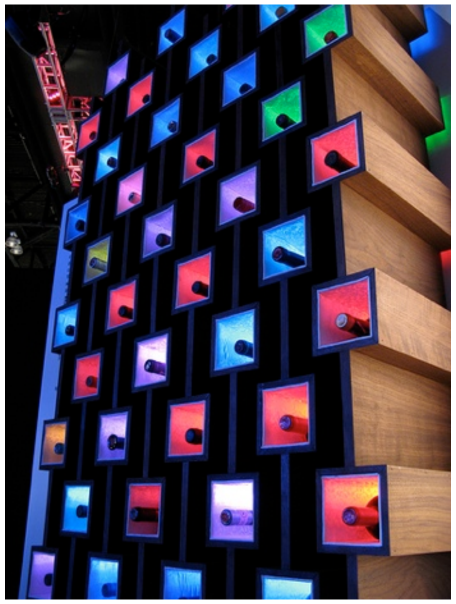
Hacking USB HID for Easy Tethered Ubicomp

Tod E. Kurt / ThingM

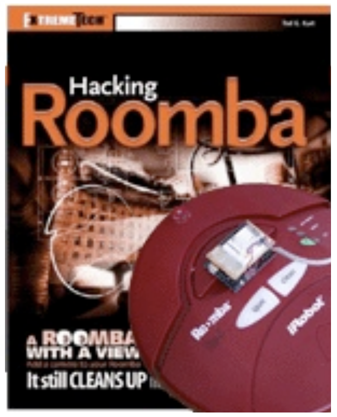
Sketching2010
24 July 2010

todbot does...

WineM

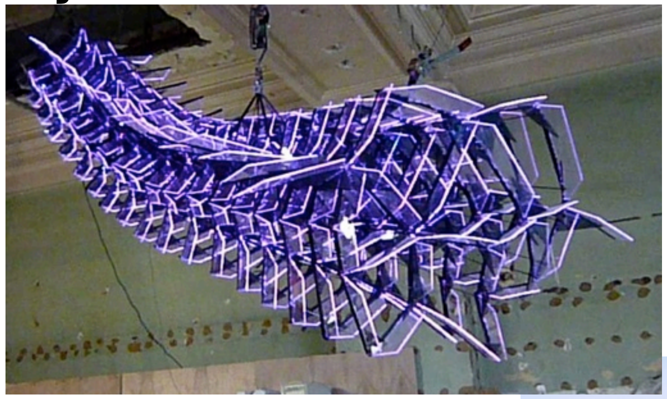


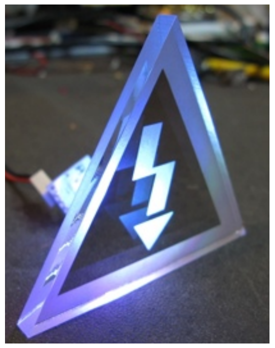
Spooky Arduino



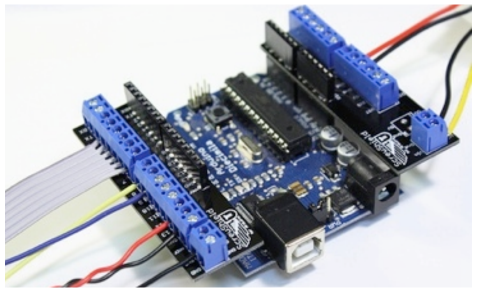
THING M

Crystal Monster



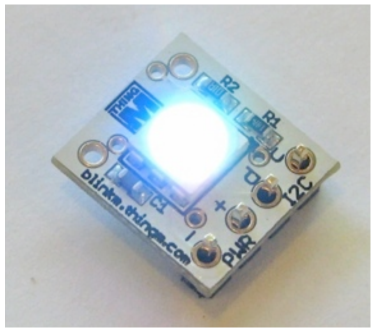


CRASH Space

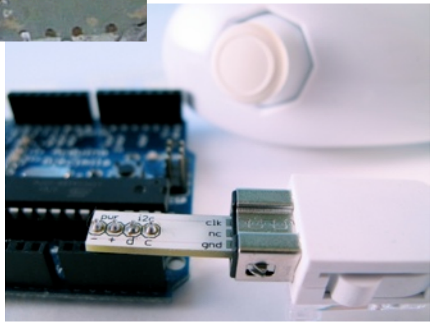


ScrewShield

BlinkM family



Wiichuck
adapter



http://thingm.com/
http://winem.thingm.com/
http://blinkm.thingm.com/
http://crashspace.org/
http://sublimina.com/work/crystal-monster
http://wingshieldindustries.com/
http://hackingroomba.com/
http://todbot.com/blog/spookyarduino/
http://todbot.com/blog/2008/02/18/wiichuck-wii-nunchuck-adapter-available/

My Conception of Ubicomp Objects

They have their own intelligence & network connectivity

They aren’t configured, or wired up, or recharged

Just like normal objects
but on the Net

Finding a cheap, low-power wireless networking solution was key

Wireless Ubicomp

“I want a <$5 network connection for my gadget”

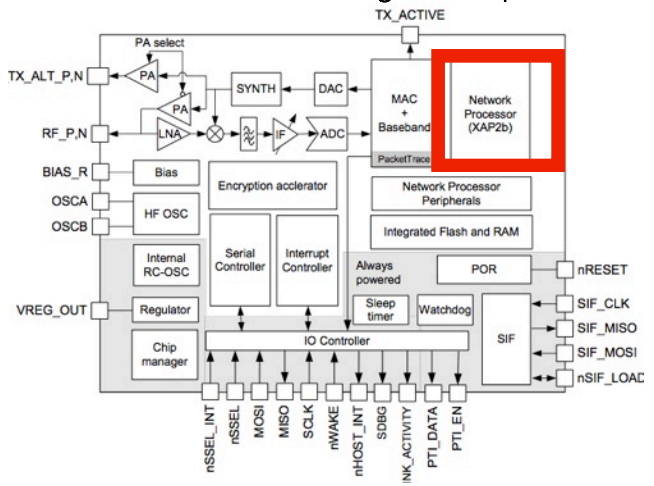
It turns out,

wireless is difficult

wireless is complex

wireless is expensive

Ember EM260 ZigBee chip



Even Bluetooth & Zigbee, still ~$20/node
(And those require gateways to Internet)

It’s difficult: requires real engineering (RF, analog, digital, protocol) to do it yourself, requires certification
It’s complex: look at the Ember EM260 Zigbee chip. Look at everything but the red square. Then the red square? It’s an ARM32 CPU
It’s expensive: basic components not cheap, costs power (WiFi)

Wireless, bah

- Per-object cost is frustrating
- Needs per-object configuration
- Needs special gateway device to Net
- Still need to solve power problem

(Nordic chips get kinda close, tho')

The Nordic nRF chips used in wireless mice & keyboards can have a BOM cost of ~$2 in large quantity. This is getting close, but still would need a WiFi<->Nordic gateway device somewhere.

