

Tom Igoe
ITP,
Tisch School of the Arts,
NYU

Thank you for inviting me, it's great to be here.




I teach at the Interactive Telecommunications Program in the Tisch School of the Arts at New York University. This is the view from my office. ITP is a two year graduate program...



ITP was started in 1979, by this woman (click), Red Burns. She was inspired to start the place when she saw this camera (click), the Porta-Pak. It was a relatively inexpensive, relatively portable video camera. Red saw the Porta-Pak as the beginning of a change in people's relationship to media. She believed that they could go from passively consuming media to actively participating in their production, so she started the Alternate Media Center at NYU, which later became ITP.



Her original vision is still our driving principle: when people have access to technologies, and the knowledge of how to use them, they're move from a passive role to an active one. It's a pattern we've seen repeated through many different waves of media & technology.



Who we are:

- * ~110 students per class (220 total)
- * 12-15 full-time faculty
- * 30 - 40 part-time (adjunct) faculty
- * 8 staff
- * 10 resident researchers
- * 60% female, 40% male
- * 60% U.S., 40% international
- * approx. 50 classes per semester
- * approx. 25% new classes every semester

Here's who we are by the numbers. We take people from a range of disciplines: arts, psychology, anthropology, marketing, sciences, design, architecture, engineering, and so on. The ideal student is in her late 20's, a few years into her professional career. She's someone who's had enough work experience to see how things work, and yet is still fresh enough to think things can change for the better.



Collaboration requires us to develop a shared language

Kate Hartman
Talking To Yourself Hat

Because students come from many different backgrounds, they have no common assumptions or training. They all look at problems differently. They have to learn to talk to each other, across their disciplines, and to develop a shared vocabulary. It takes a lot of patience and forgiveness. They have to re-learn to ask naive questions without fear of judgement.



Ayad Alkhadi, Ed Guttman
Flight Simulator

A large part of mastering
any new technology is
letting go of fear.

We teach students programming, electronics, and media production in their first semester. There's a lot to learn in a short time and we need them to get up to speed fast. Play is important to that. People let down their defenses when they play, and they discover things that they might not when working in a more directed way.



Intellectual promiscuity
fosters good ideas

We encourage students to be generous in sharing ideas with one another. Ideas are cheap, and they get better when you try them out on others and iterate on them a bit. Deciding which ones you want to act on, and then executing them, takes work.

In addition, students own their own IP at ITP, they generally not work on sponsored research.



I manage the physical computing curriculum at ITP. Our aim in those classes is to teach people to expand the range of human physical expression to which computers can respond. We start by asking “What does the user do, physically?”

How We See the Computer



We started teaching this in about 1992, and the original and perhaps still the best explanation came from Joy Mountford, who was at Apple at the time, she said, “it’s not...”

How We See the Computer



How The Computer sees us

The computer's perception of us is limited by which of our actions it can sense. It used to be difficult to describe this, but now I can just say...

“It’s what tools like the Kinect
make possible.”



And they get it. Or at least, enough that we can start the conversation.

Many of the great new physical interaction tools these days are coming from gaming, and many of them are intangible, gestural interfaces like the Kinect. But...



Tuan Anh T. Nguyen
Happy Feedback Machine

It's not just about the gesture, though. We also encourage students to consider tangible interfaces with rich sensory feedback, that take advantage of all of our tactile ability to learn.

Expressive

Matthew Richard
Estrella Intersects the Plane



What do we do with this? A few different things. Sometimes we make expressive works, like this mechanical color field painting. (click). Matthew Richard was a painter when he came to ITP, and he wanted to make color field paintings that move.



Expressive works can be works of art, but there are other forms of expression: political or instructive, for example. This piece is designed to teach you about circadian rhythms.

Primarily these are works that draw attention to themselves to start a conversation.



Then there are instrumental works. Whereas expressive works draw attention to themselves, tools allow us to forget them and focus on the activity they're enabling instead. If you're paying attention to a tool, it's not well made. These can be tools for expression, like this musical controller,



Or for everyday life, like this chair and head-mounted PS3 controller for a client with muscular dystrophy.

Tools for change

Jorge Just, Rune Madsen, Karla Calderon,
Mustafa Bagdatli, Dharmarajan Ayakkad, ThoughtWorks
RapidFTR

Photo: Jorge Just

Sometimes we make tools for social change, like this mobile application to help aid workers in family tracing and reunification in crisis areas. This was built in a class we run in conjunction with UNICEF.

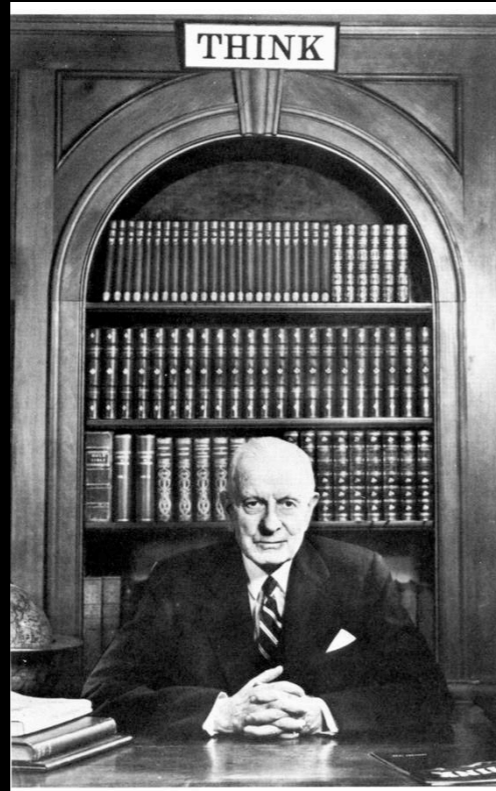


Bobby Genalo
Play Something: Data

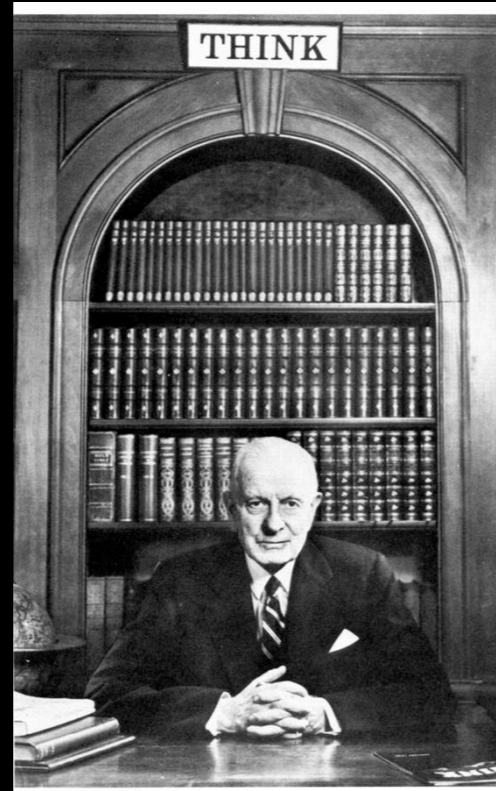
I make the distinction between expressive and instrumental contexts, because we can't count on any one context being the default in our program. A project like this, for example, will get a very different critique in a toy design class than it will in a data representation class. But it might typically be developed in either, or both.



So prior to getting involved with interactive technologies, I worked in stage lighting and projection design. That background has influenced my current work, in that performance principles apply well when designing interactive systems. It's all about what the actor does, what her motivations and intentions are. We just call the actor a user in UX design.



As an undergrad I was simultaneously an IBM Thomas Watson scholar...



...and a Lee Liberace scholar. I like to believe that somewhere in hell the two of them are playing craps for my soul.

Liberace: relentless popularizer, didn't dumb down for his audience, unapologetically sentimental, kick ass technician, dressed better than most geeks.



Before I started teaching full time, I worked on a few different physical interaction design projects. I was the technical director for Diller + Scofidio's Blur pavilion for the Swiss international expo 2002.



The idea was that visitors would wear raincoats that expressed their affinity or antipathy for each other as they passed each other in the fog.



My job was to design the technical system to make this possible.



Here's the building as built. Sadly, we never got to test the interactives, as the budget for all of them, and me, got eliminated before the build. You win some, you lose some.

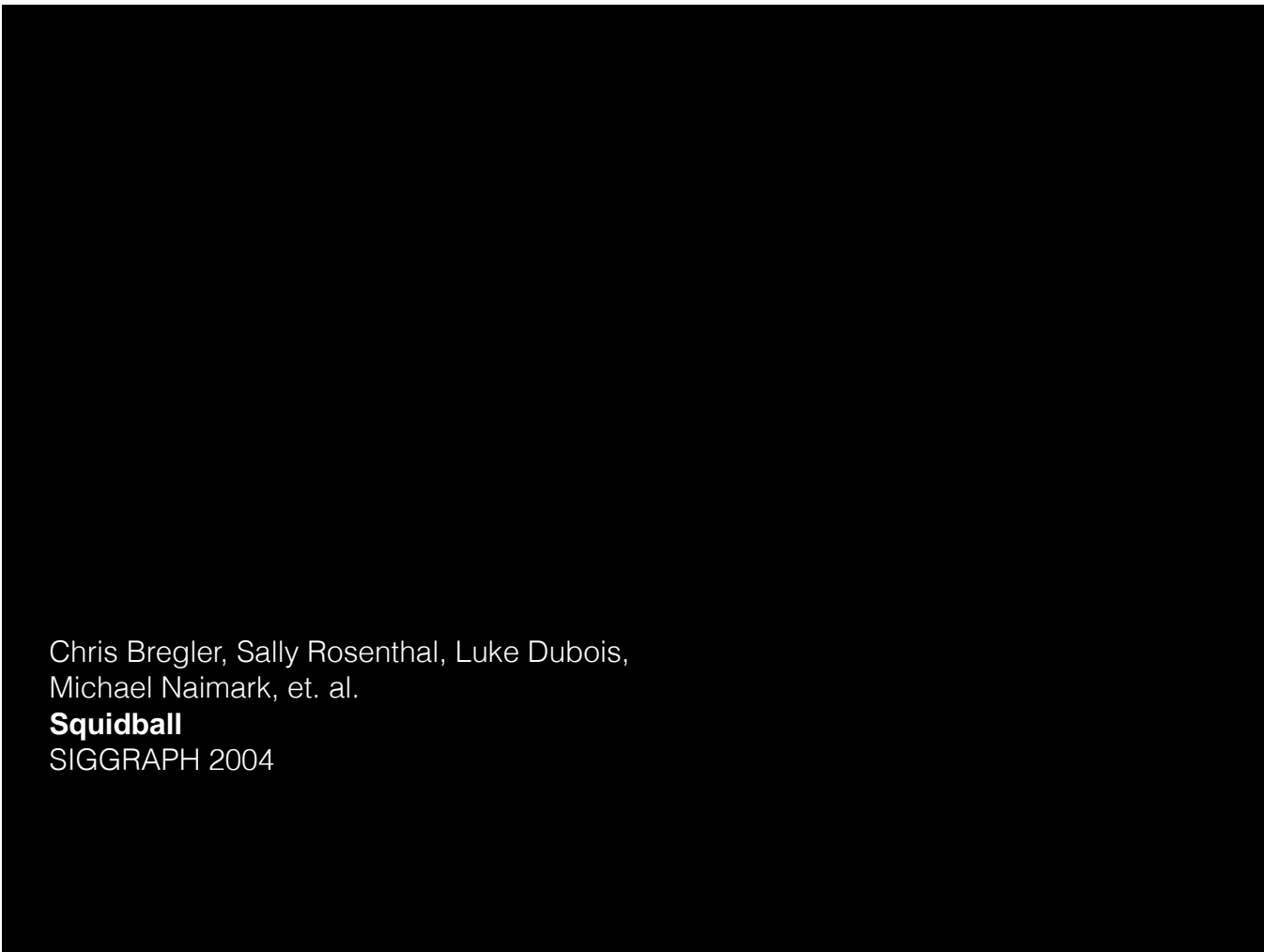


GMHC, Museum of the City of New York
Jean Carlomusto, Jane Rosett, curators
AIDS: A Living Archive

I also designed and built an interactive video memorial for the Museum of the City of New York. It was series of short video portraits of various people who'd been active in the AIDS movement at the height of the crisis in New York (most of whom are dead).



Each had a votive candle that slowly flickered when nothing was happening, and a button that triggered their video story. It was a lesson in how politics affects design for me — whose candle went where really mattered to the project stakeholders.

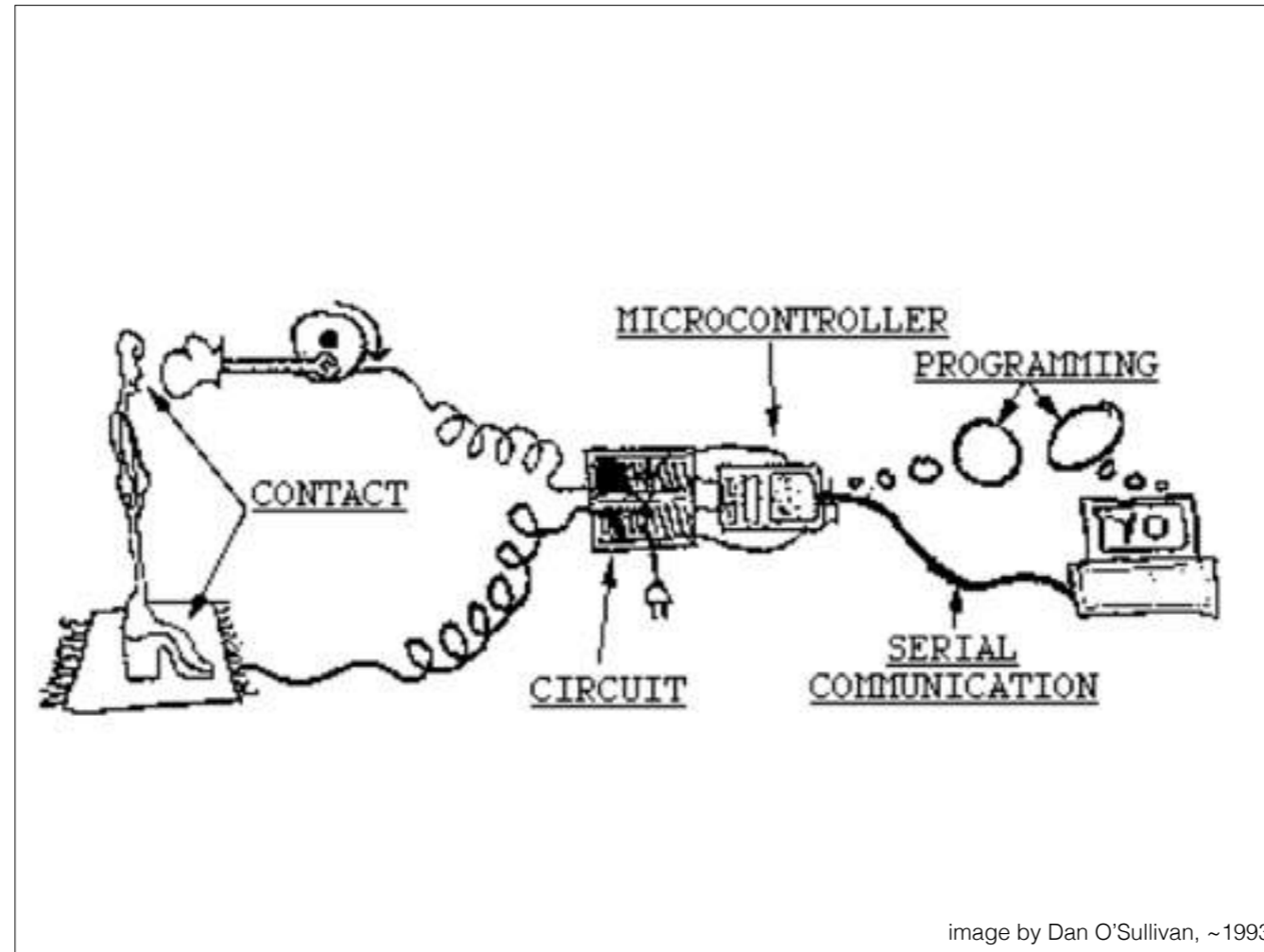


Chris Bregler, Sally Rosenthal, Luke Dubois,
Michael Naimark, et. al.

