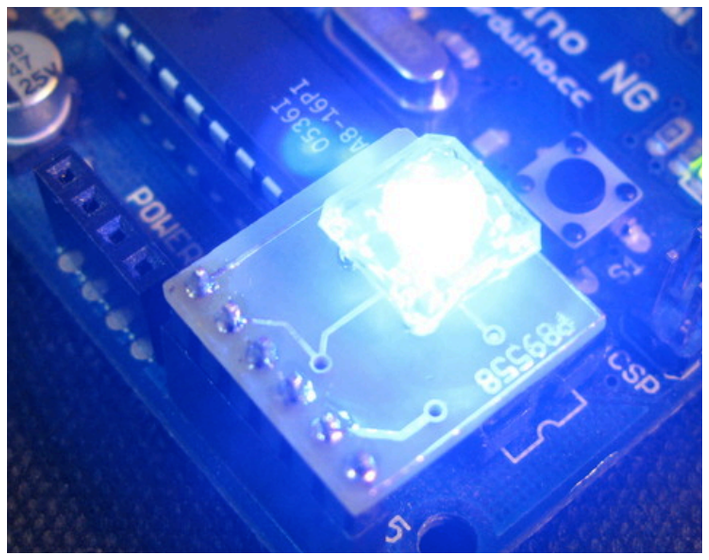
- **Communication: Multi-drop net for controlling clusters**

To truly have a drop-in replacement for a regular LED, it would need to have a similar form factor, like a bare die microcontroller in the same epoxy package as the LED dies.

See Jeff Han's work on using the LED as a sensor.

# To a real product: BlinkM Component





- **BlinkM is a productized Smart LED**
- **Multiple variations of the product:**
  - **First, a smart interface component, a hacker toy**
  - **Then, a BlinkM consumer product w/ expected end-user friendliness**

Adding the µC to the epoxying process of an LED die seems a little tough, from my research. So the first version will still be based on a PCB.

Several issues with the previous Smart LED protoypes that weren't obvious until it was in use:
- SMD LED, while bright, has a very narrow beam; not good if you want a general illuminator
- Pinout was non-standard & hard to use, but it made smallest footprint
- Controlling the several LEDs from a microcontroller was a common feature request.
- ATtiny13 µC wasn't up to task of a serial I/O protocol

# BlinkM Component Design Criteria

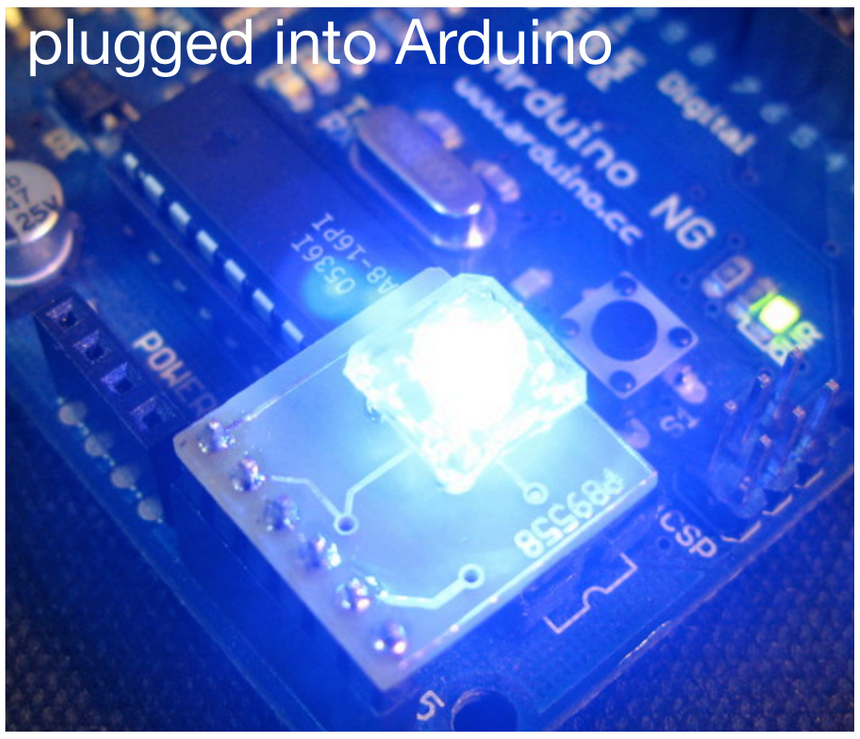
| Design constraint | Result |
|---|---|
| As small as possible | ~0.6” square |
| As bright & wide-angle LED as possible | 8000 mcd RGB LED |
| Multiple BlinkMs on minimal µC pins | I2C, ATtiny45 |
| High degree of non-trivial functionality | ATtiny45 |
| Friendly with solderless breadboards | 0.1” pin spacing |
| Easily usable with Arduino, zero-wiring if possible | 0.1” spacing & signal ordering |