“Tethered” Ubicomp

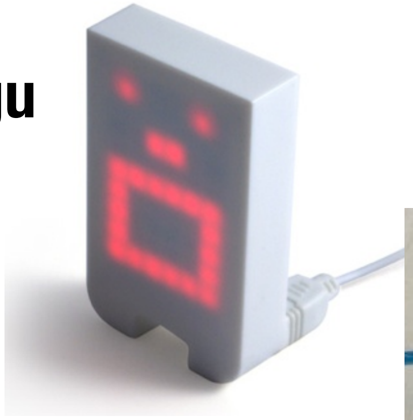
- Devices need two things: network & power
- USB allows offload of connectivity & power hassles to computer
- Need a gateway anyway, make it the computer
- But now have to deal with USB
- USB is hard. Isn’t it?

```
SSID: NearbyCafe
WEP key: 0x32adbbcd
ZigBee PAN ID: 0x00BCD1
12VDC @ 500mA wall wart
```

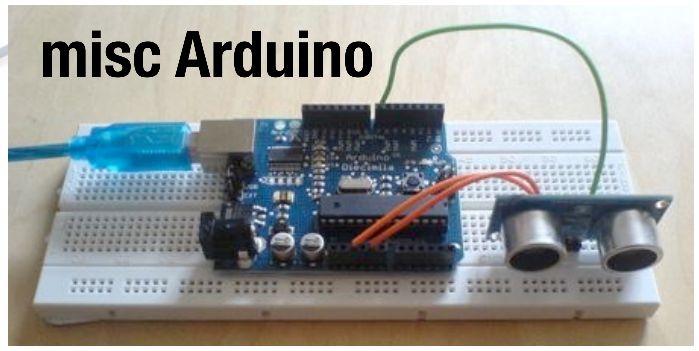
none of this stuff plz

“Tethered” Ubicomp

Tengu

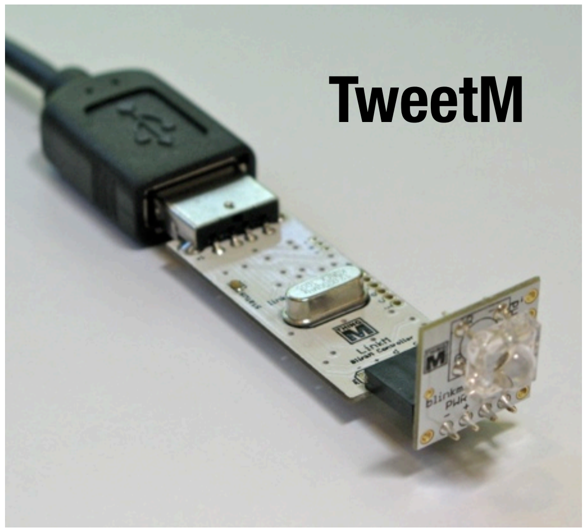


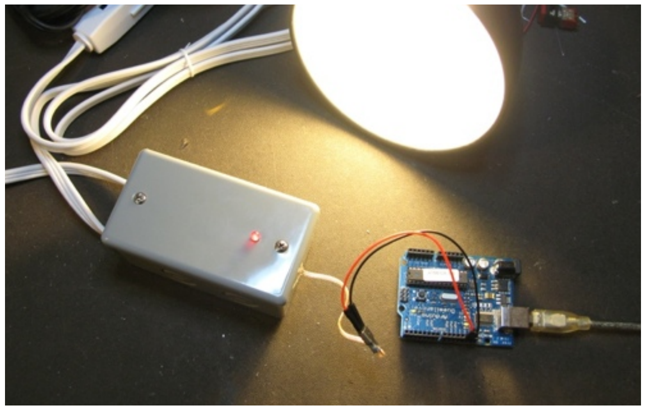
misc Arduino



Snowbot

TweetM





Net-controlled AC lamp

Availabot

Almost every Arduino project that’s serial-commandable is an example of “tethered ubicomp” to me. Also, Phidgets.