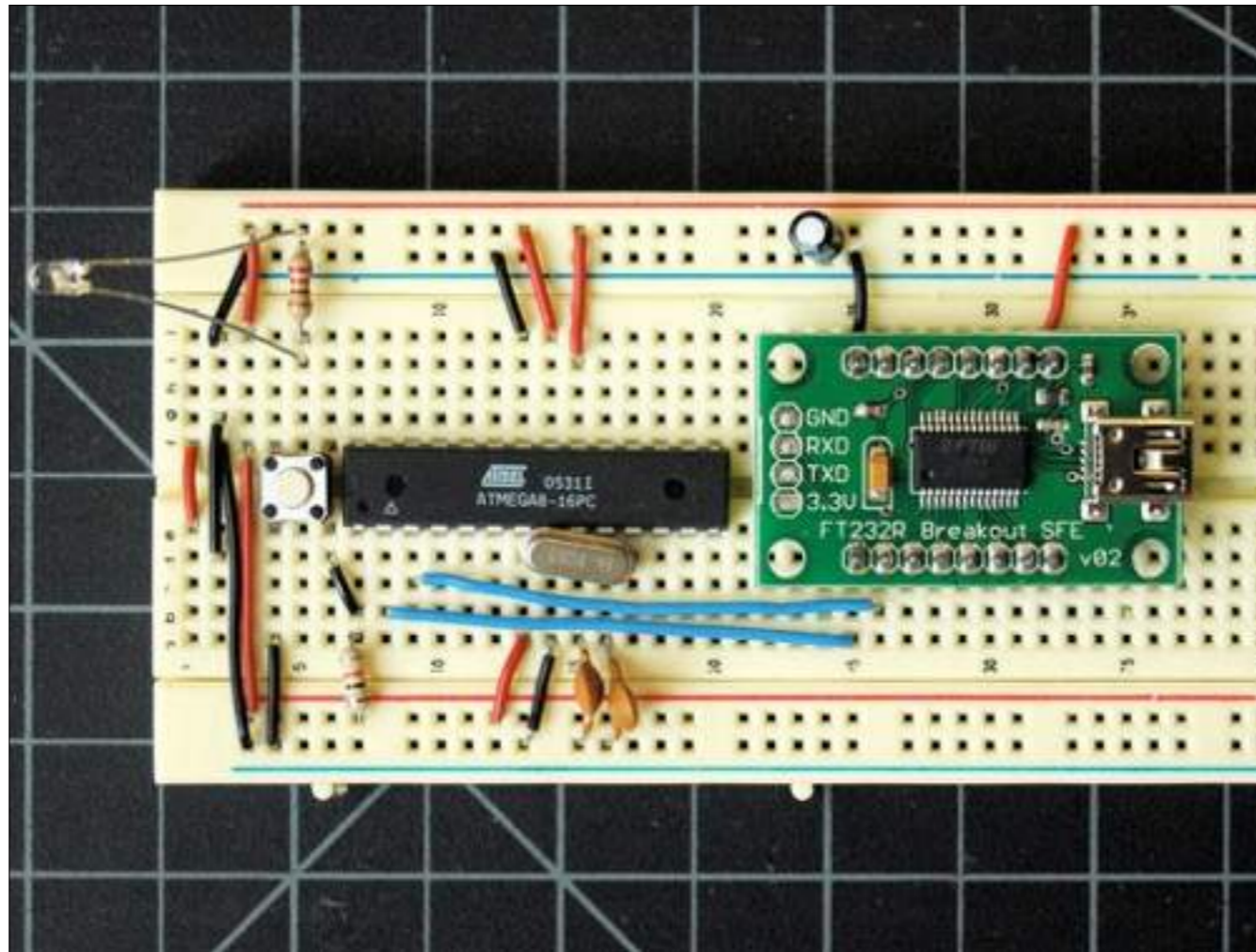
Squidball
SIGGRAPH 2004

And I worked on a project with my computer science colleague Chris Bregler to create a motion capture game for 4000 people. I think it might still hold the record for the largest motion capture rig yet.

All of these were grounded in thinking about the actor and her intentions to help define the necessary physical interfaces.



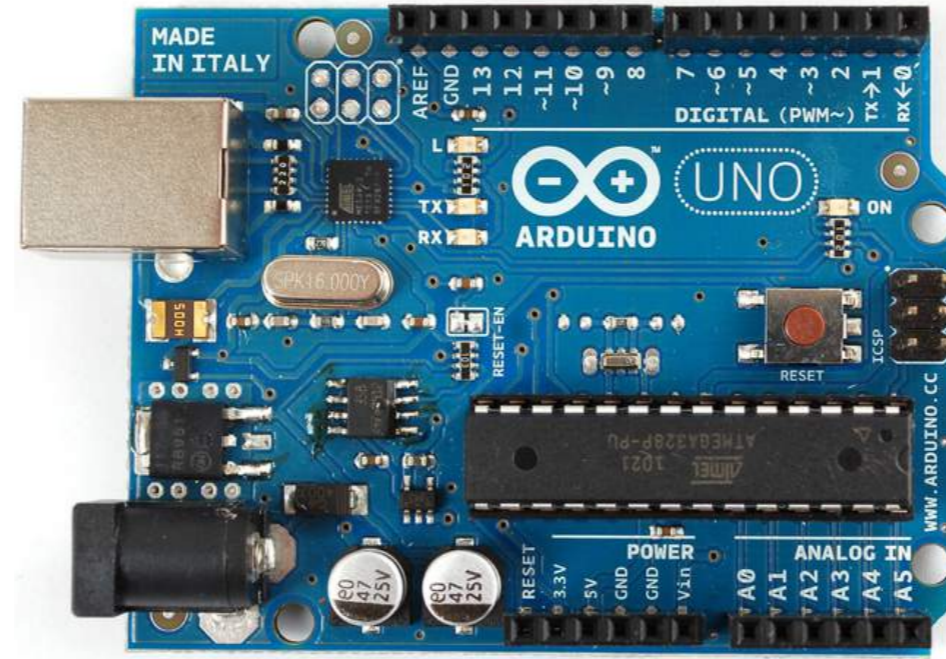
So getting back to teaching physical computing. The key technical elements are sensing, communication, responsive code, and actuation. In order to teach people to do this,



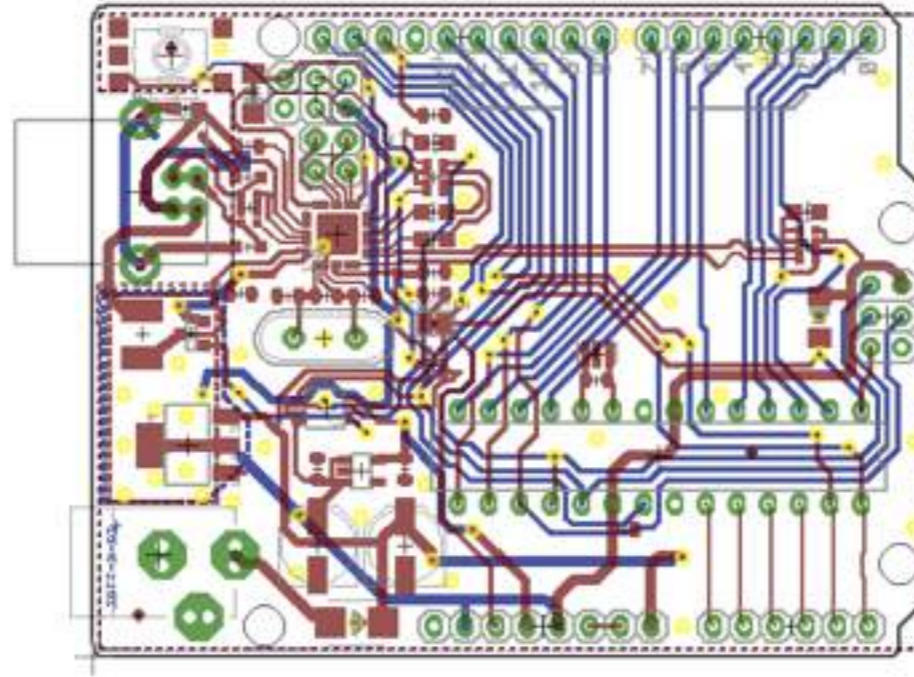
...we introduce them to the microcontroller. It's a bare-bones computer with no physical interface. You attach switches, sensors, motors, lights, and other components to it, in order to build any interface you want. You find them in almost everything these days.



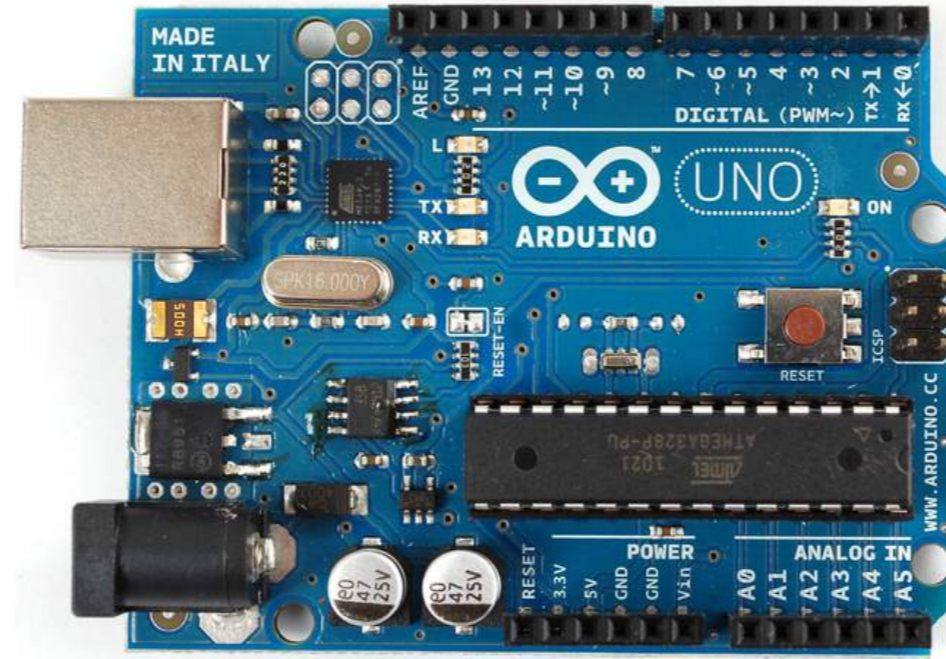
Microcontrollers were historically the domain of experienced electrical engineers, so teaching them was a challenge. About ten years ago I was invited by some colleagues in Italy to co-found a company called Arduino.



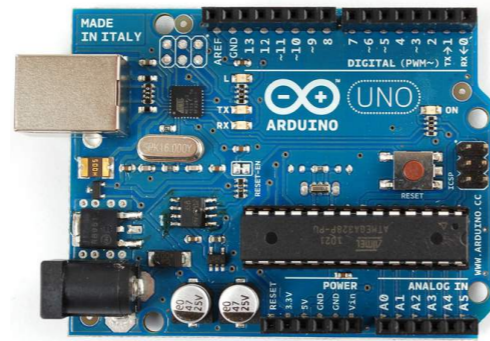
It's a system for programming microcontrollers that's designed for people without previous technical training. Our goal is to get people making interactive electronic things fast.



Arduino is open source hardware. We publish the plans for making the board and the software and code for programming it on our website, in addition to selling the boards. That means that you can make your own versions of the board itself. As a result there are many variations and derivatives on it out there, made by a lot of different people.



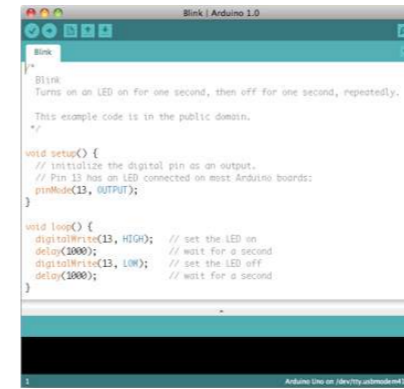
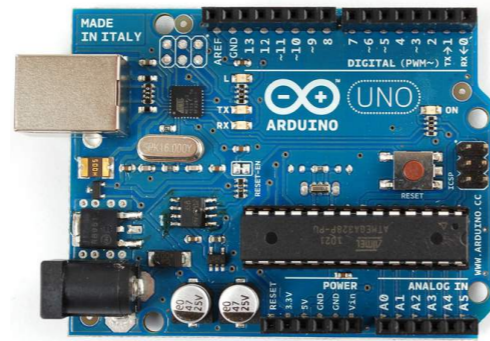
There are actually four elements to Arduino: the hardware:



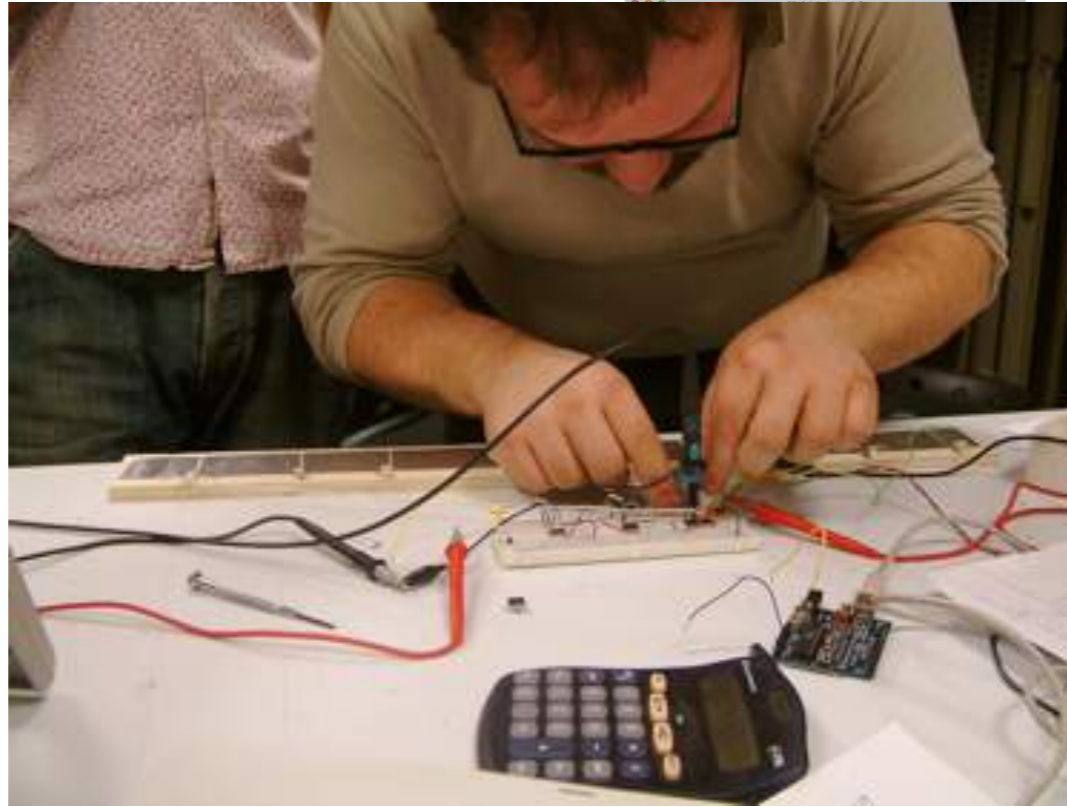
There are actually four elements to Arduino: the hardware:



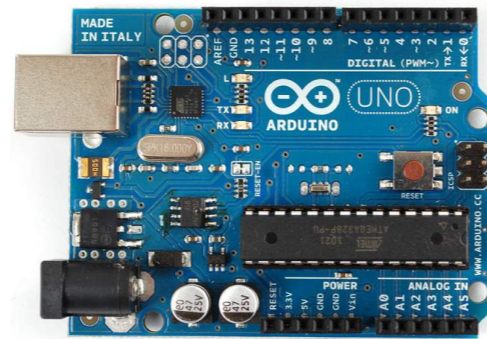
The programming environment, which is cross-platform;



The programming environment, which is cross-platform and written with non-technical users in mind;



Examples for hands-on learning.