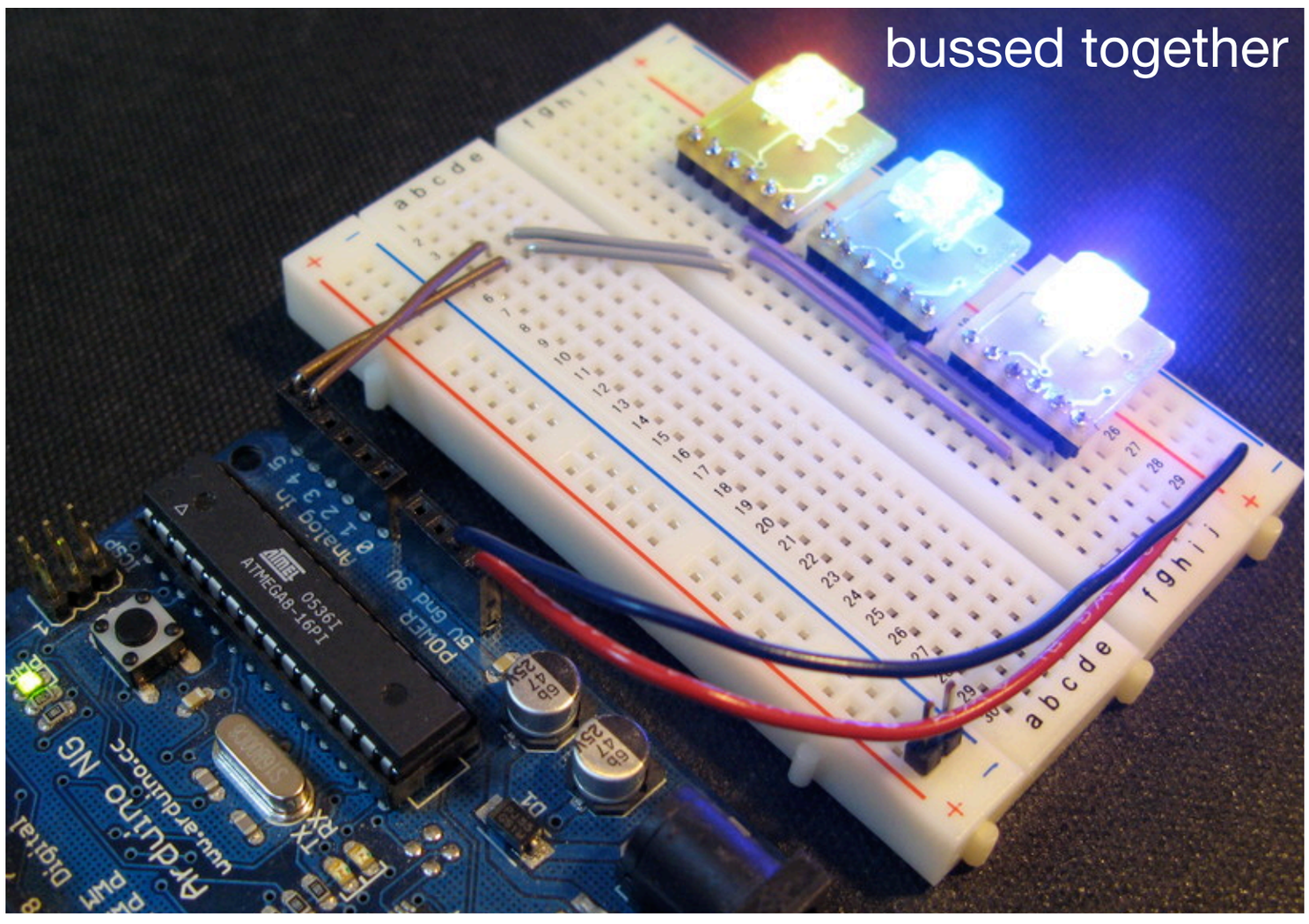
*constant interplay*

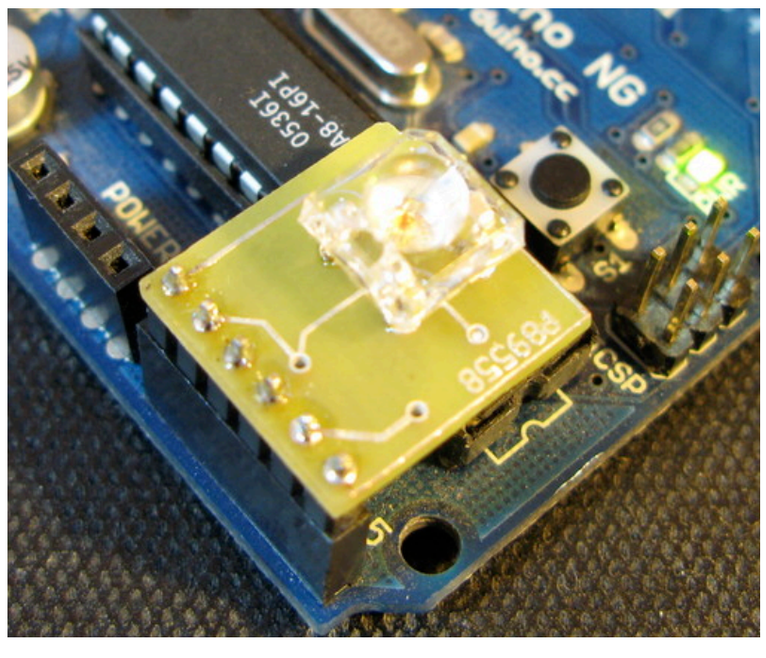
These are the main design criteria I used when designing the BlinkM component. There’s a constant interplay among the constraints. Making it hacker friendly means no tiny high-density connectors, making it small means not being able to use a µC with more pins for more I/O options, etc.

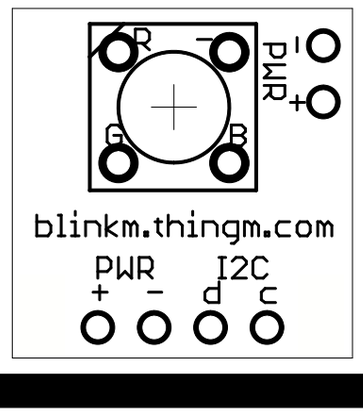
I don’t know if you’ve used I2C, but it’s a simple networked serial bus using two lines, a clock and a data, Up to 127 devices can be on the bus, each device has its own address.

# BlinkM prototype






pinout

Here's the current BlinkM thinkin'.
4-pins. Two for power, two for I2C data.
Can plug directly into an Arduino, two of the Arduino pins become a power supply.
The LED has a 140° wide beam, compared to 20° for most LEDs.

You'll be able to buy these from Sparkfun in a month or so.

# Reversible Hacking





**Manufacturers are becoming more open & more standardized.**

Let's use that.

On the other side of prototyping with new components is to repurpose existing hardware.

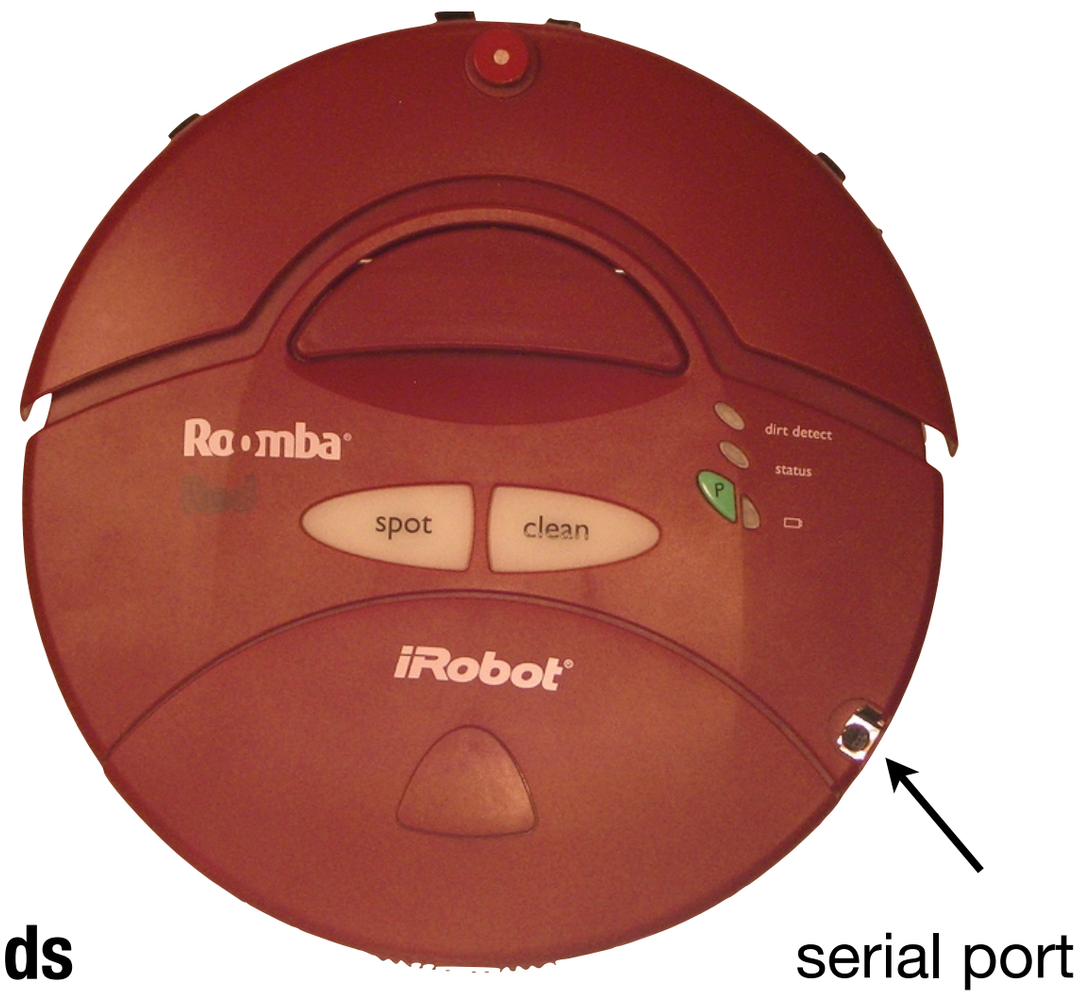
Manufacturers are increasingly using standard protocols and components. Some are embracing the hacker/open-source culture by using inventions from that world or providing hackers with tools to hack their products.