Serial vs. HID

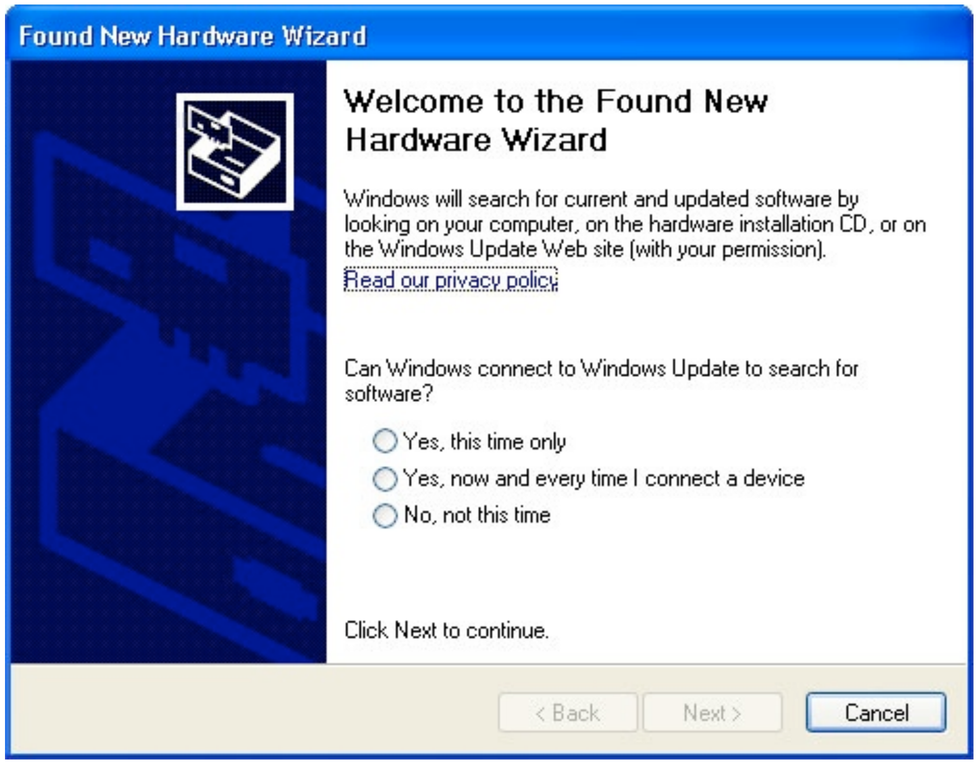
usability from a user’s perspective

Serial-port based device	HID-based device
▪ Install device driver	
▪ (Reboot)	
▪ Plug in device	▪ Plug in Device
▪ Connect	▪ Use Device
▪ Use Device	▪ Unplug device
▪ Disconnect	
▪ Unplug device	

- unplug before disconnect at your peril
- lots of confusion if you do things out of order
- still need to supply power to device

Driver Installs Suck

Reboots? INF files?



Found New Hardware Wizard

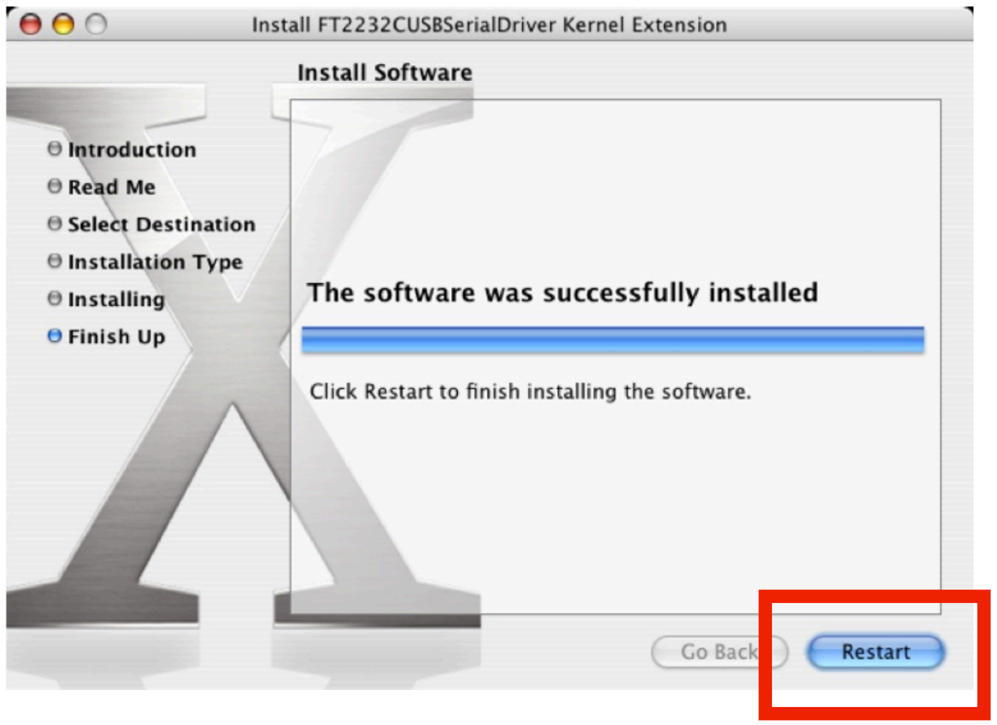
Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).
Read our privacy policy

Can Windows connect to Windows Update to search for software?

- Yes, this time only
- Yes, now and every time I connect a device
- No, not this time

Click Next to continue.



Install FT2232CUSBSerialDriver Kernel Extension

Install Software

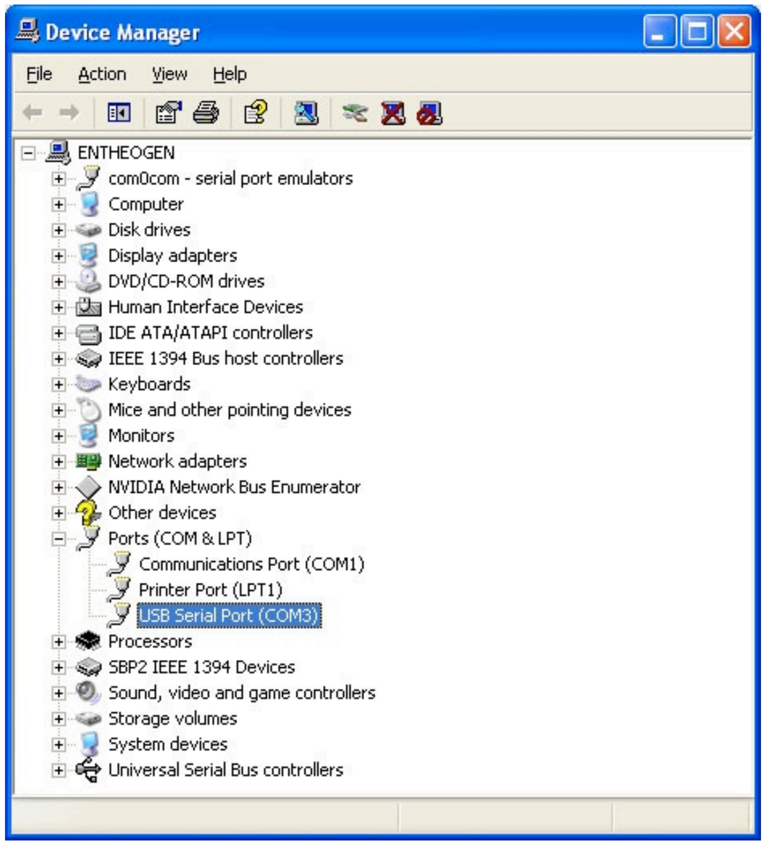
The software was successfully installed

Click Restart to finish installing the software.