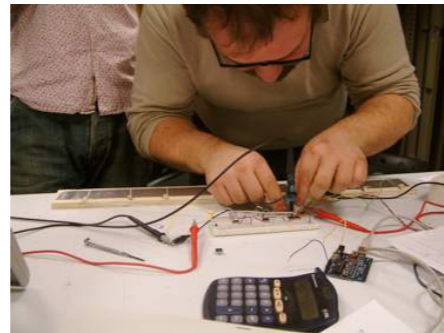


```
Blink | Arduino 1.0

Blink
Turns on an LED on for one second, then off for one second, repeatedly.
This example code is in the public domain.

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

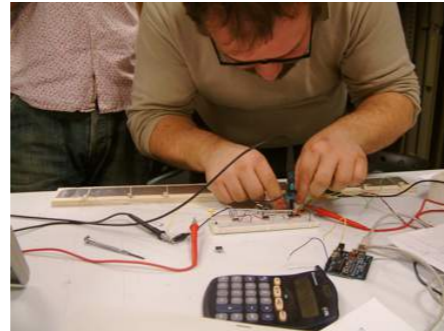
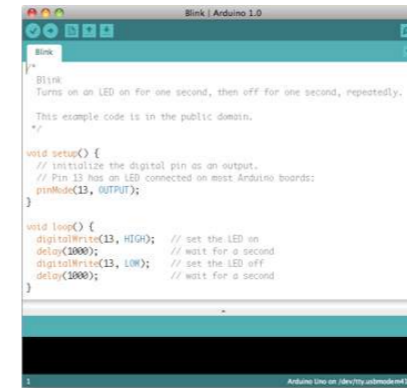
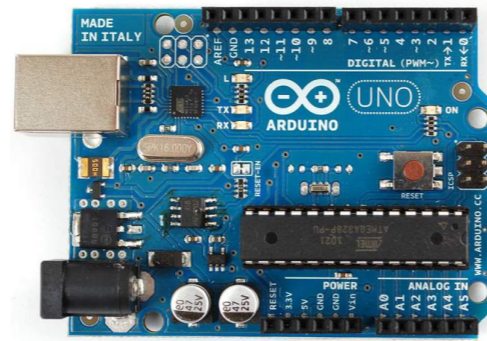
void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);             // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);             // wait for a second
}
```



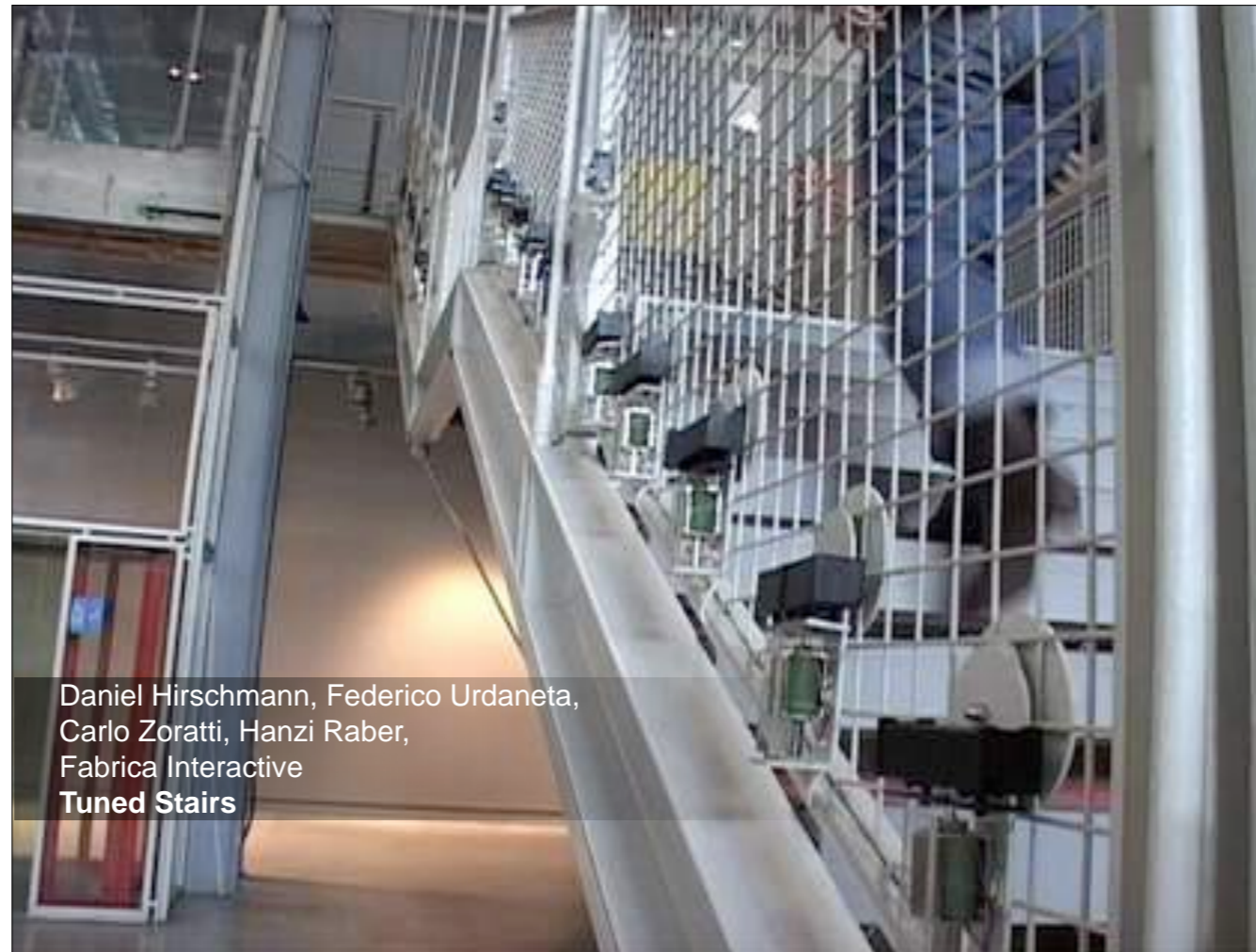
Examples for hands-on learning.



And the community that uses it.



And the community has made some amazing things....



Daniel Hirschmann, Federico Urdaneta,
Carlo Zoratti, Hanzi Raber,
Fabrica Interactive
Tuned Stairs

And the community has made some amazing things. This is one of my favorite early projects, from 2005 or 6. If you know the version of this by Fun Theory that went viral on the net a few years ago, this precedes that by about four years.



Mark Fickett made this RFID Pill meter for his mother.



Amanda Almeida, Erin Lewis, Loretta Faveri
I am Woman hear me beep

This is a kegel organ made by some students at OCAD U in Toronto. You insert it in your vagina and squeeze with your kegel muscles, and the more you squeeze, the higher the note. If you google it, you'll find a video of the creator playing Mary Had a Little Lamb, if I remember correctly.



What we're aiming for, both at ITP and with Arduino, is a technological literacy. These technologies are a part of everyone's lives nowadays, and the more we understand them, the more control we have over how they affect our lives.



Everyone uses language, Sometimes we use it plainly and simply, as a tool to convey information....



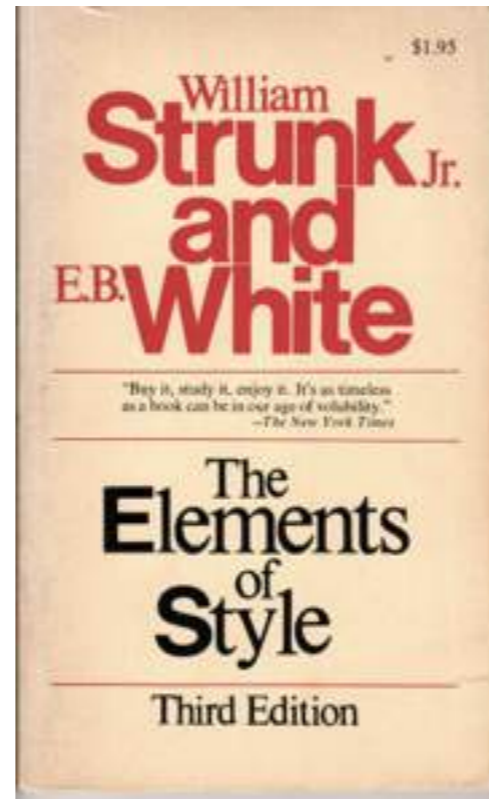
We use it expressively, when we tell stories or sing, or recite poems...



We use it playfully, to make puns, or tell jokes. These are all different contexts for the use of language, just as we have different contexts for the use of tech.



When we're using language casually, or "non-professionally", we leave words out, we use the wrong words sometimes, but we get the message across. We forgive each other a lot for not using language precisely.



There are professional users of language: writers, journalists, translators, diplomats, and many others. We expect professional language-users to know all the rules of the languages they use, and to use them precisely and carefully. But when your writer friends hit you upside the head with a copy of Strunk & White every time you dangle a participle or split an infinitive, it can get annoying.



Likewise, when programmers slap someone with Kernighan & Ritchie every time they format their braces incorrectly, or when EEs hit you with Horowitz & Hill every time you get a resistor wrong, it's just as annoying. Being constantly corrected can actually discourage the learning of good habits.

RTFM

Engineers aren't by nature as forgiving of casual uses of engineering as writers are of casual uses of language. And computers are even less forgiving. If you go to a programming forum on the internet, you're expected to know the "right way" to use a given tool before you ask a question, or you're likely to get told off.



(c) from 16mm film taken by Barney Elliott

In some ways, this is understandable. Engineering is serious business. If you build a bridge wrong, people die. But not every use of engineering skills is a life-and-death matter.



*Tell me more about
what you're trying to do....*

I think we need to change our attitude about this. If we're going to advocate for "programming literacy" or "technological literacy", we need to be more forgiving. There are kinder ways to help. (click)

We need to allow for non-professional uses of technology. I see change happening on this in some forums, and it's encouraging to see.

Programming a simple web server

Start a simple server:

```
from http import BaseHTTPRequestHandler as BaseHandler
import socket
import sys

app = BaseHandler.create_server(('', 8080))

def handler(request, response):
    response.writeHead(200, {'Content-Type': 'text/html'})
    response.write('Hello World!')

if __name__ == '__main__':
    app.listen(8080)
```



The tools for web programming have developed some of this flexibility. It's a lot easy to write a web server program now.

Programming a simple web server

Sinatra (simplest):

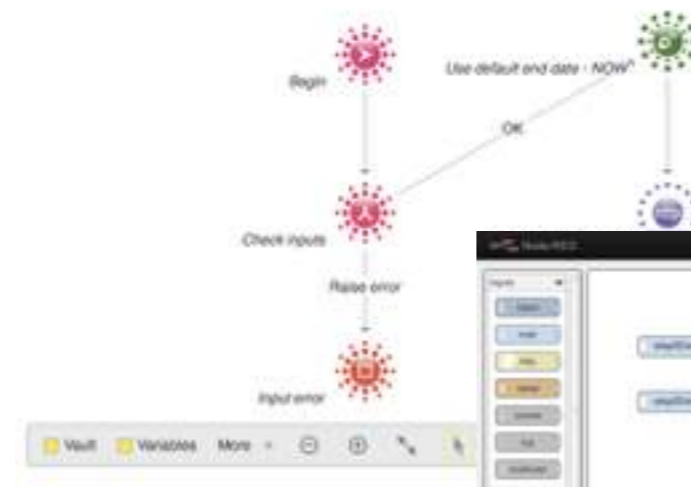
```
require 'sinatra'

get '/hi' do
  "Hello World!"
end
```

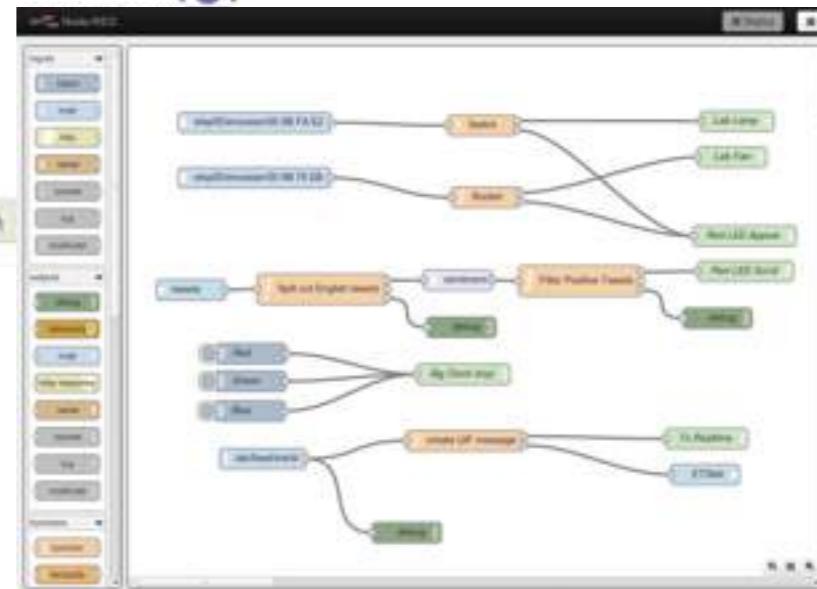


It's not surprising that many people start learning to program with the web. First, it's part of our everyday life already so we can imagine the applications that programming can be used for. Second, languages like Python, Ruby, or JavaScript are much more flexible than C and lower level languages used for hardware.