Why are they doing this? Partly it's a time-to-market issue. There's a large pool of useful code and chips out there and engineers who know how to use them. Some are realizing that their customers might come up with new & lucrative uses for their products and so are leaving the backdoors unlocked, and even publicized.

There are lots of neat technology in consumer electronics, already figured out in the most cost-effective way. Many devices can be hacked for your own purposes and then reverted back to their original state. I call this "reversible hacking". These are some of my favorites.

# iRobot Roomba

- **Serial port for commanding**
- **2 high-torque motors w/ gearing, built-in odometers accurate to 2mm**
- **6 IR detectors for bump/floor detection**
- **IR command receiver**
- **3 buttons on top**
- **Internal logic w/ high-level commands**
- **$150**



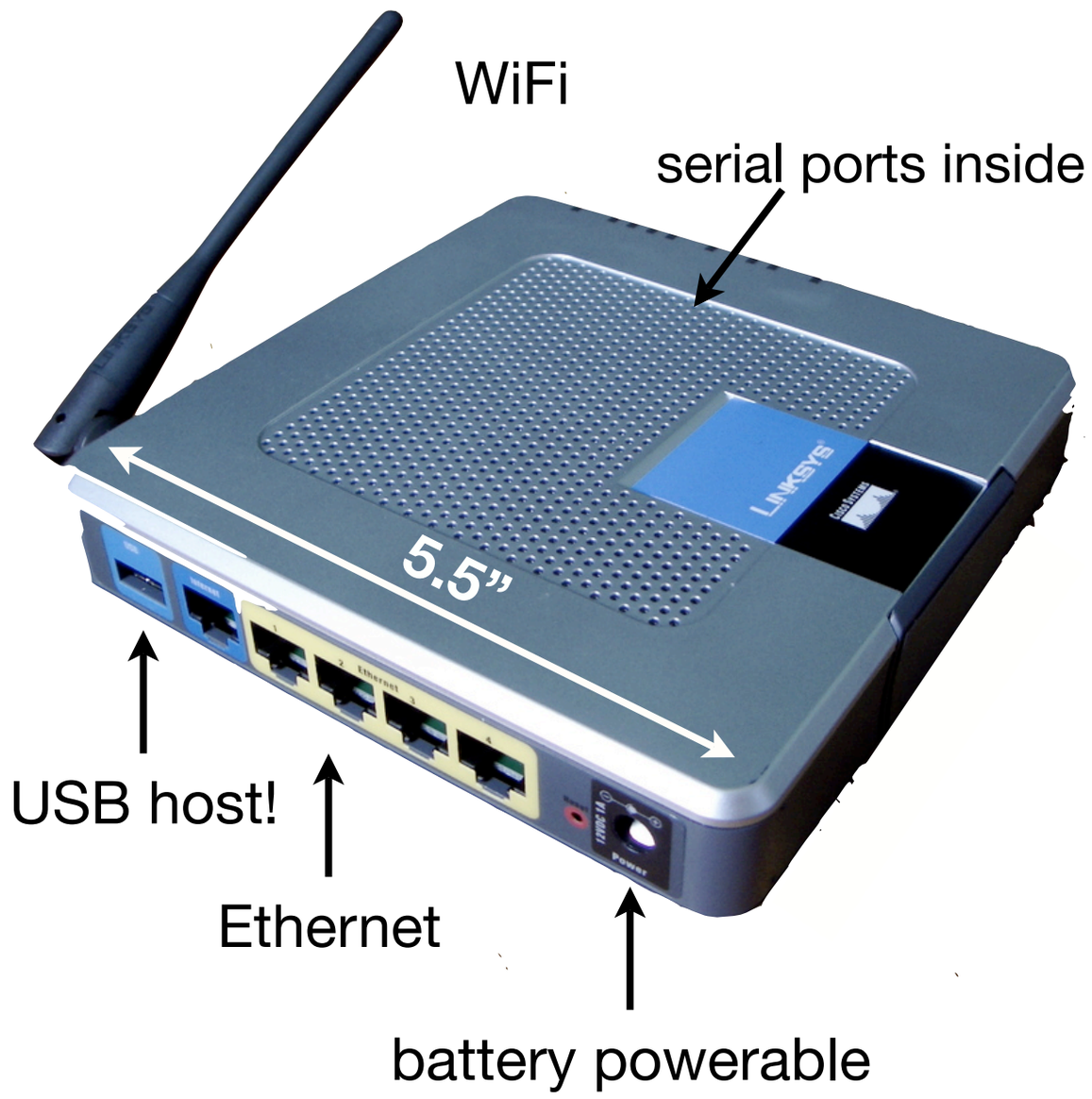

The Roomba is an amazing piece of hardware.
It's a true robot, with gobs of interesting I/O devices all wrapped up in a robust container.

Can often be found for $70 online.
None of the hacks in my book void the warranty because iRobot said it's okay to hack the Roomba's serial port.

iRobot released the iRobot Create robot in response to hacker interest.

# Linksys WRTSL54GS

- **Many Linux distros for it**
- **266 MHz processor**
- **32 MB RAM / 8 MB Flash**
- **2 built-in TTL serial ports**
- **USB 2.0 host**
- **Ethernet & WiFi**
- **< $100**




There are tons of wireless routers that run can Linux, the Linksys WRT54G was the first. This router is the “storage” version of it, and has a USB port.
It runs its own version of Linux from the factory.