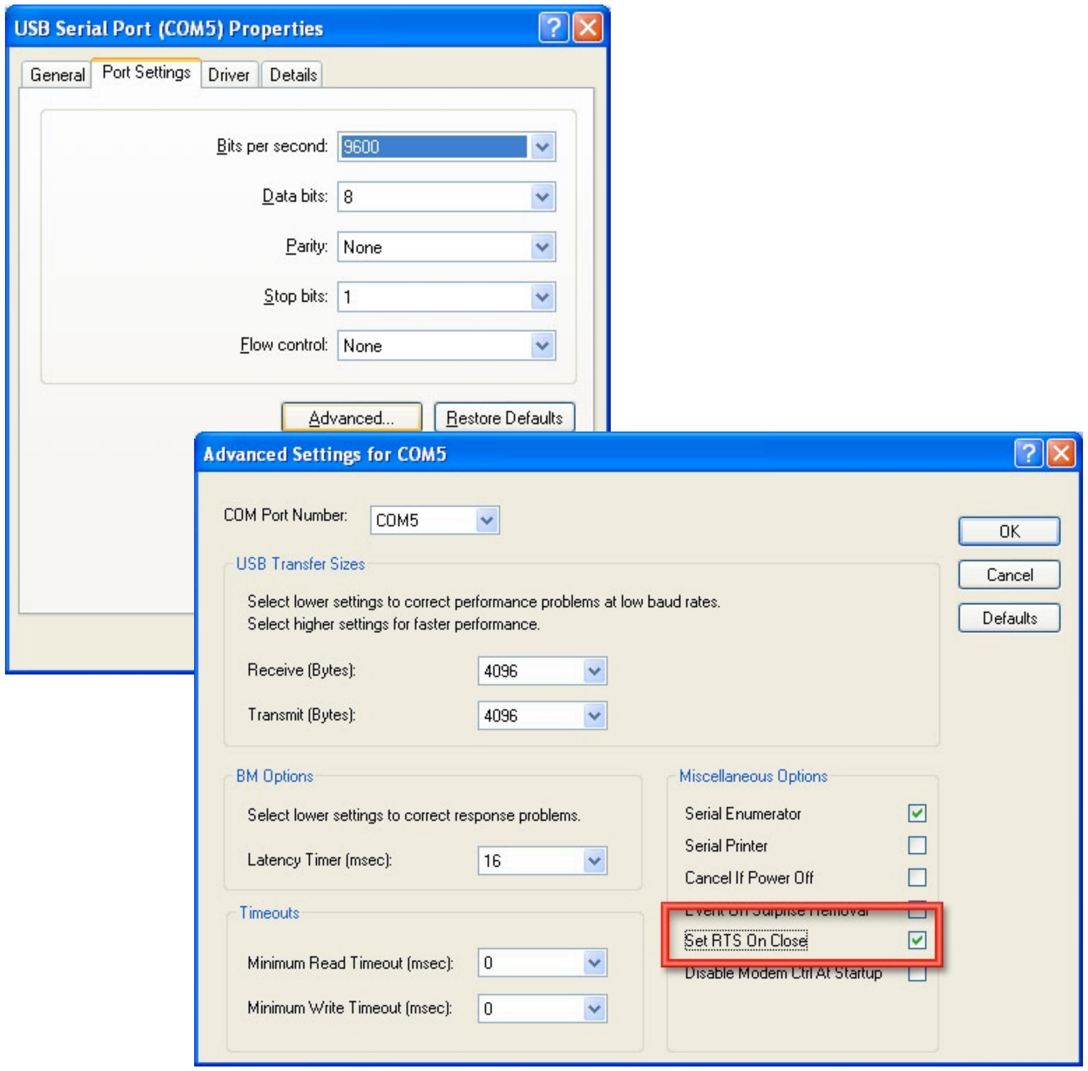
Really? In 2010?

Serial-Oriented Semantics Suck

COM ports? Sigh.



- ENTHEOGEN
  - com0com - serial port emulators
  - Computer
  - Disk drives
  - Display adapters
  - DVD/CD-ROM drives
  - Human Interface Devices
  - IDE ATA/ATAPI controllers
  - IEEE 1394 Bus host controllers
  - Keyboards
  - Mice and other pointing devices
  - Monitors
  - Network adapters
  - NVIDIA Network Bus Enumerator
  - Other devices
  - Ports (COM & LPT)
    - Communications Port (COM1)
    - Printer Port (LPT1)
    - USB Serial Port (COM3)
  - Processors
  - SBP2 IEEE 1394 Devices
  - Sound, video and game controllers
  - Storage volumes
  - System devices
  - Universal Serial Bus controllers



USB Serial Port (COM5) Properties
General Port Settings Driver Details
Advanced... Restore Defaults
Advanced Settings for COM5
OK
Cancel
Defaults

Bits per second: 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None

COM Port Number: COM5

USB Transfer Sizes

Select lower settings to correct performance problems at low baud rates.
Select higher settings for faster performance.

Receive (Bytes): 4096
Transmit (Bytes): 4096

BM Options

Select lower settings to correct response problems.

Latency Timer (msec): 16

Timeouts

Minimum Read Timeout (msec): 0
Minimum Write Timeout (msec): 0

Miscellaneous Options

Serial Enumerator	✓
Serial Printer	
Cancel If Power Off	
Event On Surprise Removal	
Set RTS On Close	✓
Disable Modem Ctrl At Startup	

Which port? What port settings?