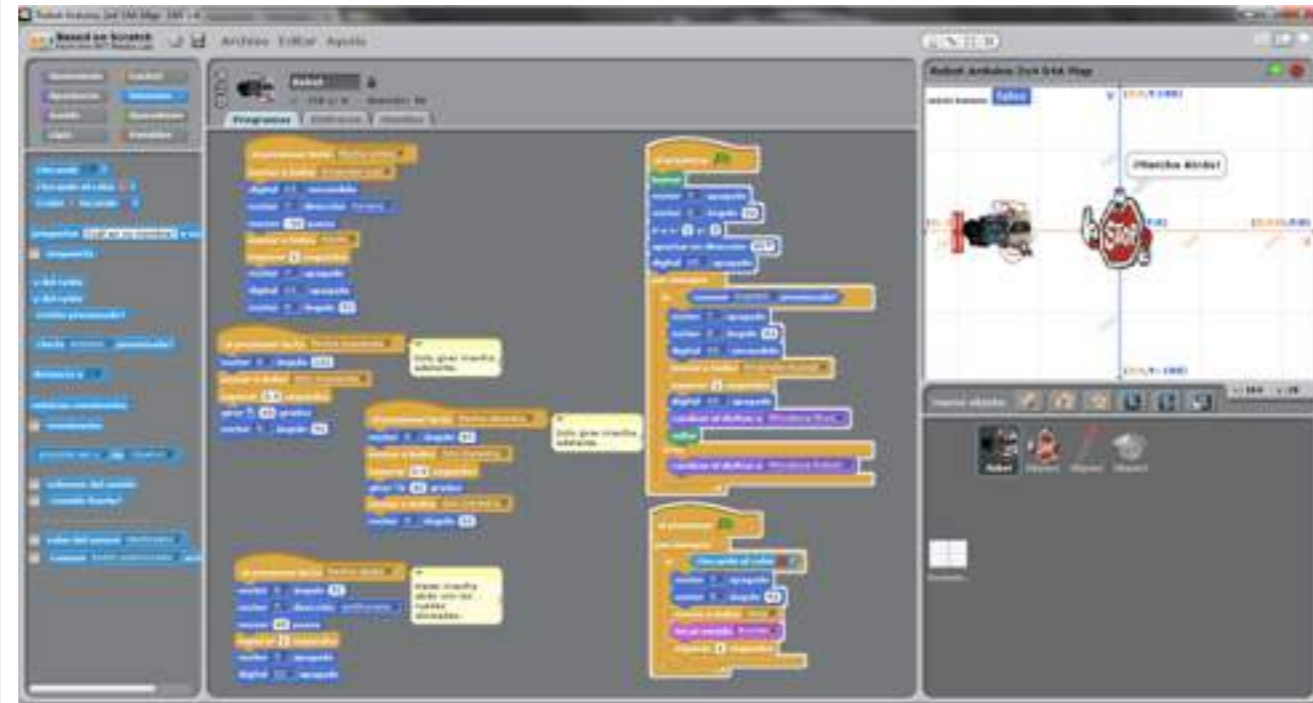
Twyla (temboo.com)



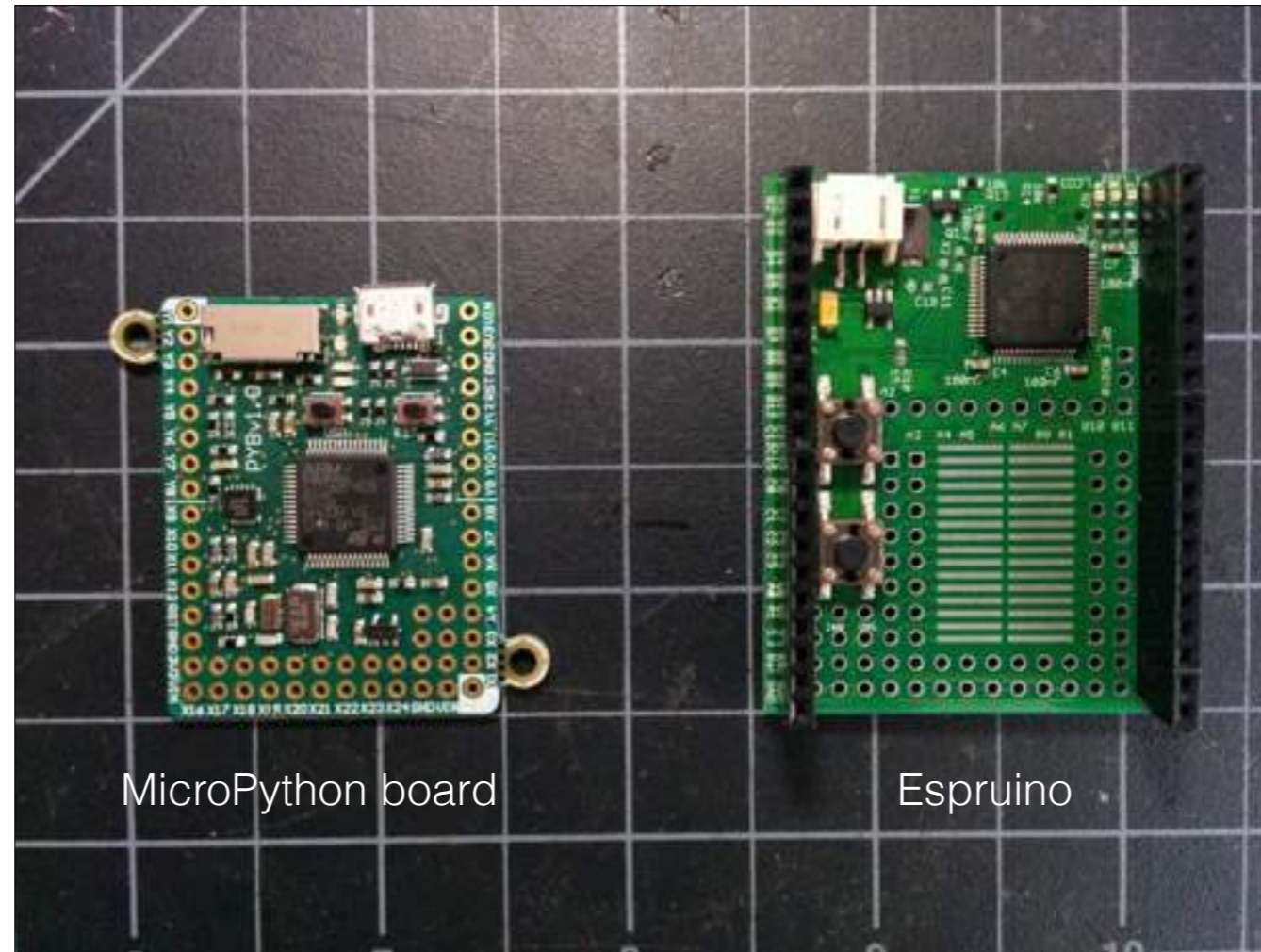
Node-Red (nodered.org)



Tools like Temboo's Twyla and (click) IBM's Node-red are abstracting network transactions into graphic languages as well. Network transactions lend themselves to simpler abstractions very well.



In hardware, and especially in microcontroller programming, we need to catch up. Even the best graphic tools, like Scratch for Arduino, don't remove the structural complexity of programming. They just remove the syntax problems. But this is beginning to change.



MicroPython board

Espruino

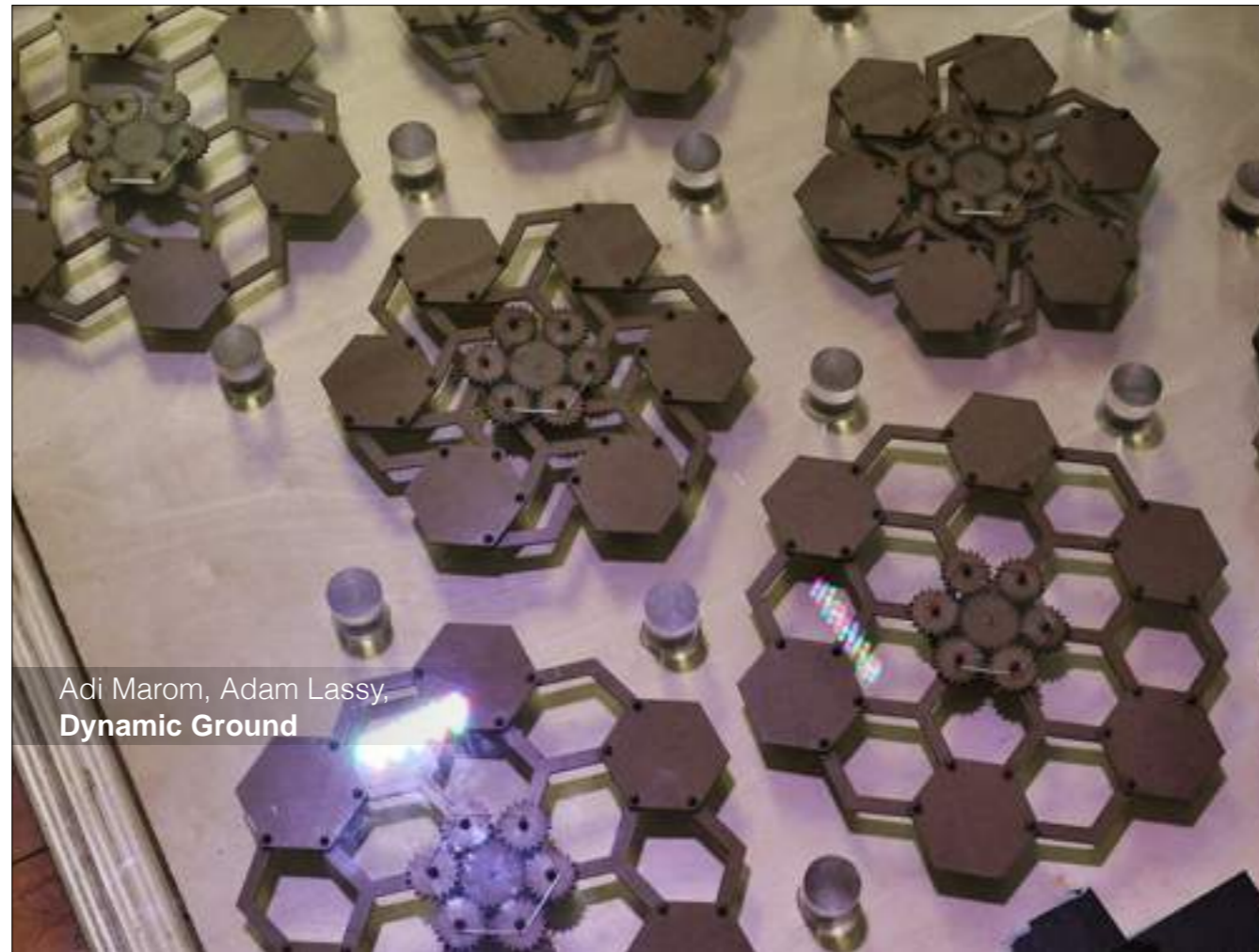
These are two new entrants to the market, the Micro-python board and the Espruino board, they'r running Python and Javascript respectively without an OS. We're also starting to see some interesting non-textual process flow tools and code generators as well.



We don't need to make everyone better programmers.
We need to make programming better for everyone.

For me, the changes we're seeing in programming and electronics come down to this: We don't need to make everyone better programmers. We need to make programming better for everyone.

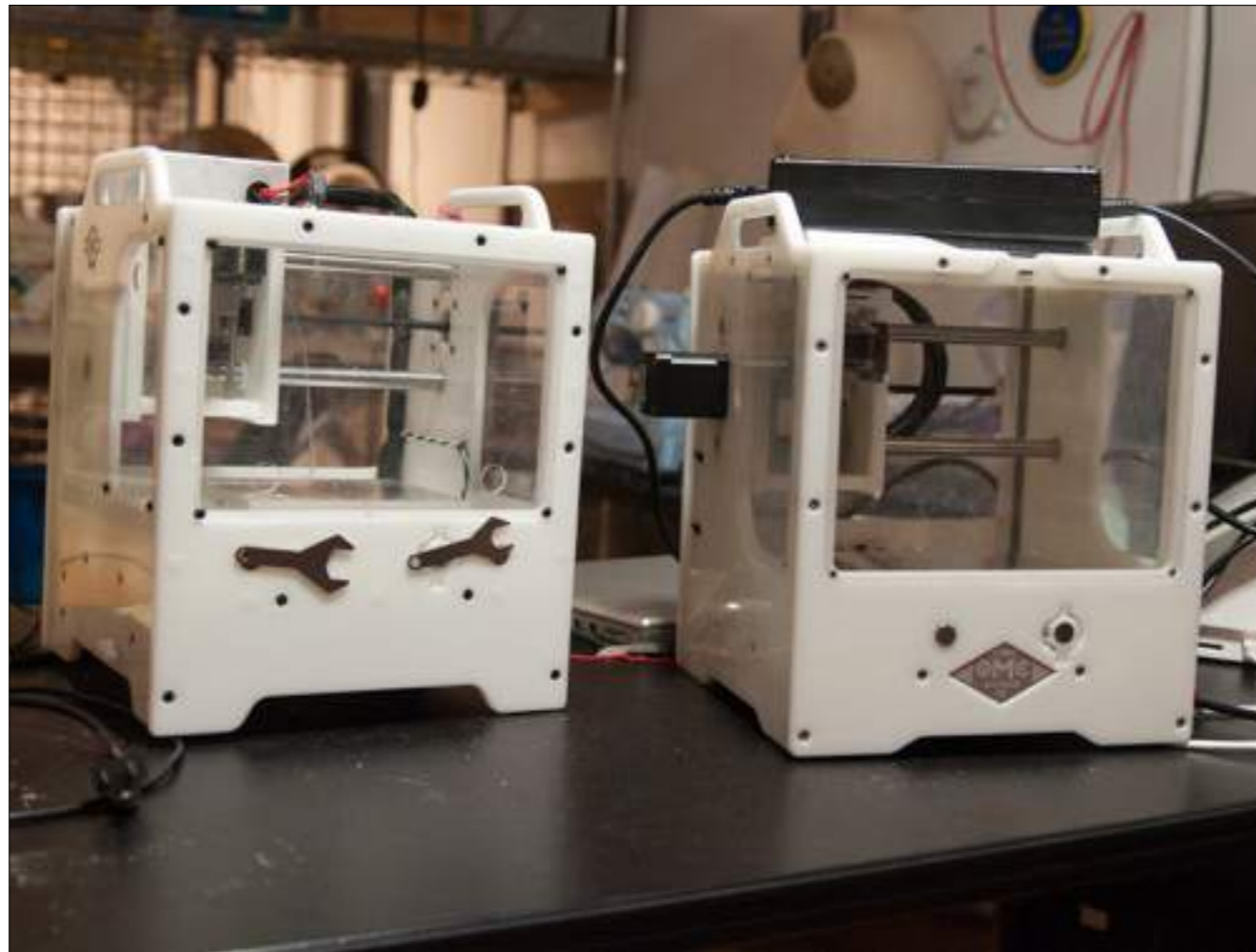
We've talked code literacy and electronics literacy, but there's another literacy I want to talk about...



If Arduino was the porta-pak for interactive electronics, then digital fabrication tools are the porta-pak for physical design. They make it possible for people with little training in fabrication to make precision mechanisms and structures. At ITP, they've expanded the physical computing curriculum by making it easier to teach hands-on mechanics and construction techniques we couldn't previously touch.



Automated cutting tools like laser cutters and mills are actually the most important digital fab tools, in our shop. These tools capitalize on what Ikea's known for decades: training humans to cut and shape is hard, but training them to assemble is not so hard. They don't totally automate the process of building, they just simplify the harder parts.