These devices are really great bases for more advanced hacking. This one’s not much bigger than a paperback book and can be run off battery pack.

Linux means a large library of existing code to play with, and writing more is easy.
The presence of WiFi & Ethernet mean connectivity to the world is easy.
A serial interface admits connections to hacker gadgets like GPS, Arduino, etc.
The USB host interface allows connectivity to huge array of peripherals: webcams, audio in/out, mice, joysticks, etc.

# Nintendo Wii Nunchuck

- Standard I2C interface
- 3-axis accelerometer w/ 10-bit accuracy
- 2-axis analog joystick with 8-bit A/D converter
- 2 buttons
- $20

don't forget the Wiimote!

If you look at the architecture for the Nintendo Wii and its peripherals, you see an almost un-Nintendo adherence to standards. The Wii controllers are the most obvioius examples of this. The Wii controller bus is standard I2C. The Wii remote speaks Bluetooth HID to the Wii (or your Mac or PC)

Because it uses standard I2C, it's easy to make the Nunchuck work with Arduino, Basic Stamp or most other microcontrollers.

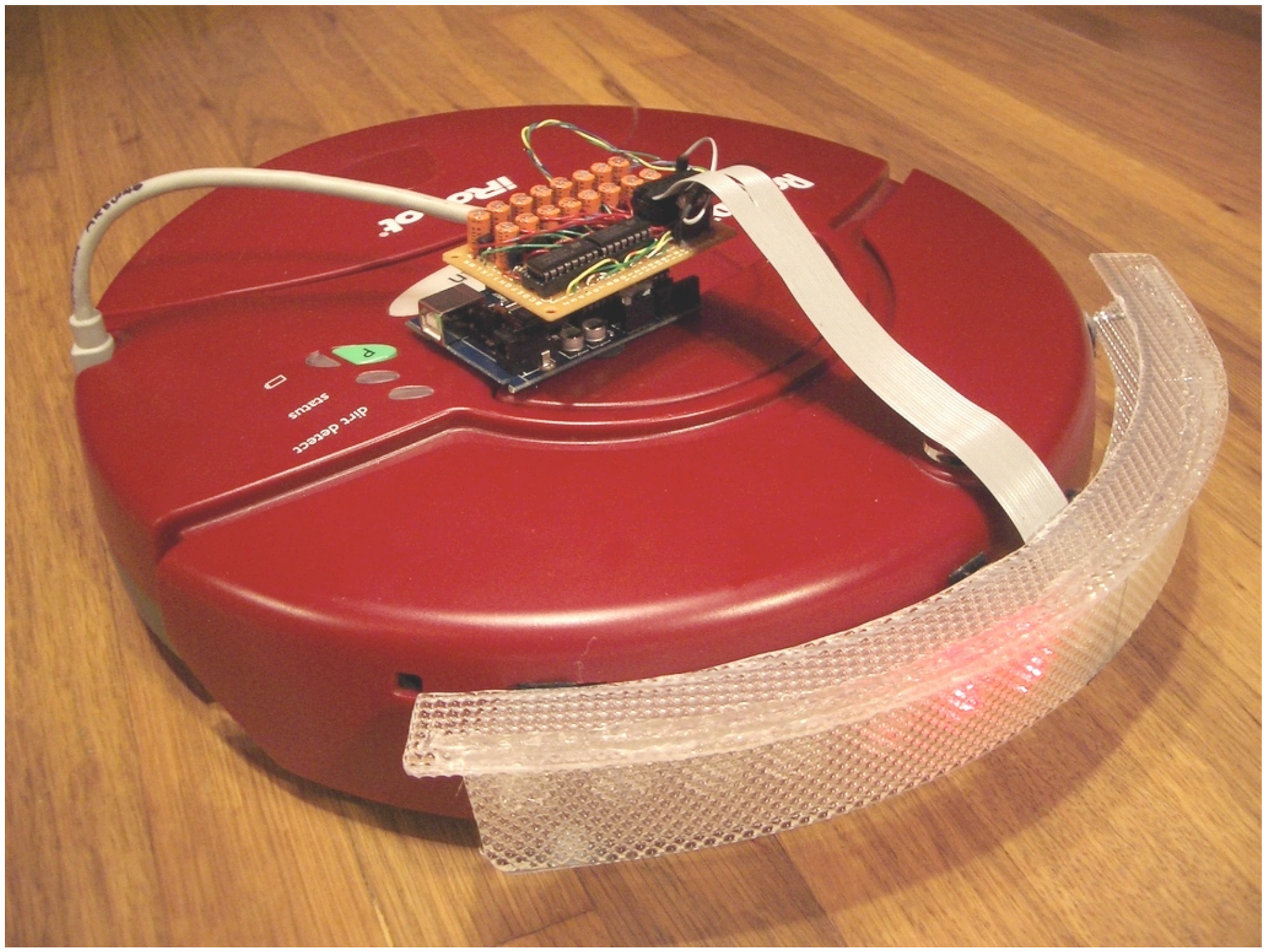
See: http://www.wiili.org/index.php/Wiimote/Extension_Controllers/Nunchuk
and: http://www.windmeadow.com/node/42
and: http://todbot.com/blog/2007/10/25/boarduino-wii-nunchuck-servo/

And then there's the Wii Remote, besides Bluetooth HID, it also has accelerometers, buttons, speaker, memory, and is I2C master.