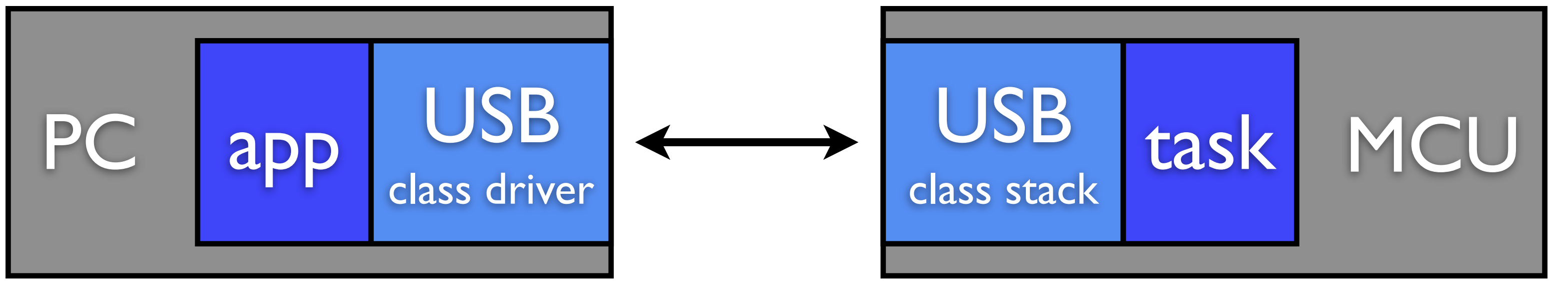
Still need to create data protocol

A Solution: Use USB “Class” Drivers

Pre-defined USB drivers already in your OS

- Mass Storage
- Printer
- Audio
- Video Camera
- CDC
- HID

All these need USB 2.0 (faster, harder to implement), except for HID
All except Storage, Audio, HID kinda wonky, depending on OS



HID not just for Keyboards & Mice

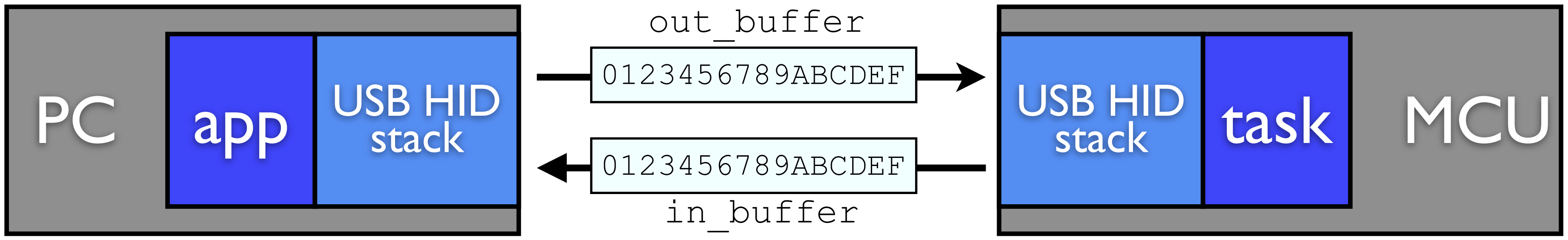
HID devices speak in bi-directional structured data packets

Arbitrary data structures can be defined, called “reports”

Reports can be any size*

Multiple reports can be defined

Let’s say our reports are a single 16-byte buffer, used in both directions



* up to 512 bytes realistically

Could define multiple reports for different purposes
HID bootloaders do that: one for data payload, another for status msgs

Host-side API for Generic Data thru HID

```
-open()
-command( in_buffer, out_buffer )
-close()
```

"open()"
- scans bus or opens specific device by VendorID/ProductID
- doesn't have to open exclusively

"command()"
- sends a buffer of data to device
- optionally receives a buffer of data from device

Host-side Examples

Sending data to device:

```
// set color to white
out_buffer = {‘c’, 0xff, 0xff, 0xff};
command( out_buffer, null );
```

Reading data from device:

```
// get I2C address
out_buffer = {‘a’};
command( out_buffer, in_buffer );
println( in_buffer );
```

Embrace Connectionlessness

User can pull device at any time

```
// I don’t care if this succeeds
command( out_buffer, null );

// I want to know what’s going on
try {
  command( out_buffer, null );
} catch(IOException ioe) {
  devicePresent = false;
}

// periodically, do
if( !devicePresent ) {
  lookForDevice();
}
```

Think UDP not TCP

Have a separate thread or otherwise periodically check for device insertion.

Device-side API for Generic Data thru HID

One function:

```
handleMessage()
```

Global buffers:

```
in_msg_buff
out_msg_buff
```

Doing a transaction:

```
void handleMessage() {
  cmd = in_msg_buff[0];
  if( cmd == ‘v’ )   // version
    out_msg_buff = {0x13, 0x37};
}
```

This is a simplification of what’s going on in the LinkM firmware, but can apply to the general case, even across chip types like to a PIC 18F4550 USB.

Platforms?

Host code works on “all” platforms

On Mac OS X & Linux: libusb
On Windows: hidsdi.h/hidpi.h

Abstraction layer in C for same codebase on all three

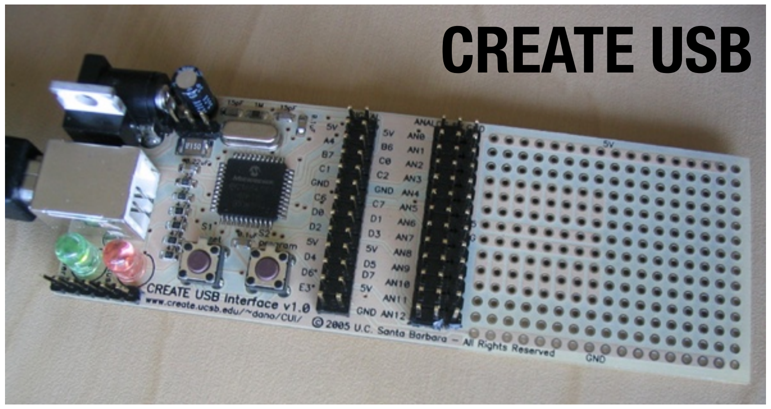
Abstraction layer written by Obdev.at (makers of V-USB for AVR), packaged lightly by ThingM

Java wrapper library & Processing library by ThingM

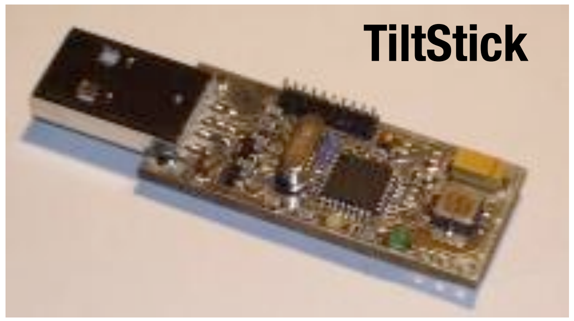
Python library upcoming, hopefully

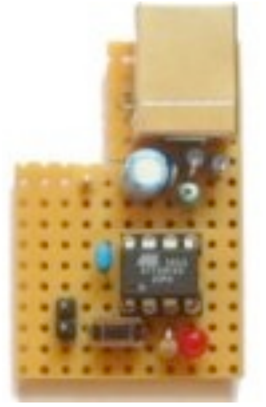
Check out the “hiddata” example project in V-USB for the basis of what was used.