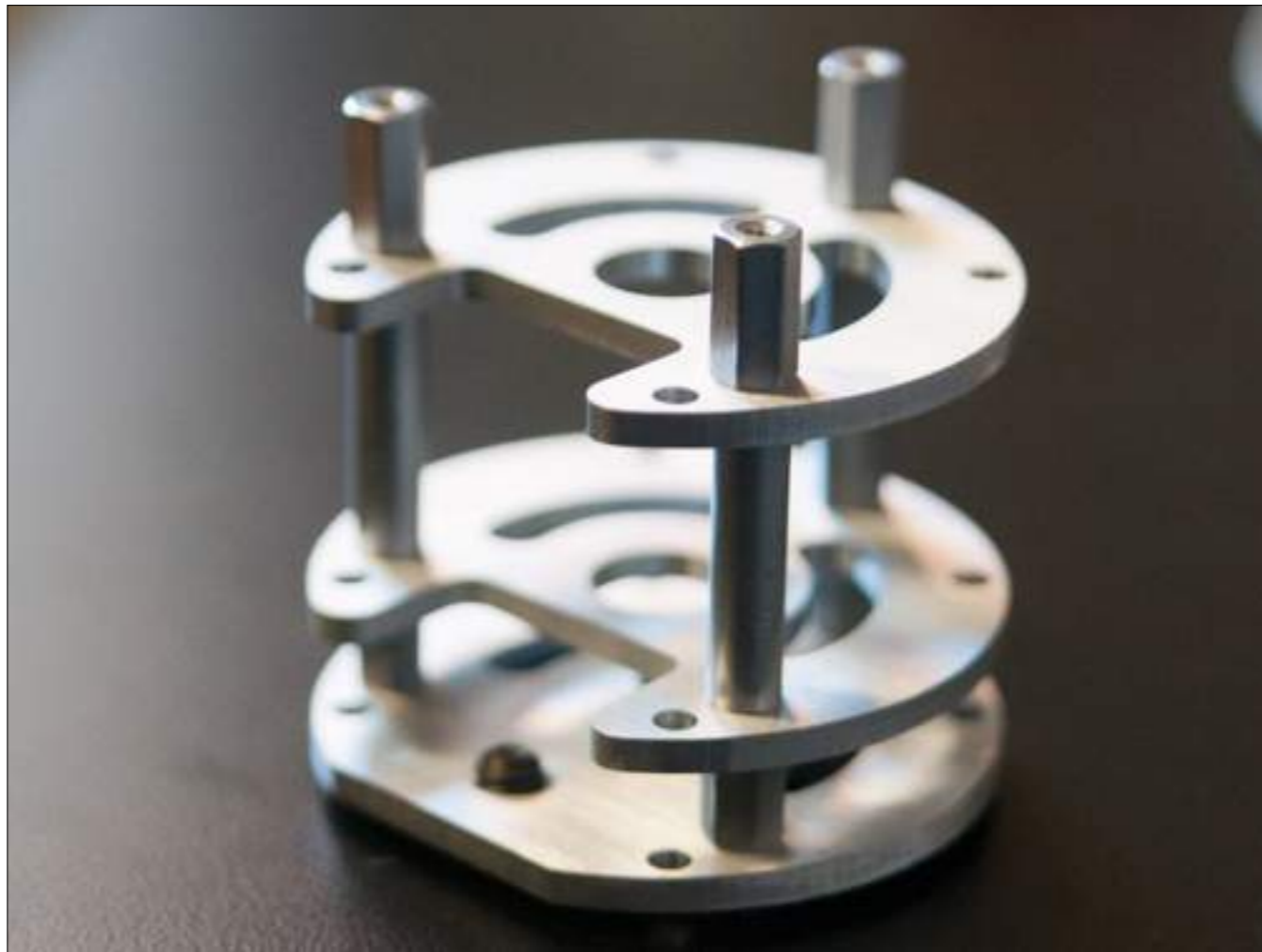


By now you're probably sick of hearing about 3D printers, and you've seen enough of your office-mate's 3D scanned head, so I won't show you those. But here's my favorite new tool: the Othermill. Normally CNC mills like this are both expensive and unforgiving of error in their operation...

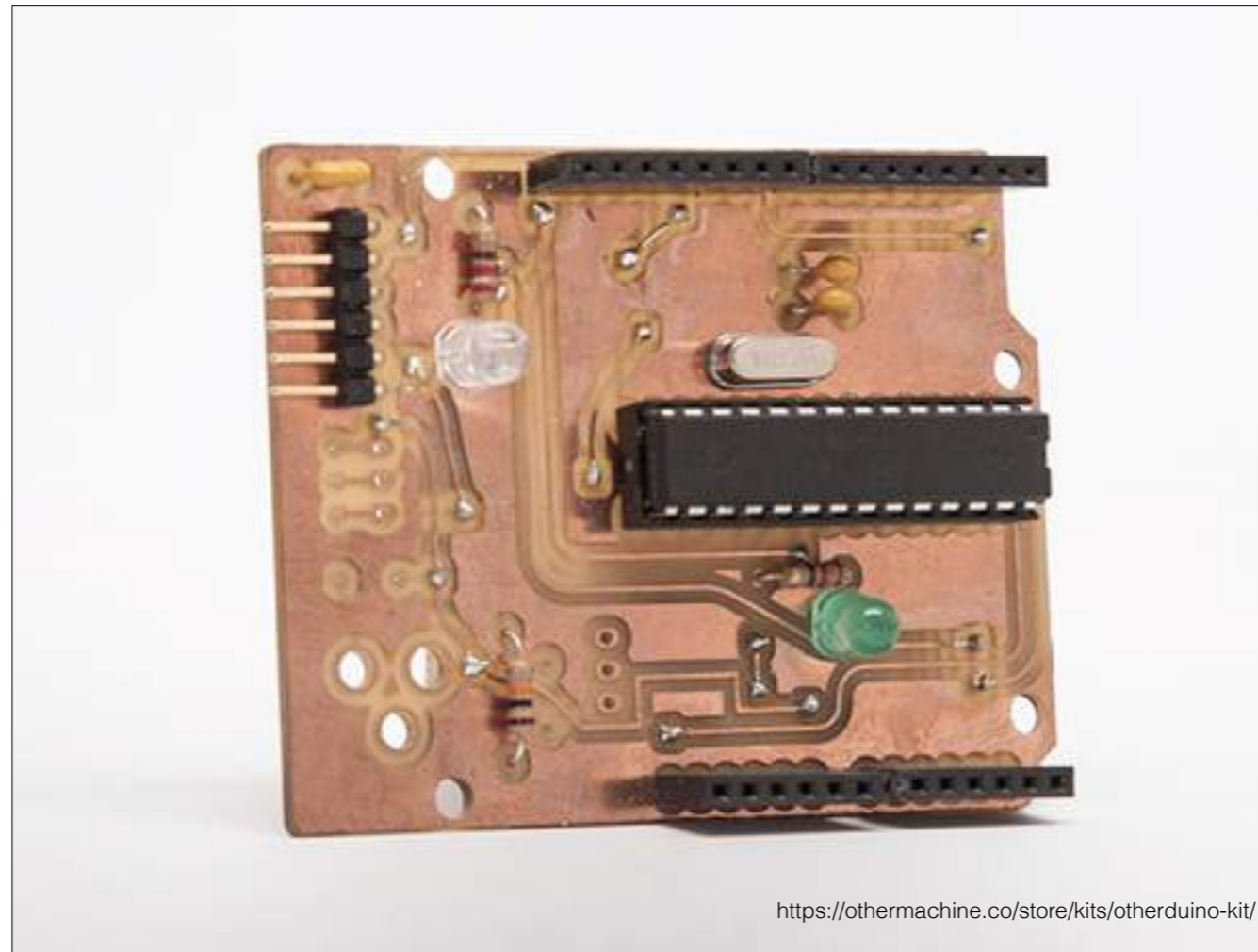


...but the Othermill folks wanted to make a tool for a general audience. So their price is around \$2K, and their software is free and easy to use.

The software's OSX only, which has pissed off some traditional mechanical engineers, but it's gained them a whole new audience.



And with it, you can make things out of metal and other sturdy materials, like this motor mount.



<https://othermachine.co/store/kits/otherduino-kit/>

Or you can mill circuit boards, like this Arduino derivative. I have to give my colleague Ben Light credit for his teaching and tutorials in making this such an indispensable tool for us.



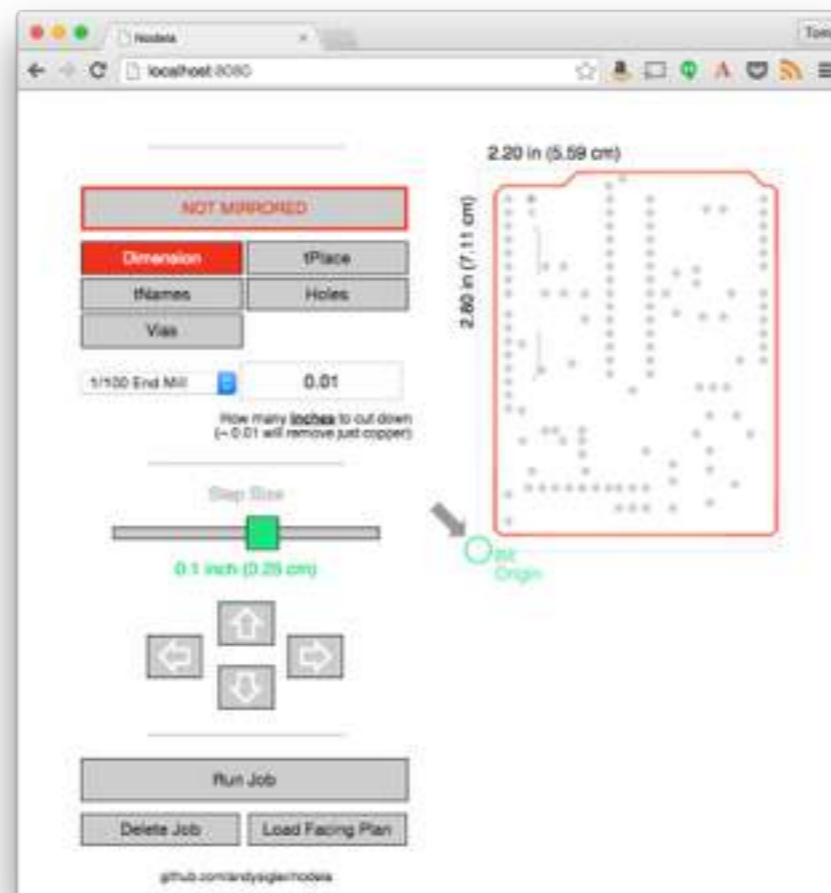
Things get really interesting when you combine it with tools like this machine, which allows you to precisely place and solder millimeter-scale electronic components. This particular machine is great because it teaches you how the more automated pick & place robots work, by placing one component at a time, by yourself. Again, it doesn't automate the whole process, just the hard part.



These machines are still not consumer-level but they're getting closer. They've reached desktop scale, and small-business affordable, which is a huge step forward.



To me, what makes tools like these so interesting to an audience like this one is that they're connected devices. They can be programmed if you know the protocols, and addressed using standard serial ports or network ports. There's no reason you couldn't make your own interface if you find the existing one too complex. That's what our resident researcher Andy Sigler did.



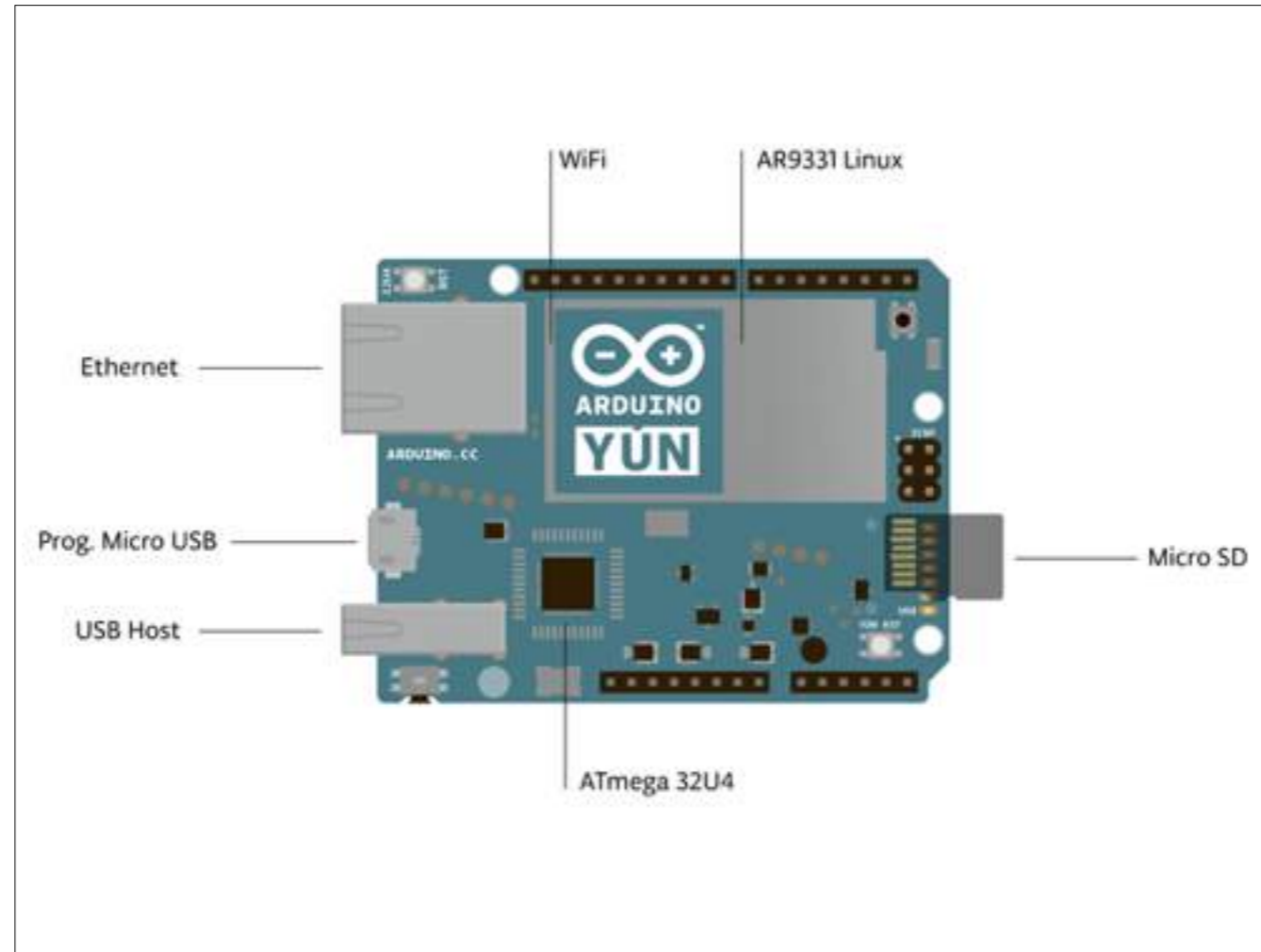
<https://github.com/andySigler/nodela>

Andy wrote a program called Nodela in node.js. He wrote it to simplify the user interface for the Roland CNC mill you just saw. The original software is written for Windows only and its not very user friendly. Nodela runs in a browser. It doesn't do everything the mill can do, but it makes it a whole lot easier to mill circuit boards, our main use of the machine.