# Combine them like Lego

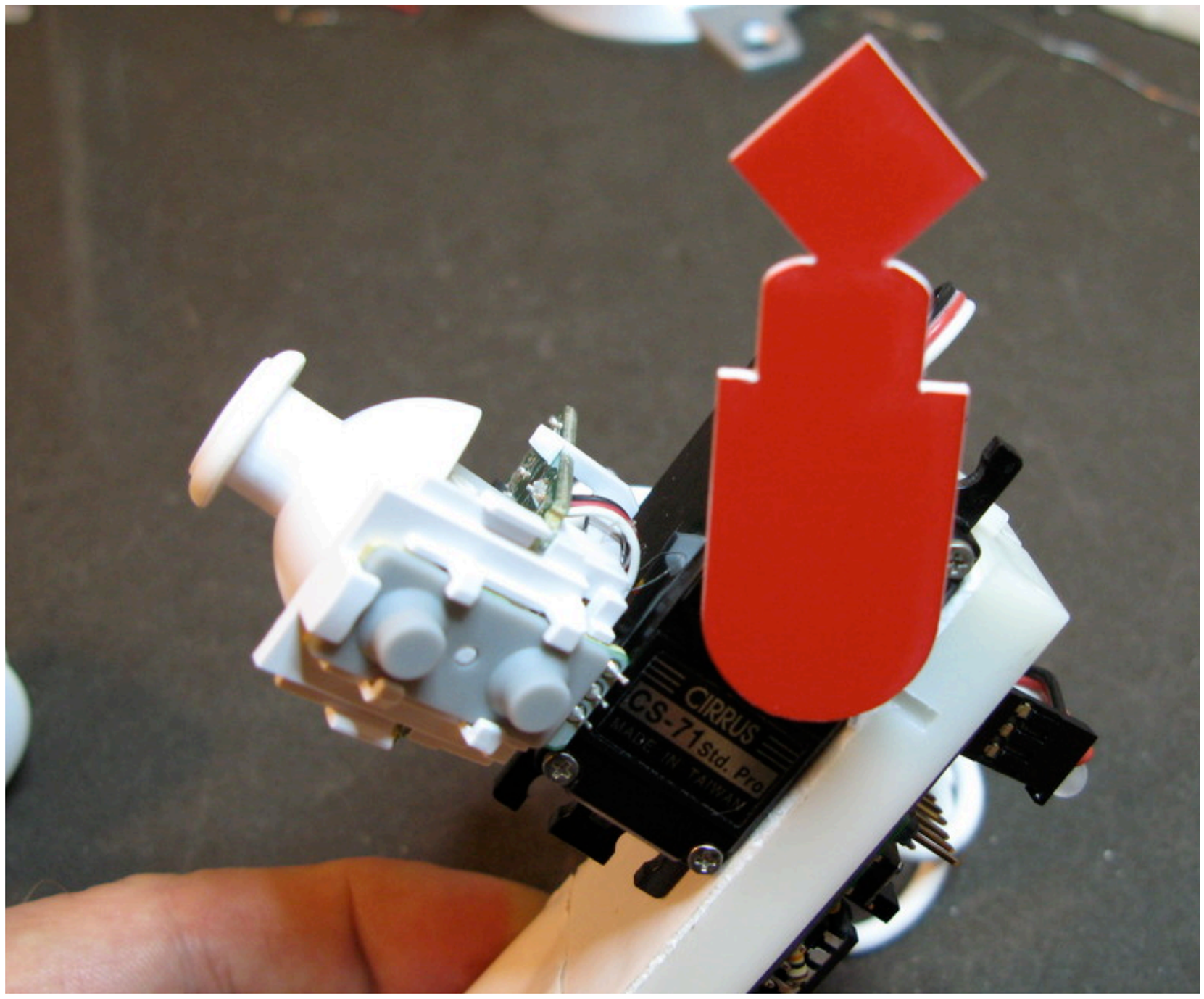

Linksys WiFi router running Linux + webcam + 1 GB thumbdrive + Roomba = telepresence rig and home defense system.