Others doing USB HID



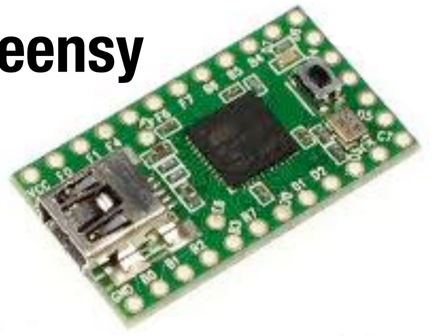
Microchip USB: PIC18F4550




EasyLogger

V-USB: ATmega & ATtiny

Teensy



Atmel USB/LUFA: ATmegaxUx

http://microchip.com/usb
http://www.obdev.at/products/vusb/
http://www.fourwalledcubicle.com/LUFA.php
http://www.atmel.com/products/AVR/usb.asp

http://www.create.ucsb.edu/~dano/CUI/
http://www.pjrc.com/teensy/

Downsides to HID

Well, it's tethered.

- 5 meters (16') cable length by spec
- But you get power on that cable (up to 500mA)

Also, slow. (compared to USB 2.0)

- full-speed HID: 640 kbps (64 bytes / 1msec frame)
- low-speed HID: 8 kbps (8 bytes / 10msec frame)
- But that's good enough for most cases

Bureaucratic Hassles of USB

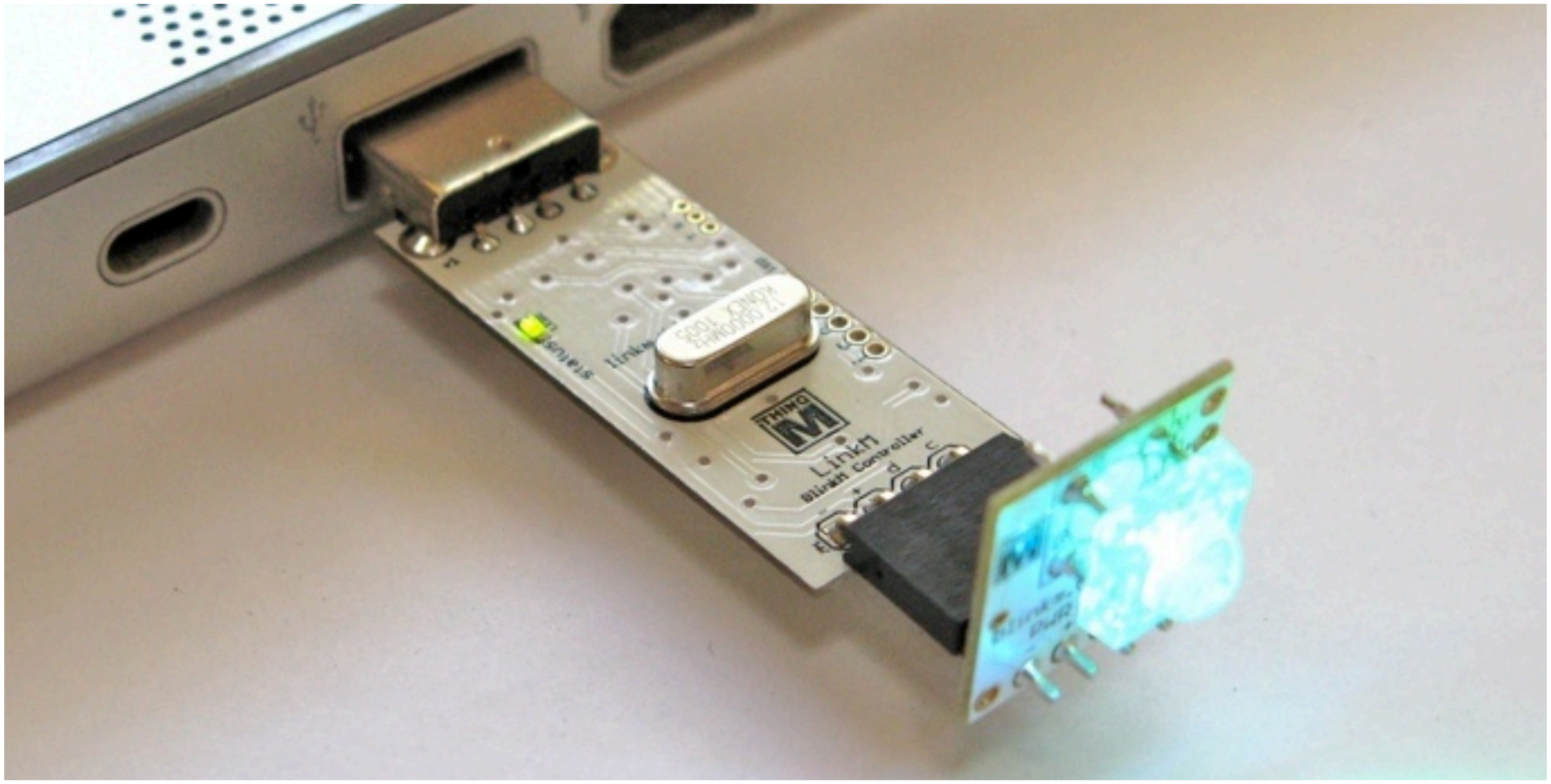
Need USB-IF compliance testing for USB logo use
- Need to be USB-IF member ($4k/yr)
- So, don’t use logo

Need unique VendorID & ProductID for each gadget
- VendorID: $2k/2yr from USB-IF (non-member)
- V-USB: free PIDs for OSS use
- V-USB: two VID/PIDs for $500 prof. license
- Microchip: free PID under their VID
- Atmel: use their VID/PID, no PID of your own