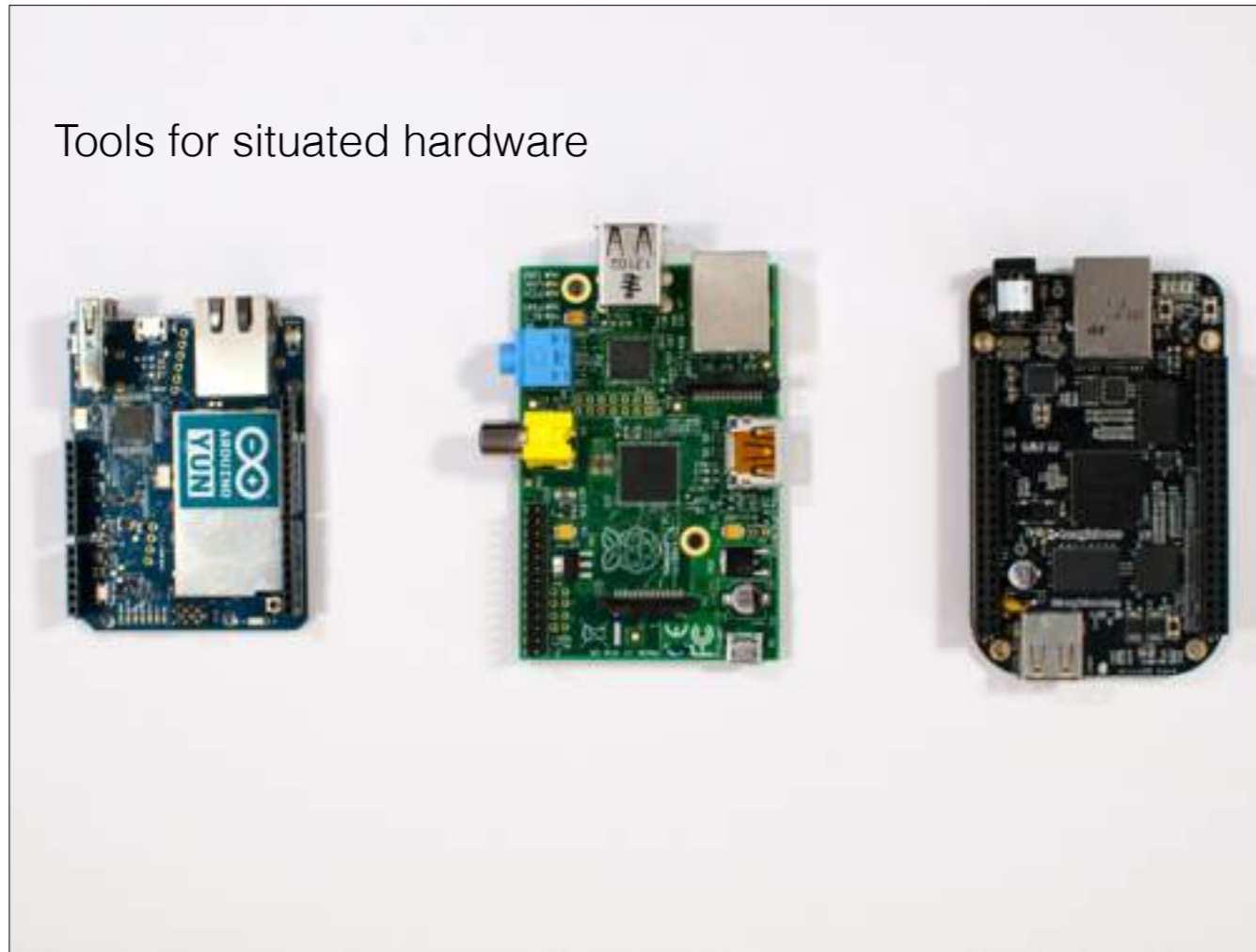
<https://github.com/voodootikigod/node-serialport>

Projects like nodela are really exciting to me at the moment, because they show that controlling devices doesn't have to be limited to enterprise programmers. If you can program in Javascript, you can communicate through a network port or a serial port, and if you know a little HTML along with that, you can make browser-based interfaces for physical devices now.



The Arduino Yún is designed for building those kinds of devices. It's a bit like other embedded linux computers you may have heard of, like the Raspberry Pi or the BeagleBone, but we combined the little linux machine with a realtime microcontroller and simplified the programming workflow.

Tools for situated hardware



Eleven years ago, my colleague Clay Shirky wrote about the rise of software applications that didn't scale, but solved one particular group's problems well enough. He referred to it as "situated software", and he credited accessible web tools for it. We speculated at the time about whether there would be such a thing as "situated hardware". I hope we're going to see that become a reality.



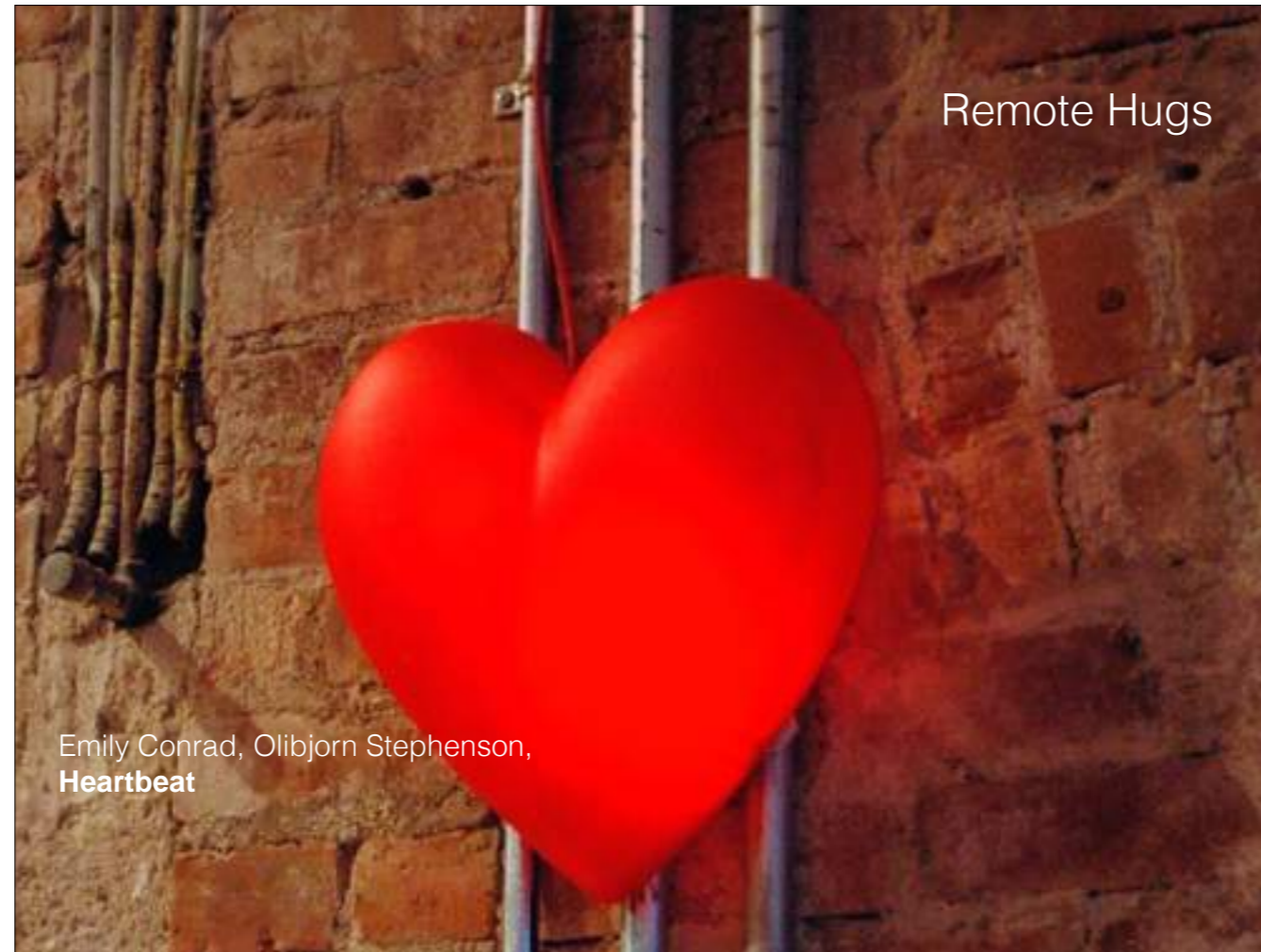
What makes me happiest is that tools like the these enable personal expressiveness. For Christmas last year, my partner asked if we could make it so that every time you ring the bell on our Christmas tree, an angel gets its wings, just like in “It’s a Wonderful Life.” With a Yún, a tin bell, and a string of programmable LEDs, I was able to give her that present.

Make Things That
You Will Use Yourself

Matthew Epler,
Kinograph



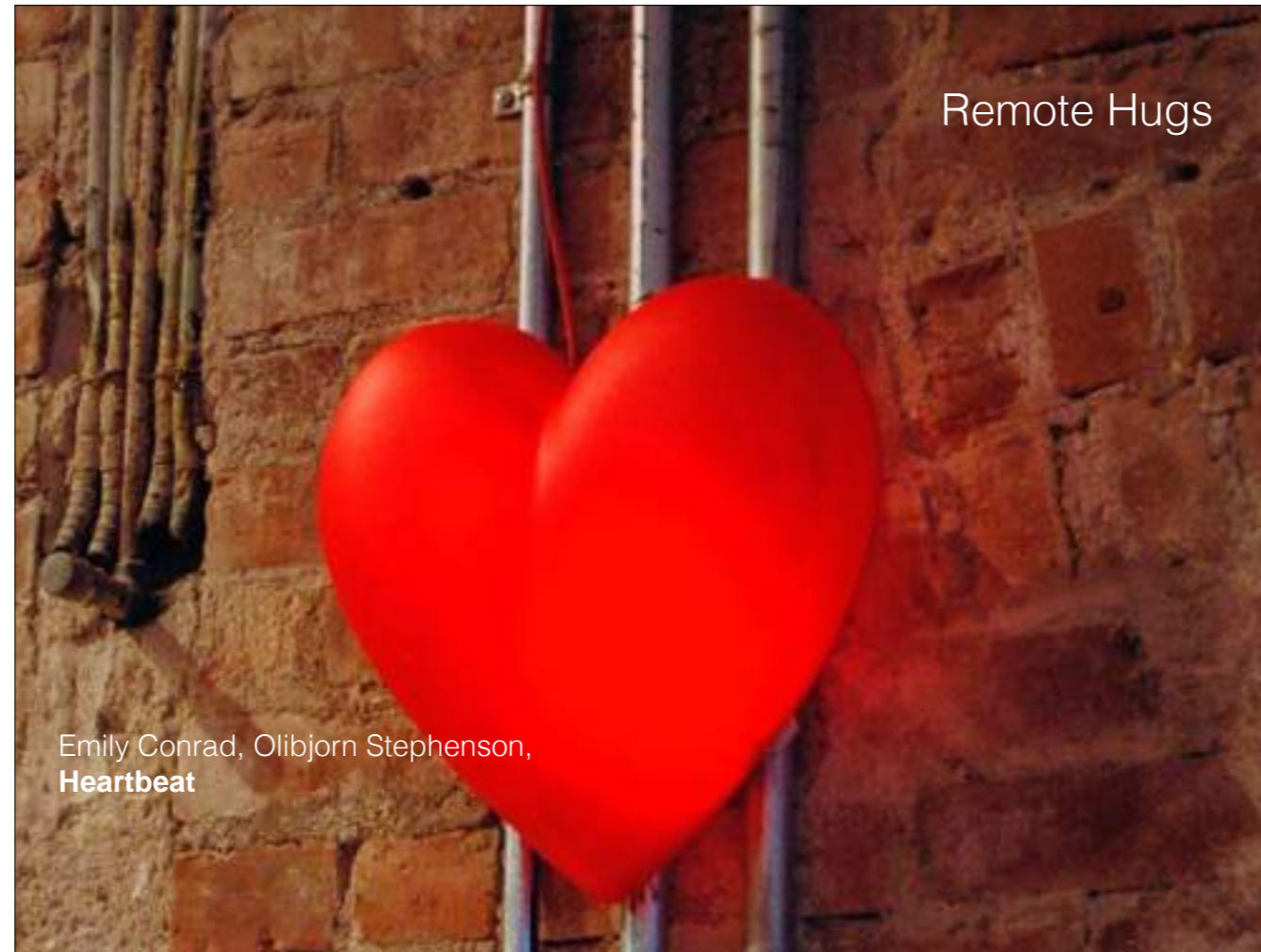
Here's a more practical example. Matt Epler, a filmmaker, threw together this film digitizer with a Raspberry Pi and a few motors for much cheaper than a professionally built rig would cost him. And he published the plans for the whole thing for others to copy. It uses his existing DSLR for capture, automated by the Pi.



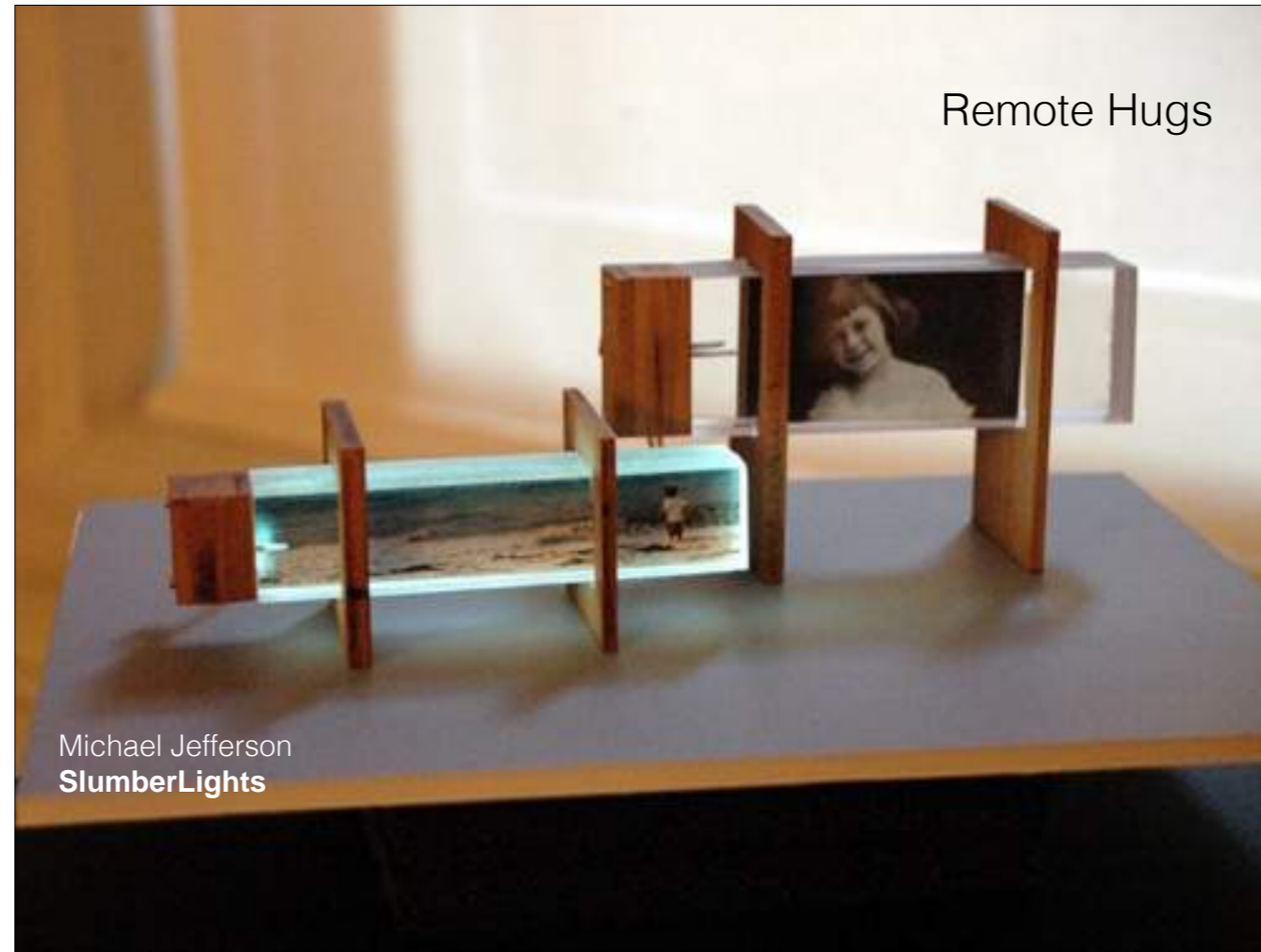
Remote Hugs

Emily Conrad, Olibjorn Stephenson,
Heartbeat

I've been teaching people to build Connected Devices or Networked Objects since 2001. One of the most common projects that comes up (and I've seen it in other people's classes on embedded networks as well) is what I call the Remote Hug.



