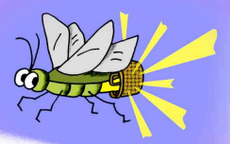
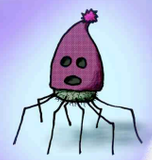
# Debugging

Press Play: Interactive Device Design | July 19, 2011

# DEBUGGING RULES!





Understand the system

Make it fail

Quit thinking and look

Divide and conquer

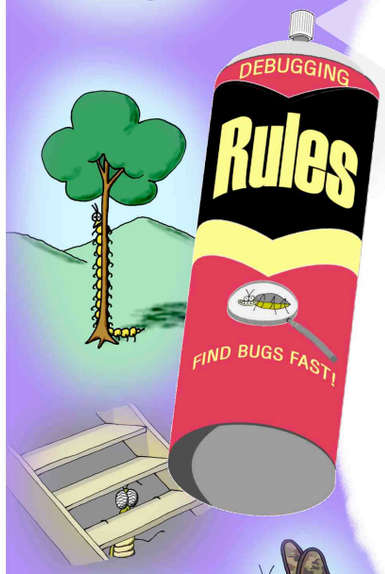
Change one thing at a time

Keep an audit trail

Check the plug

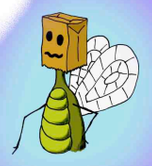
Get a fresh view

If you didn't fix it, it ain't fixed



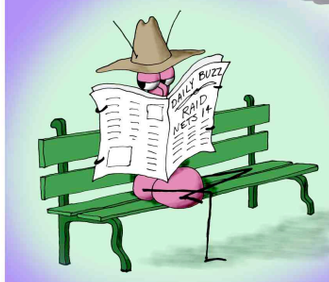


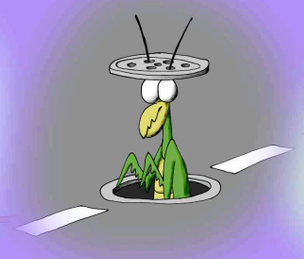






*To get the book or download this free poster, go to*
www.debuggingrules.com



# Debugging

## Advice & Philosophy

*Image from flickr: reuben*

- Debugging is an inevitable part of the design process.
- It ALWAYS takes a disproportionate amount of time.
- Plan accordingly.

# Debugging a Problem

## Check Power & Ground

- Don't assume that you connected them right earlier.
- Use your multimeter or oscilloscope.