=> Can avoid most hassles fairly cheaply

LinkM

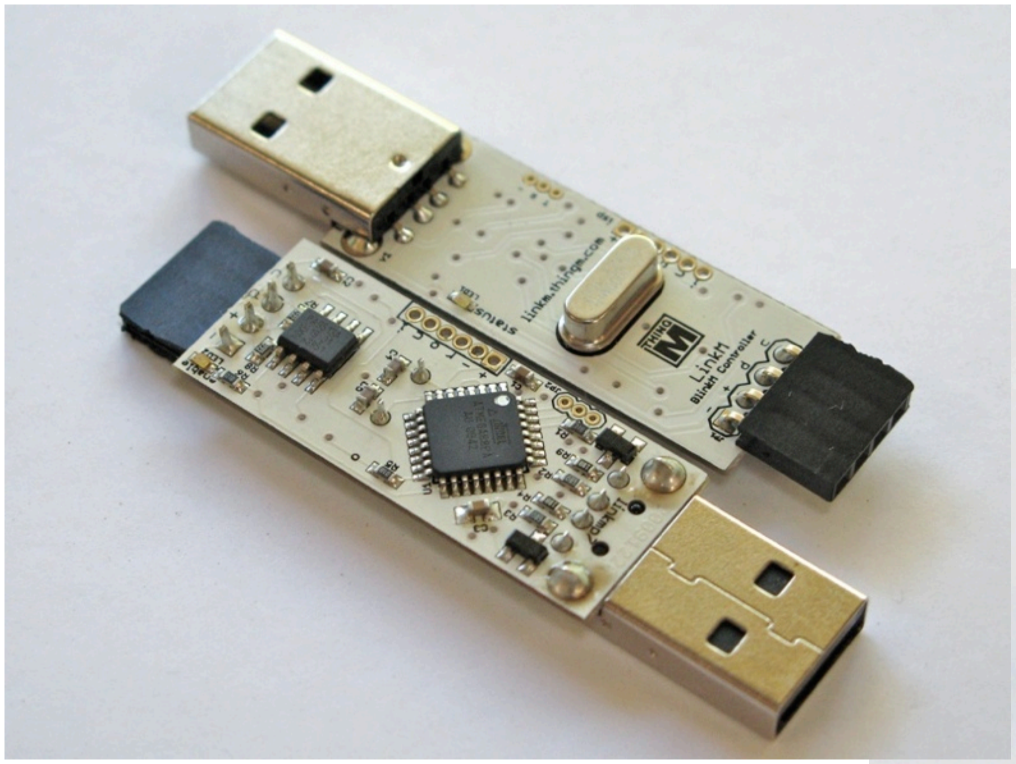
low-cost USB-to-I2C adapter



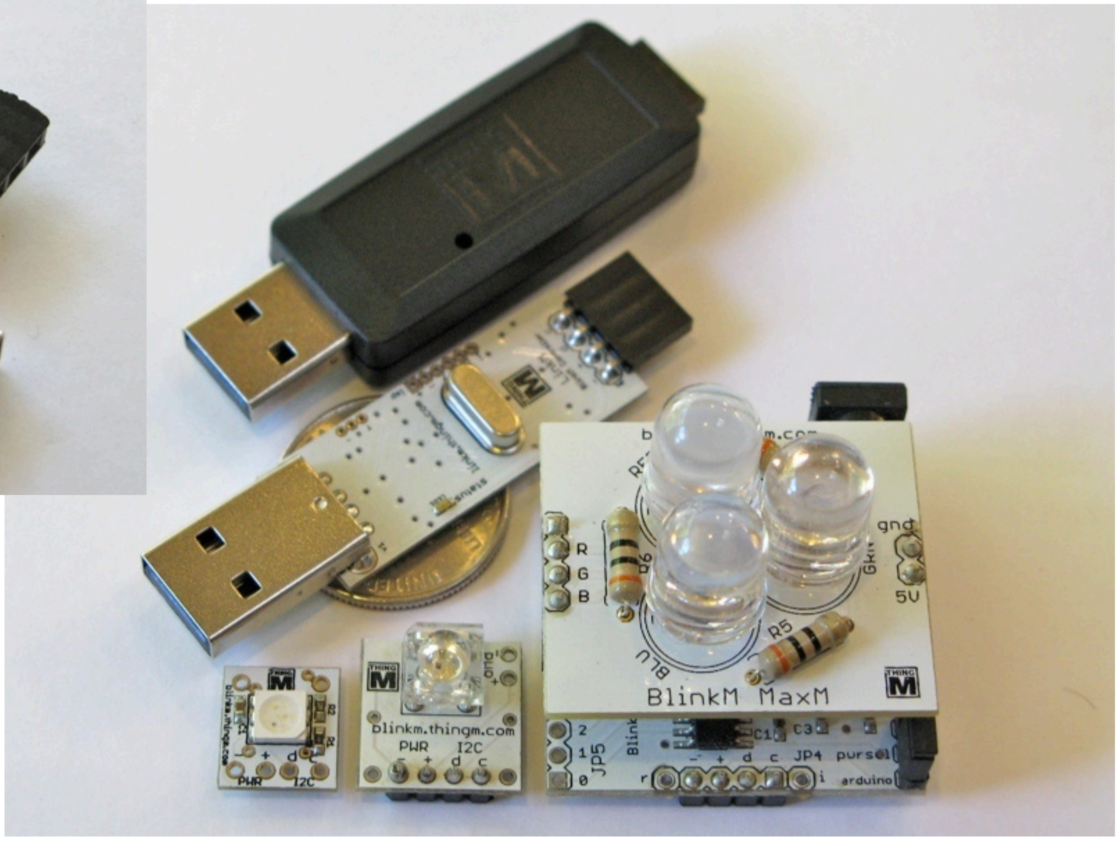
LinkM can directly power from USB 8 regular BlinkMs or one BlinkM MaxM.