# Combine them like Lego




Arduino + Roomba + LED driver = Cylon Roomba

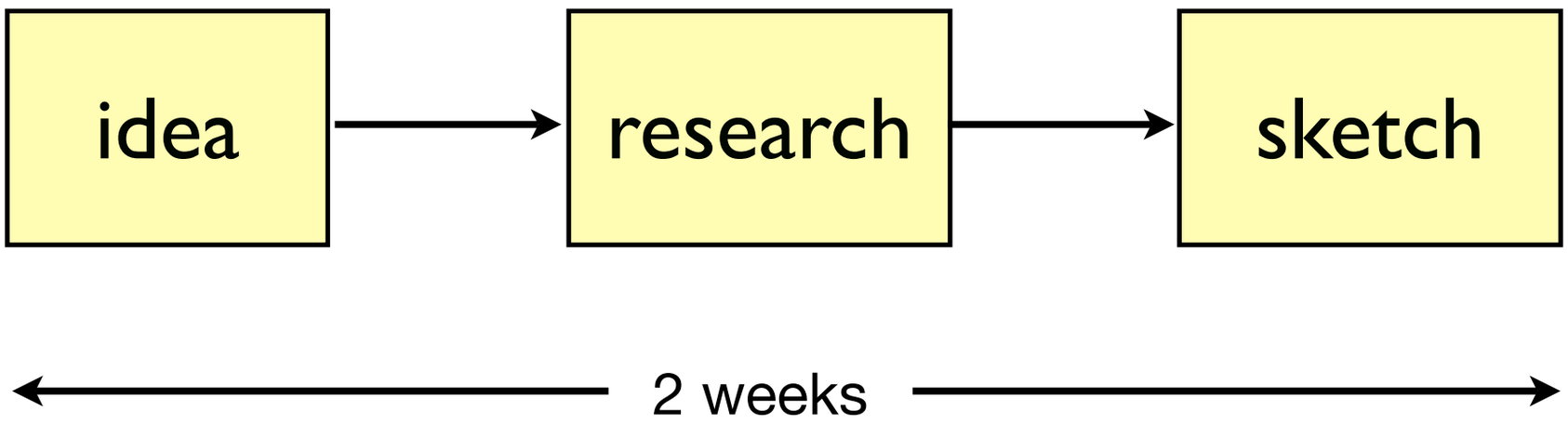
# Combine them like Lego




Wii Nunchuck + servo + Arduino = Segway balancing emulator

# Technology Sketches

**Videos of ideas as if they already existed**




Semi-functional prototypes created in a few weeks

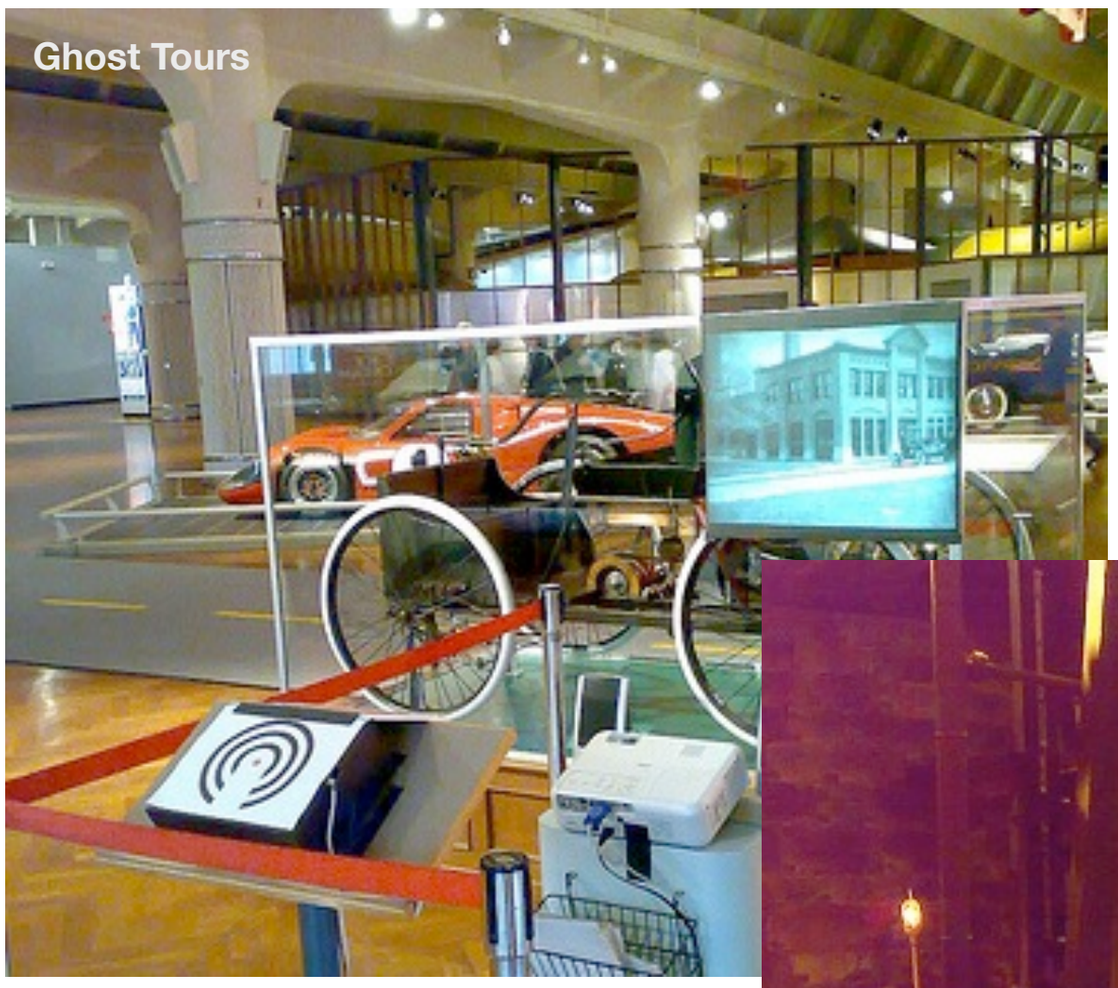
So why have I spent so much time thinking about fast prototyping tools?

We work via fast iterations of sketches and prototypes, exploring a possibility space as quickly as possible. Arduino makes this possible. As do smart interface components and consumer products that can be reversibly hacked.

When I say “sketch” I’m referring to a very particular method we use, we call it a “technology sketch”. It’s a short video of a semi-functional prototype showing an idea in everyday use. Each sketch is completed in less than a week, after about a week of research. If a sketch’s concept proves promising, we iterate towards a more real prototype, and from there perhaps to a real product.

The production style of the videos is purposefully without narration, and only a few minutes long.

# Technology Sketches






For the Henry Ford Museum this summer, we created three completely different new experiences for museum goers.

**Ghost Tours** — Created story-driven audio-visual museum tours using RFID-embedded tickets. Allowed multiple views of an artifact with no museum layout changes.  Enables visitor demographic analysis by the museum.
**Knowledge Periscope** — An update to those coin-operated binoculars, this augmented-reality device let you see inside museum artifacts, see them in their original location, or see them in action.
**Historiimote** —  Wave a magic wand at objects and get either expanded descriptions or engaging stories. The technology was based on low-cost Wii Remotes.

# LoveM

- “Memory chocolate box”
- Heart-shaped LCD
- Each chocolate triggers a memory
- Memories are photos or videos
- LEDs light up box

Essentially a battery-powered LCD photo frame, but in the guise of a chocolate box. The giver chooses “memories” to add to the chocolate box. When the receiver removes a chocolate, one of the memories play.

# LoveM Technology Sketch


It's hard to make this idea come alive with just feature points, so this was the technology sketch video demonstrating the idea.  It was created in a few days, with friends of ours as actors.

http://thingm.com/sketches/lovem

# WineM

- “Smart wine rack”
- Wine information display & browser
- Uses the wine itself as a display
- LEDs & RFID reader per bottle
- Touch pad selects data slice


Here’s another idea we did a sketch for.
Searching through wine is difficult when you have a large collection. WineM is an attempt to make that task easier by displaying information about wine on the wine itself. The wine rack becomes the information display.

A handheld touchscreen lets you navigate the data space via a facet map browser. RFIDs on each bottle uniquely identify it, and RGB lights in the rack behind each bottle shine information directly on the wine.