A remote hug is generally an object or a pair of objects held by two people who already have a relationship. The idea is that when I'm thinking of you, I do something to my object, and it sends a message to your object. Your object then lights up, or dances, or does something to tell you I'm thinking of you.



Remote Hugs don't convey data, they perform a social function. They support phatic communication over distance.

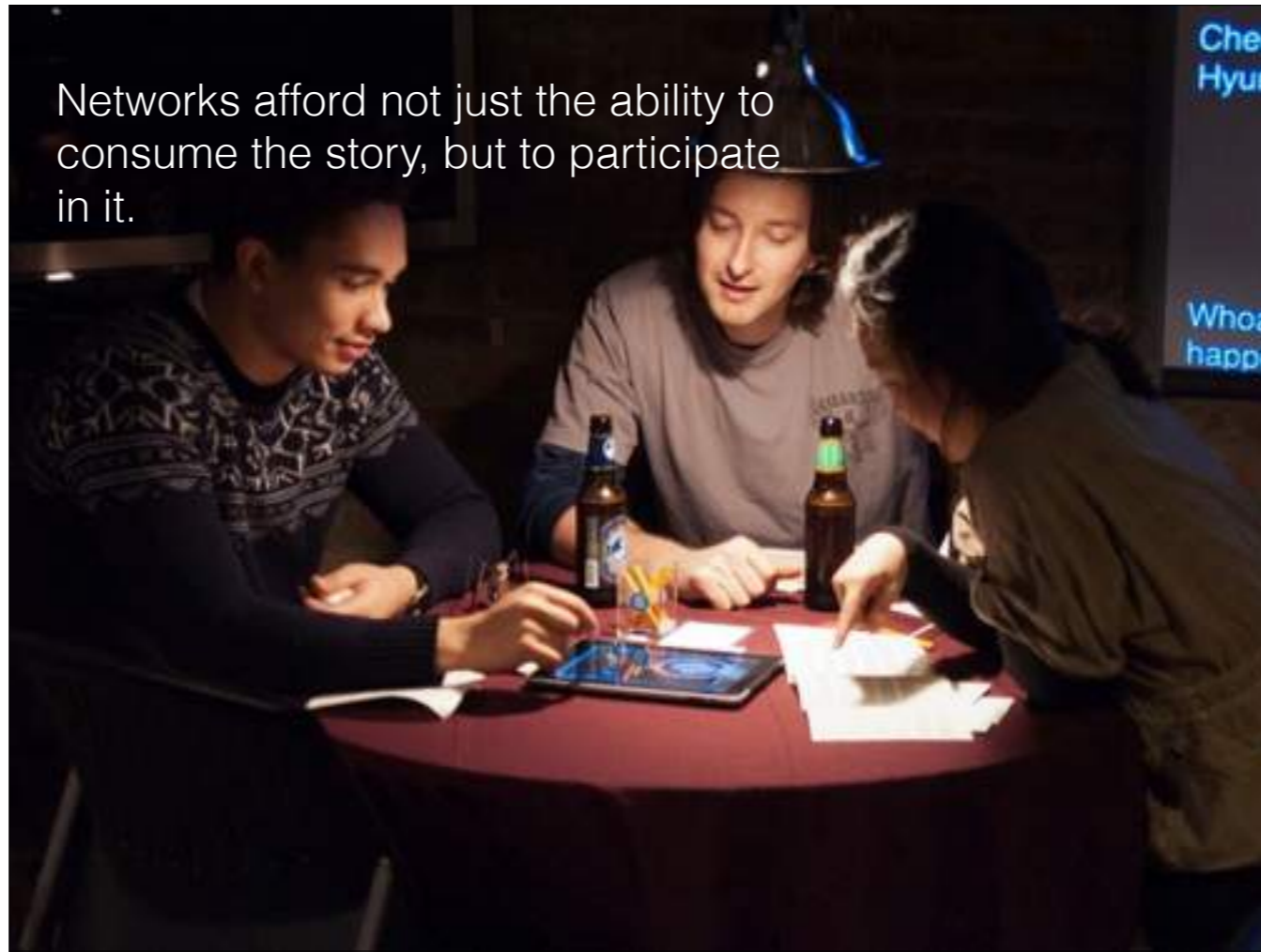
Reduced to data, the gesture of the remote hug is meaningless. the hug's power lies in what its behavior evokes, not in the information it conveys, or in the meaning the objects have to the two users because of their shared history -- paired photographs remind us of a shared experience, for example.

Remote Hugs

Doria Fan
Networked Flower

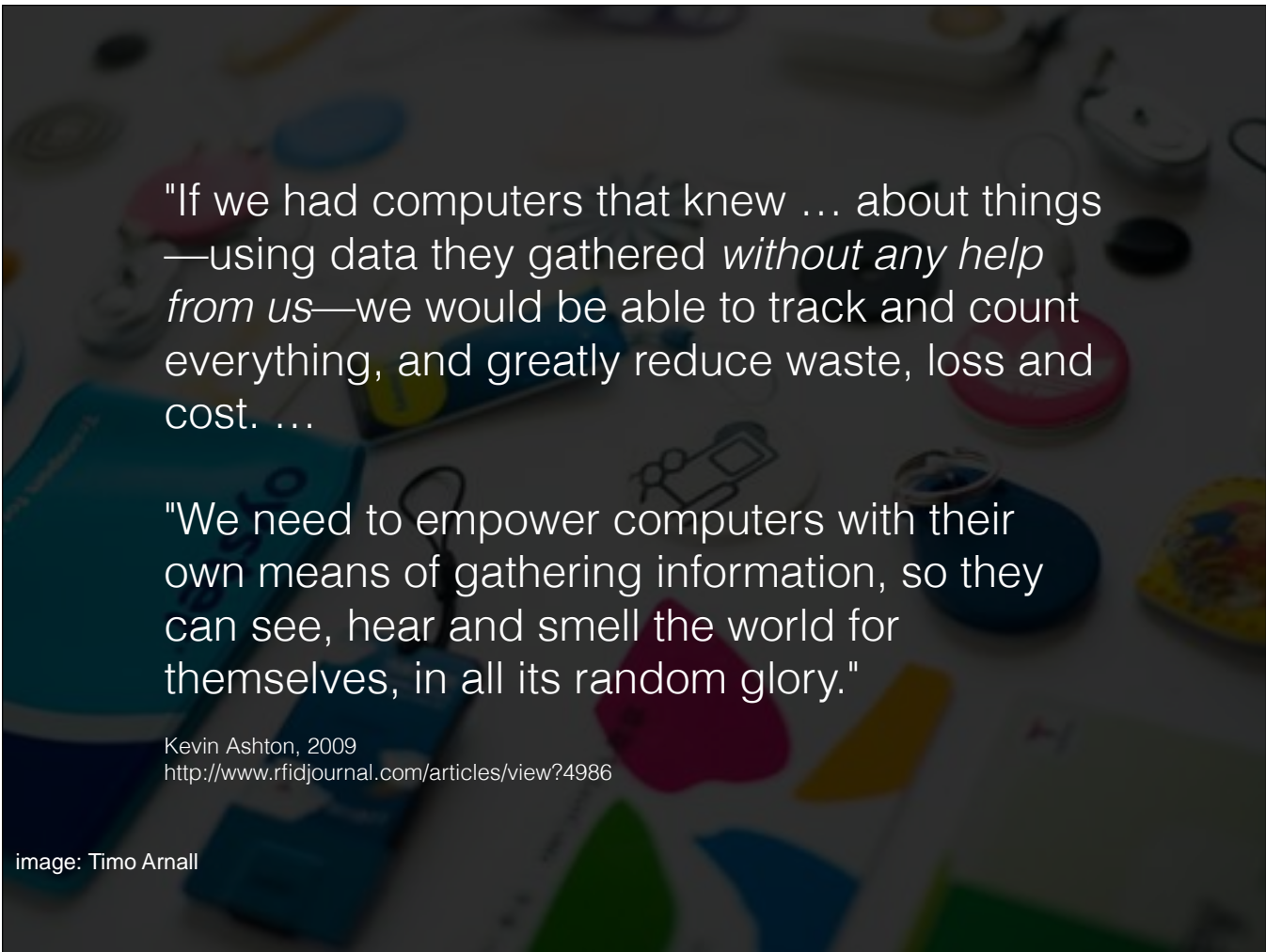
I don't think that remote hugs are scalable as a product. Every one of them I've seen made was an intimate work, shared between the maker and someone they loved. The fact that they come up so often in embedded network classes in programs from design to engineering to performing arts suggest that we have a widespread interest in phatic communication over networks, just as we do in unmediated life.

Networks afford not just the ability to consume the story, but to participate in it.



The worldwide web got interesting when it started to support the ability for users to generate content, to participate in the story, instead of just consuming it.

Networks can support and encourage cooperation, we know this. And we know that given the opportunity to engage each other, people are inclined to do so, and often for beneficial or even altruistic reasons.



"If we had computers that knew ... about things
—using data they gathered *without any help*
from us—we would be able to track and count
everything, and greatly reduce waste, loss and
cost. ...

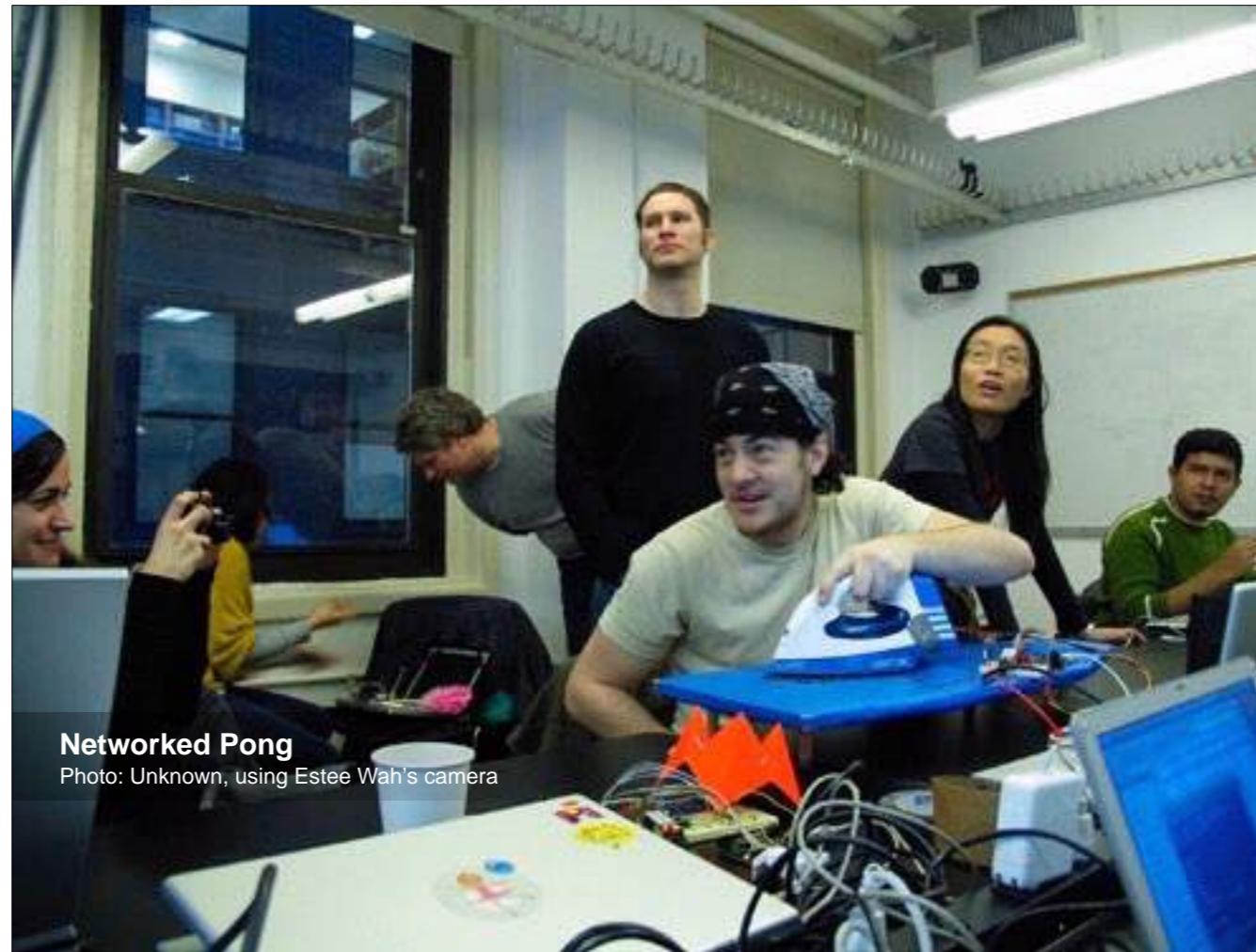
"We need to empower computers with their
own means of gathering information, so they
can see, hear and smell the world for
themselves, in all its random glory."

Kevin Ashton, 2009
<http://www.rfidjournal.com/articles/view?4986>

image: Timo Arnall

But this is not the vision of the Internet of Things. It's about total data collection, not engagement.

Ashton got the last part of that right, but the rest terribly wrong. Things are just the media through which we perform our relationships. And data is not the part that gets people involved, it's communication.