LinkM

based on ATmega88P, using V-USB software USB stack

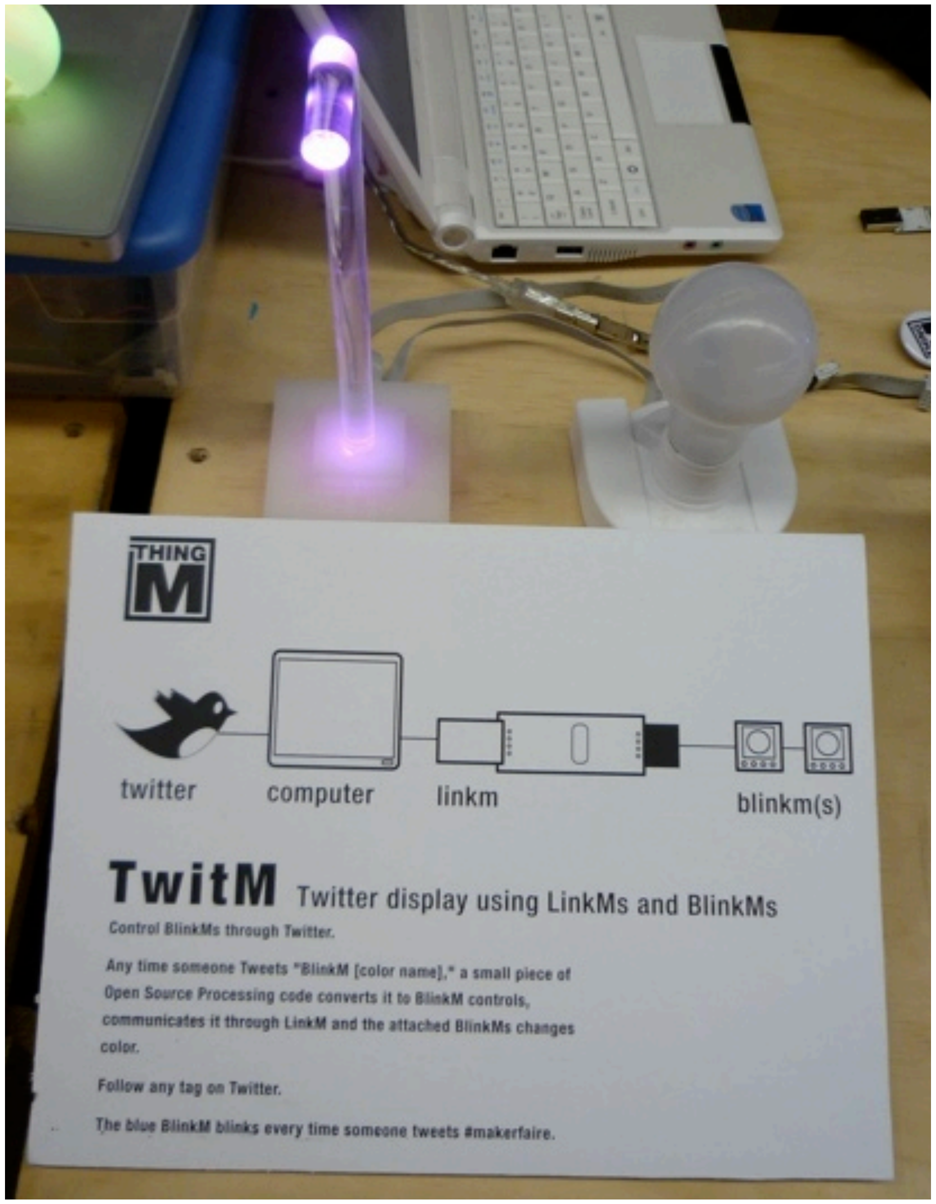


100% open source
Two chip solution,
including I2C line driver



Actually contains two
independent USB HID
stacks: task & bootloader

TweetM



TwitM Twitter display using LinkMs and BlinkMs

Control BlinkMs through Twitter.

Any time someone Tweets "BlinkM [color name]," a small piece of Open Source Processing code converts it to BlinkM controls, communicates it through LinkM and the attached BlinkMs changes color.

Follow any tag on Twitter.

The blue BlinkM blinks every time someone tweets #makerfaire.

Control an LED via Twitter

LinkM + BlinkM
Gateway in Processing
Multiple BlinkMs addressable

Called “TwitM” at Maker Faire. Had two BlinkMs in that case, one responded to any tweets with “makerfaire”, the other was commandable with the command language “blinkm <colorname>”

LinkM Next Steps

Better unified build environment for host-side code
- Compiling for 5 platforms: Win32, Linux, Mac OS X (i386/x86_64/ppc)

Test with other I2C devices
- Wii nunchuck
- Capacitive touch sensor

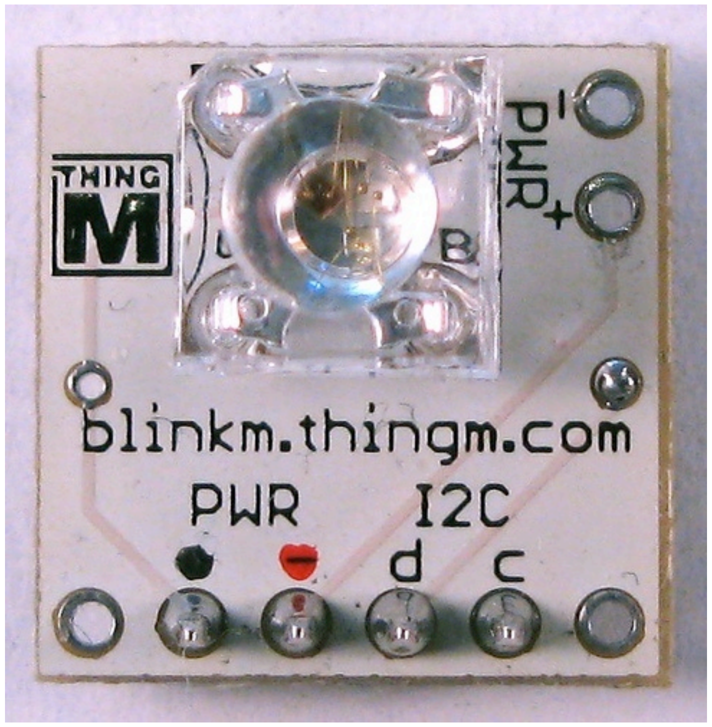
Would like to move to Microchip PIC18F450
- Better chip availability, better USB stack
- But Windows-only dev environment, yuk

LinkM Links

http://linkm.thingm.com/

http://linkm.googlecode.com/

Some TBD link for a generalization of the USB-HID tricks



Tod E. Kurt
http://thingm.com/
http://todbot.com/blog/

Sketching2010
24 July 2010