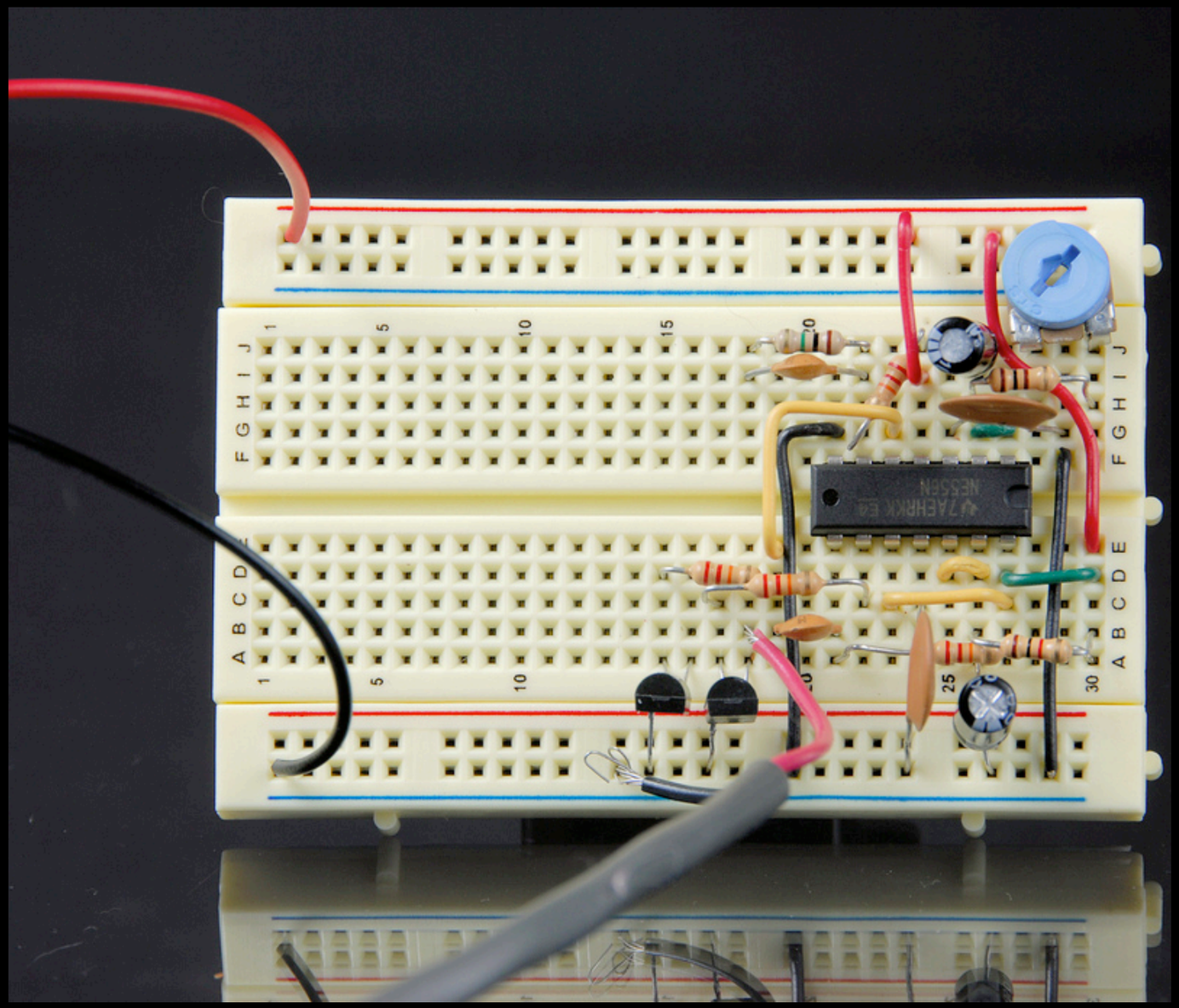


*Image from flickr: leprechaun947*

# Debugging a Problem

## Do a Quick Route-Trace

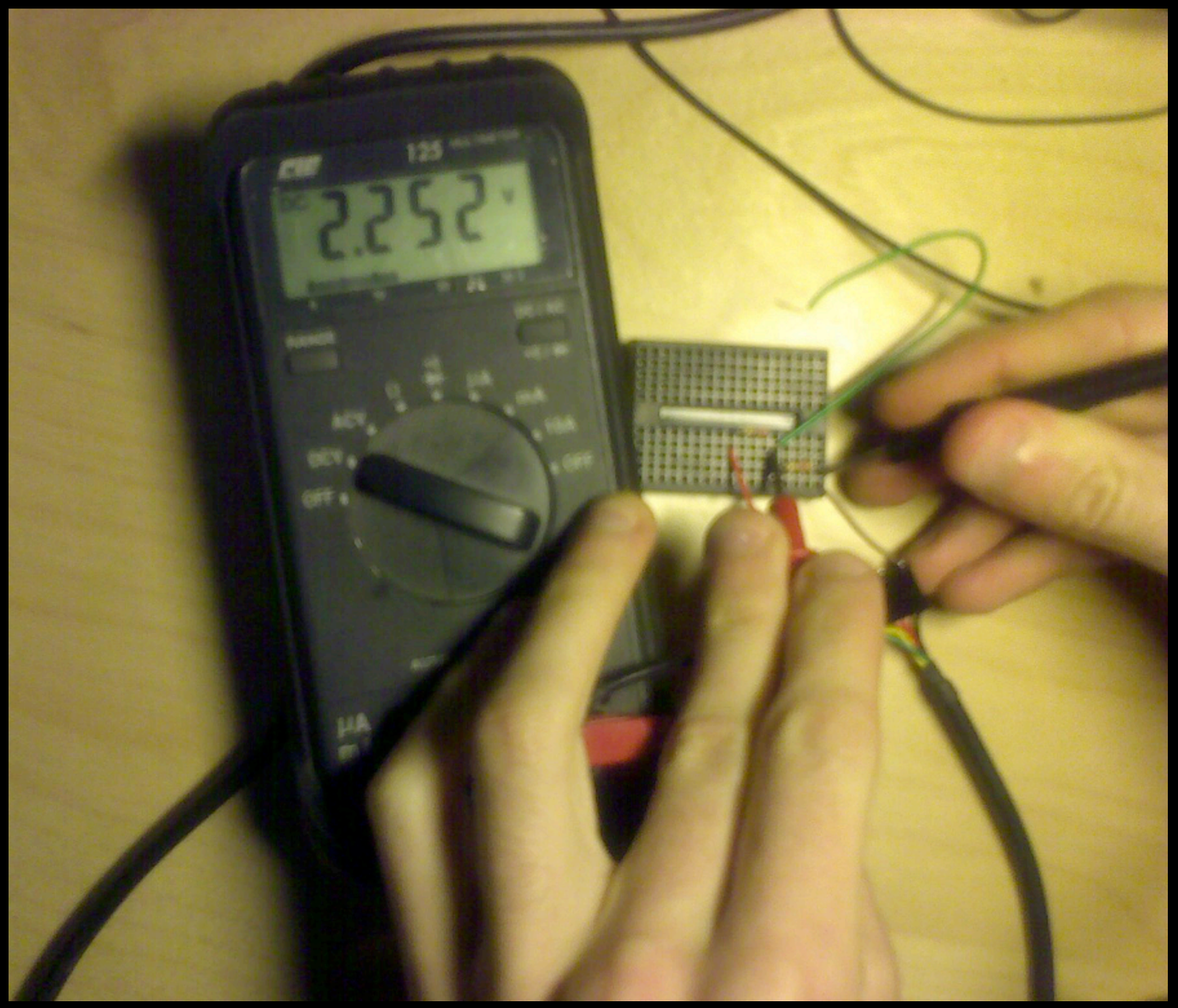



*Image from flickr: deadhacker*

- Make sure that the voltages along each path make sense.
- Double check pin-outs and other such problems against the datasheets.