# WineM Technology Sketch

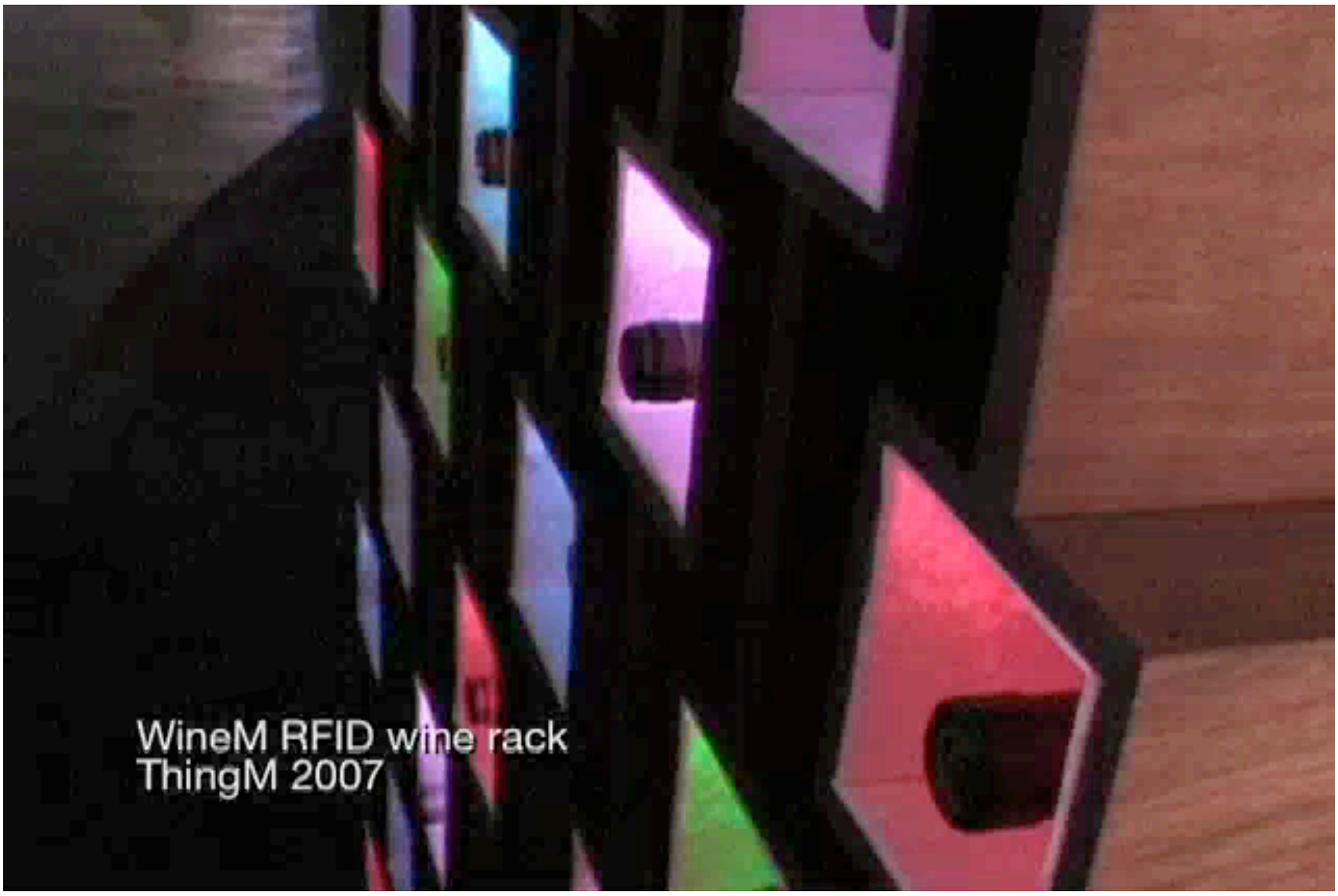

This was the resulting technology sketch video.

We posted it to the Net and went on with other things. We were amazed with the response we got from it. Wine collectors kept contacting us wondering how they could buy it. Then Wired invited us to show off WineM at their NextFest conference. Since WineM at that point existed mostly via the magic of video editing, this meant we had to build it for real.

We weren't even sure if it was technically feasible to have so many RFID readers and tags in such close proximity. While I focussed on these technical challenges, Mike and our industrial designer intern went about creating a design that communicated this wasn't any ordinary winerack. After about 60 iterations, they chose design #38.

http://thingm.com/sketches/winem

# WineM Prototype in Action



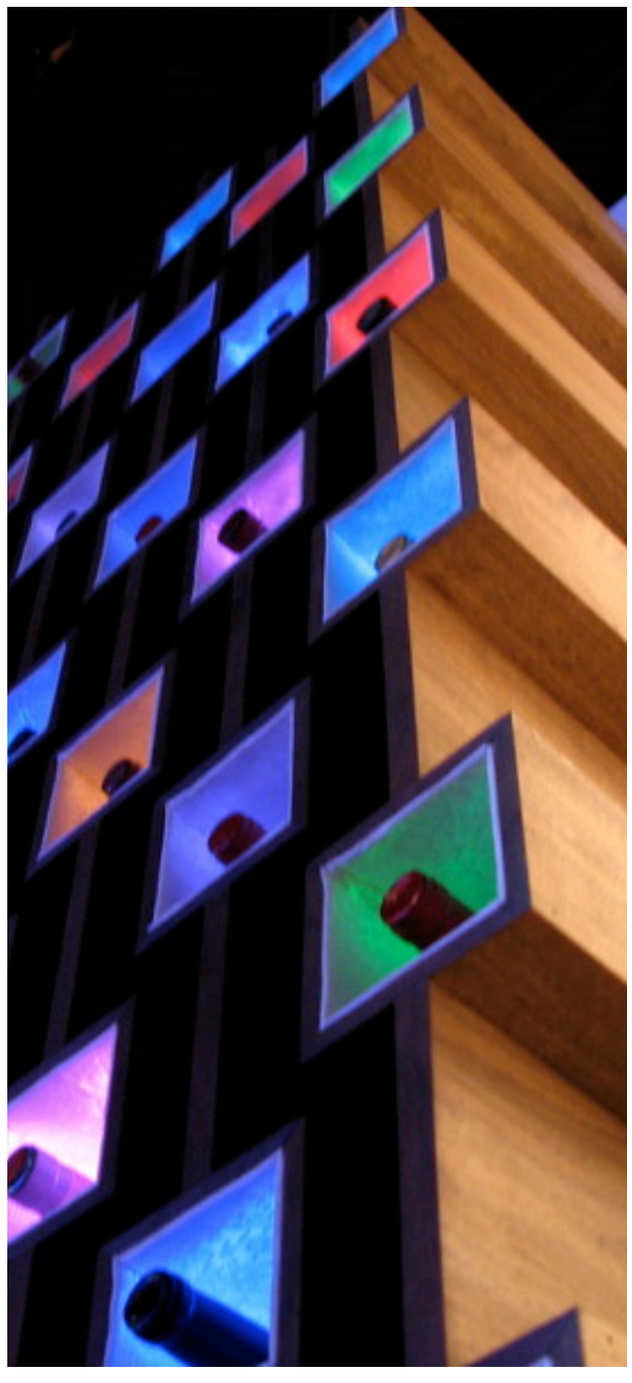

Here's a video of it in use at NextFest.  In this video, the user is browsing their collection.
"Show wines by Grape, choose Cabernet, show by Year, choose 2003, show by price."
It knows when (and which) bottles are added and removed.

We're currently designing updated prototypes.

---

"WineM is a ubiquitous computing tool for managing your wine. It keeps track of what wine you have and where it is in the rack, dynamically. It makes all of the information that's available about your wine collection available to you, displayed on the bottles. For example, you can light up all california wines in different color by varietal, then narrow that down to only the ones that are drinkable right now, then select the ones currently worth less than $50, have a high level of procyanidin, and are different than the wines your friends have, so you know who to invite over." --mikek

# Informational Objects



# Informational Shadows of Physical Objects

*by raindog on flickr*

| | |
|---|---|
| Tracking Number: | 1Z 5R8 939 03 0059 515 9 |
| Type: | Package |
| Status: | **In Transit - On Time** |
| Scheduled Delivery: | 11/07/2007 |
| Shipped To: | PASADENA, CA, US |
| Shipped/Billed On: | 11/05/2007 |
| Service: | GROUND |
| Weight: | 2.00 Lbs |

**Package Progress**

| Location | Date | Local Time | Description |
|---|---|---|---|
| BALDWIN PARK, CA, US | 11/06/2007 | 10:00 A.M. | IN TRANSIT TO |
| ANAHEIM, CA, US | 11/06/2007 | 9:59 A.M. | ARRIVAL SCAN |
| PHOENIX, AZ, US | 11/06/2007 | 2:05 A.M. | DEPARTURE SCAN |
| PHOENIX, AZ, US | 11/05/2007 | 8:56 P.M. | ORIGIN SCAN |
| US | 11/05/2007 | 8:28 P.M. | BILLING INFORMATION RECEIVED |