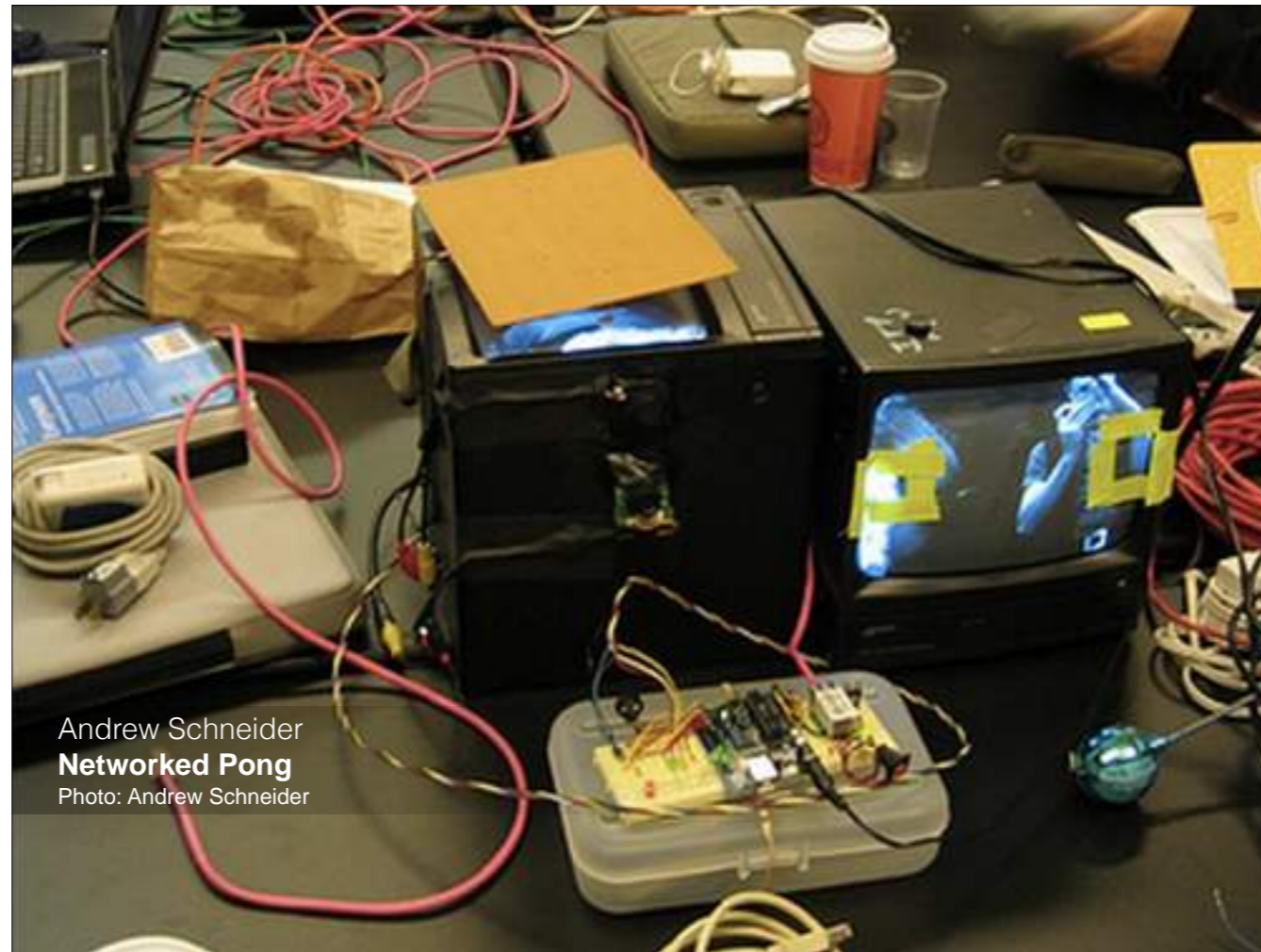
Networked Pong

Photo: Unknown, using Estee Wah's camera

I've been learning a lot about this through a networked game I play with my class. It's a variation on the Pong-like ball games of the 80's: a ball drops from the top of the screen, and the players all have to keep it alive; for each paddle the ball hits before it drops off the screen, the whole group gets one point. Each individual's score is shown only when the game is over.



Over several iterations, I've changed the game from competition to cooperation, and what I've found is that the students tend to want to play more when they have to cooperate. They communicate more about how to play as well.



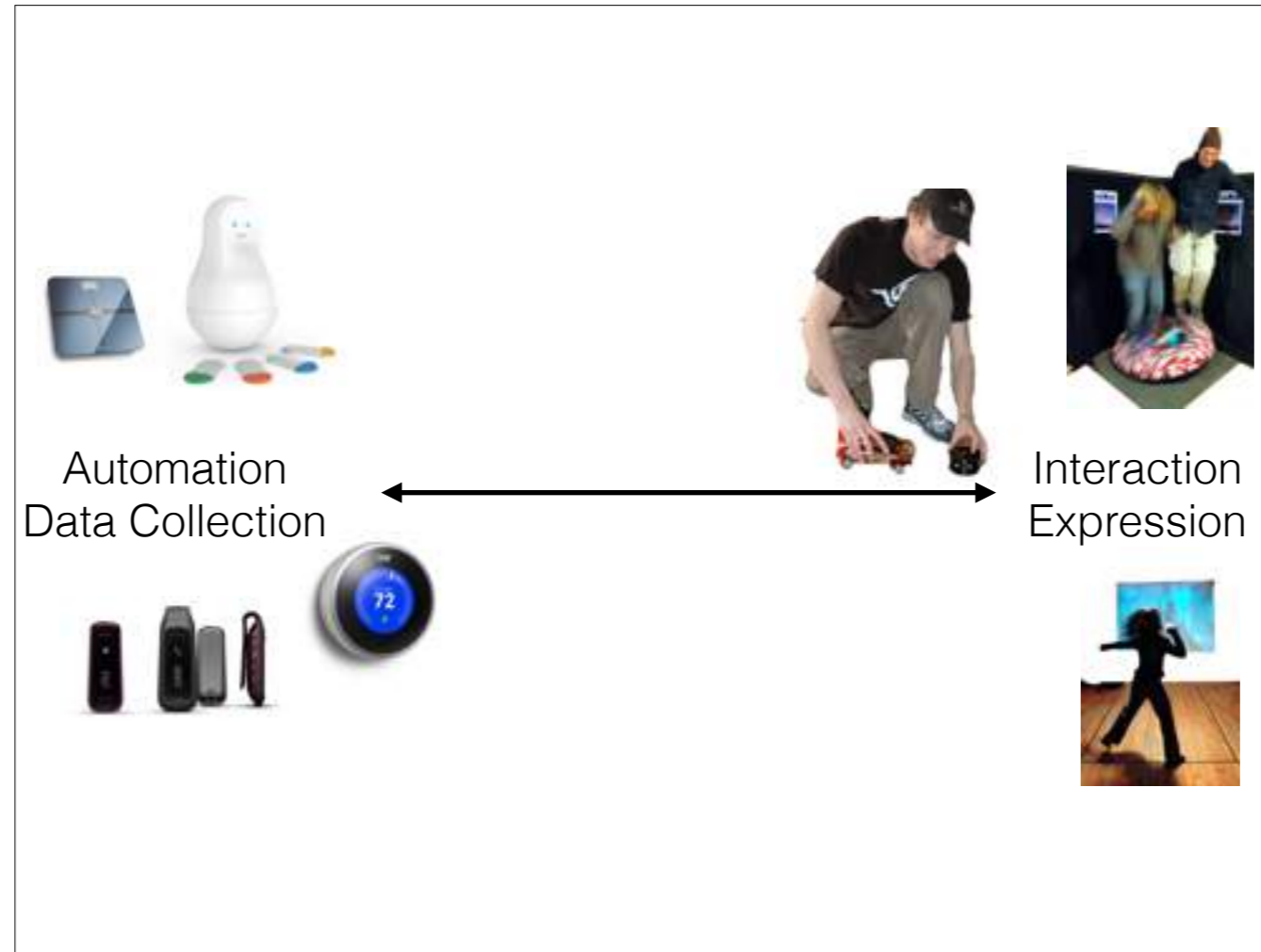
Andrew Schneider
Networked Pong
Photo: Andrew Schneider

This is Andrew Schneider's pong client. He's got two photocells on the screen where those two yellow squares are, and a networked microcontroller that reads them for changes in the light of the image. Then,



He stripped off his shirt and used his skin to trigger the paddle moves. **Spontaneity of physical expression is a big part of the game. The players appreciate each others' style as much as their skill.**

Our current network interfaces support plenty of spontaneity when it comes to purchase or verbal expression, but we're still far behind in building systems for spontaneous physical interaction and expression in networked settings.



IoT's core values are automation and data collection, whereas I think the more fertile ground is in interaction and expression. And with that I'll leave you with one last thought,



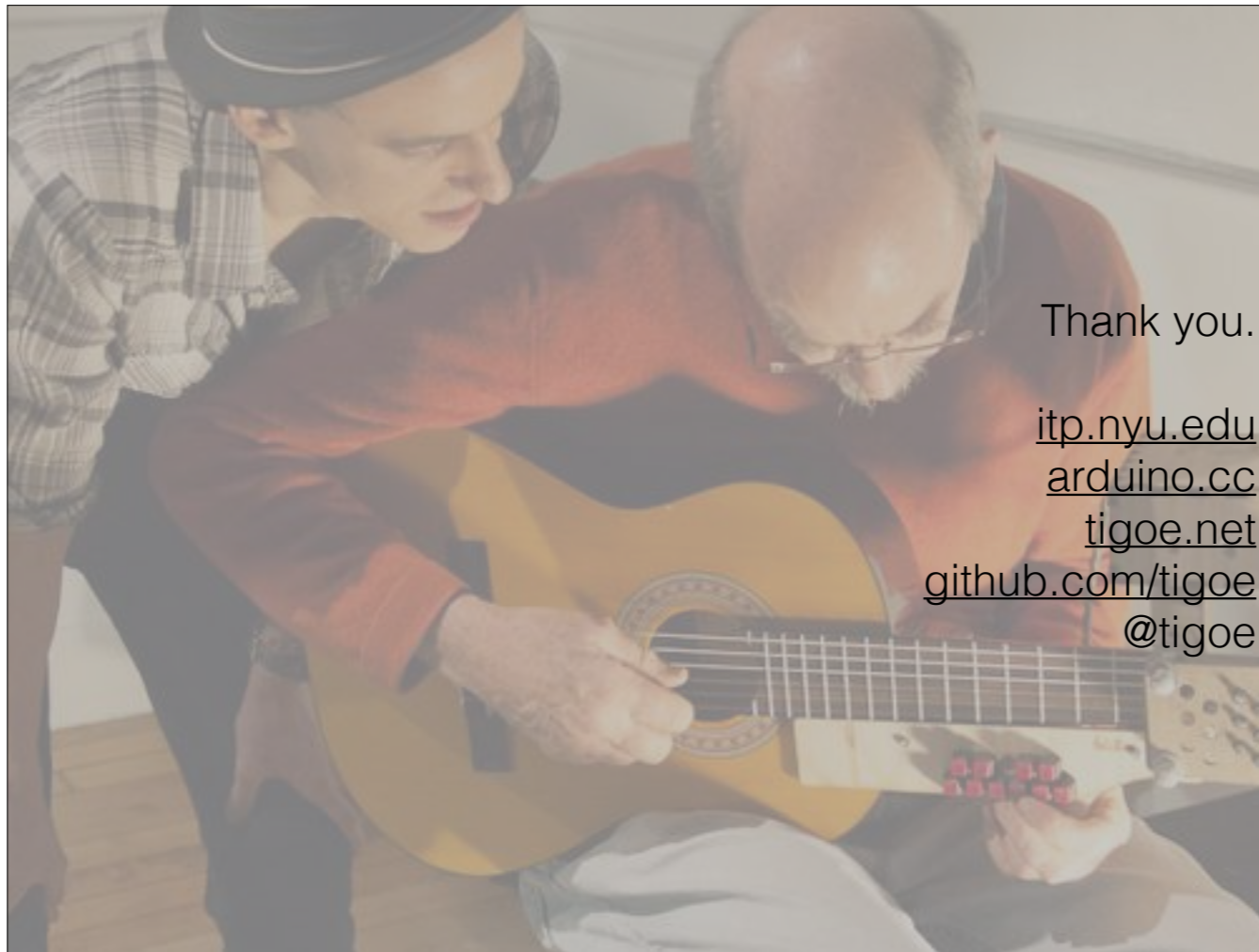
A story about Justin Lange and his dad, a guitarist. This, to me, is physical computing at its best, and a good example of situated hardware. It doesn't scale up, but it's perfect for the person for whom it was designed. (click)



The things we make are less important
than the relationships they support.

Justin Lange
Folkbox

And it's a reminder that the things we make are less important than the relationships they support.



Thank you.

itp.nyu.edu

arduino.cc

tigoe.net

github.com/tigoe

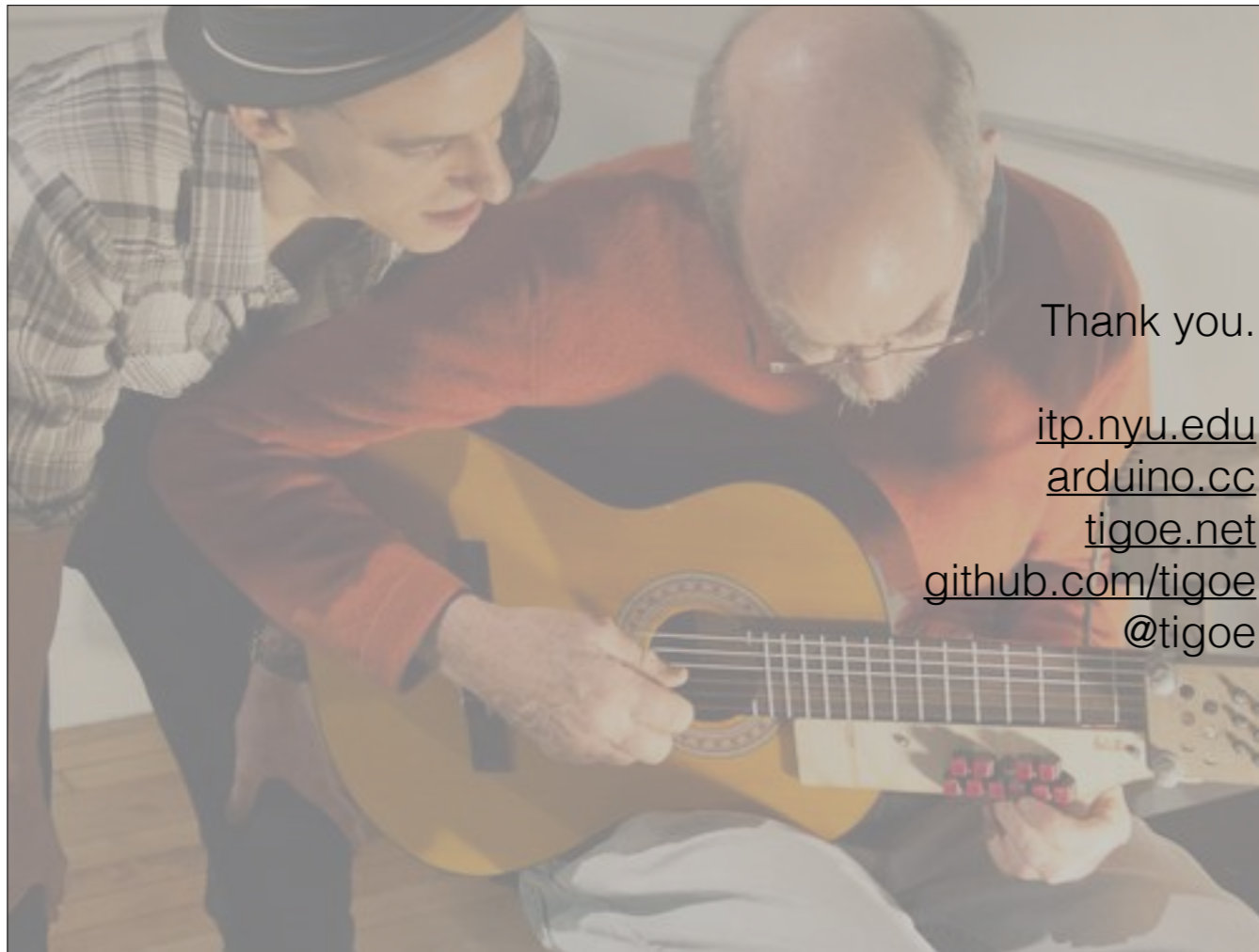
@tigoe

Thank you.



github.com/tigoe/BluetoothLE-Examples/sensorTagSlideshow

One last thing: The slides were all controlled using node.js, appleScript, and a Texas Instruments sensorTag. The Bluetooth low energy part of this I learned from Don Coleman, who's speaking later this morning, so consider this a plug for his talk.



Thank you.

itp.nyu.edu

arduino.cc

tigoe.net

github.com/tigoe

@tigoe

Thank you.