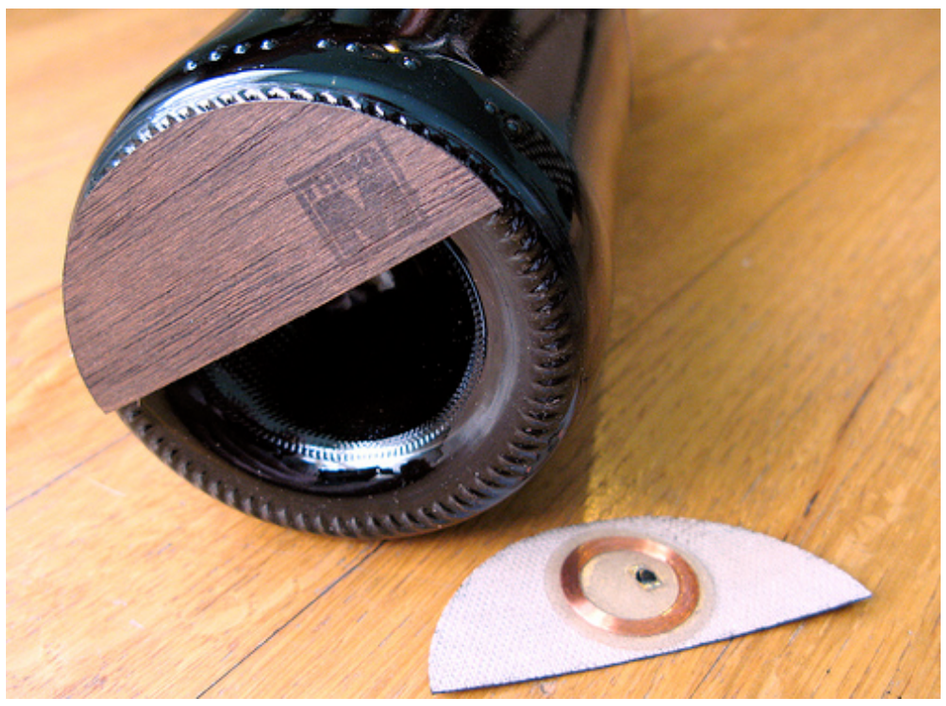
Tracking results provided by UPS: 11/06/2007 1:55 P.M. ET

Our packages are already blogging to us

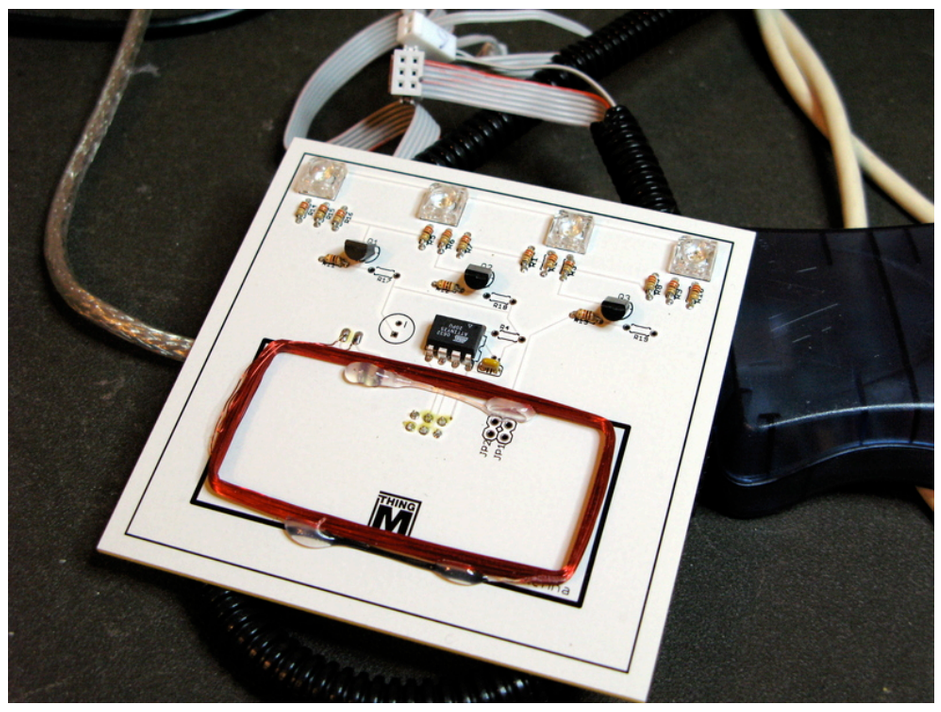
Physical objects are casting shadows in information space.
Only a few objects have their two halves connected, packages are one.

c.f. “blogjects” – http://www.nearfuturelaboratory.com/files/WhyThingsMatter.pdf
photo by raindog on flickr

# Need Real ↔ Info Gateways



labels



label readers & annunciators

For right now, these labels enable a transitory form of smart objects

Packages are one of the few objects that have this binding.
Until all objects become smart and networked, there's a need to develop gateways between the physical and informational. These are labels or stickers or codes we apply to physical objects that are pointers into information space. It turns these objects almost-smart. RFID tags have embedded logic but are generally passive (must be powered by the RFID reader) and act only as information holders.

RFID tags are a decent bridge between the physical and informational, better than UPC barcodes because they offer the chance for unique identification ("my bottle of two-buck chuck vs. all bottles of two-buck chuck")

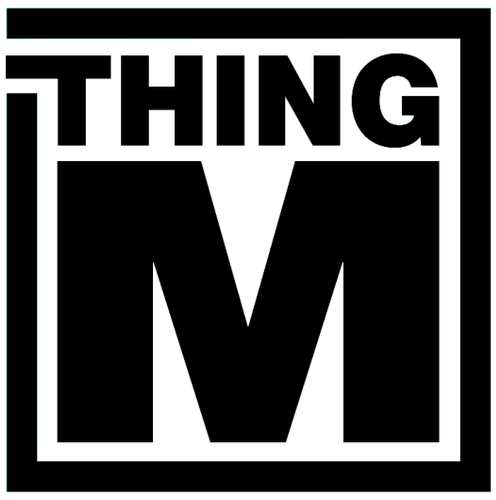
Shown here is the walnut veneer RFID tags for wine bottles the per-bottle reader board we developed for WineM.

The per-bottle reader board is another type of smart interface component. It is a stand-alone combination of Smart LED + RFID reader. It contains enough intelligence to live on an I2C network with its peers, telling the "coordinator" when a bottle appears or leaves and can be told to illuminate the bottle.

# Wine as Informational Object


One of the main reasons we did WineM is that wine exists in both information space and physical space. It has a rich information presence (varietal, year, region, reviews, ratings, etc.) in addition to its physical presence (the bottle). But the two are not connected, and bottles are hard to identify: no ISBN for wine, barcodes are unreliable. WineM brings the two aspects of wine together.

Bits are so much more malleable than atoms and WineM is an attempt to treat your wine atoms more like bits. The same techniques can be used for other items have a lot of information about them and are hard to organize. Books are an example, as are shoes for some people.




**Tod E. Kurt**

tod@thingm.com

PDF of this talk available at:

http://todbot.com/blog/

Thanks!

Feel free to email me if you'd like to chat about any of these things I've talked about.