# Debugging a Problem

## Divide & Conquer

- Can you establish whether the problem is occurring in software or hardware?
- In the first half of the circuit or the second?
- In one particular function or another?

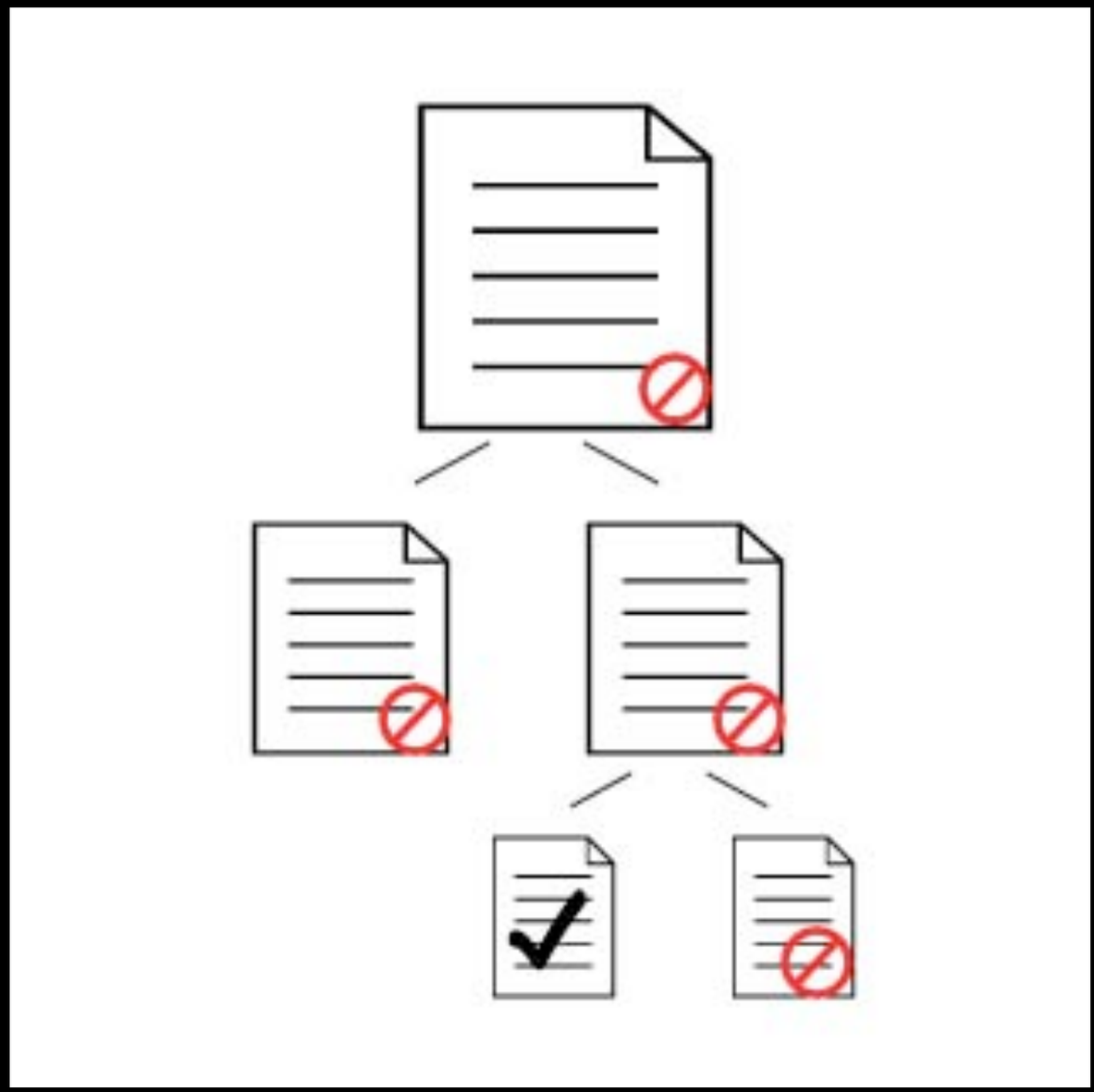


*Image from technochakra.com*

# Debugging a Problem

## Get a Fresh Perspective

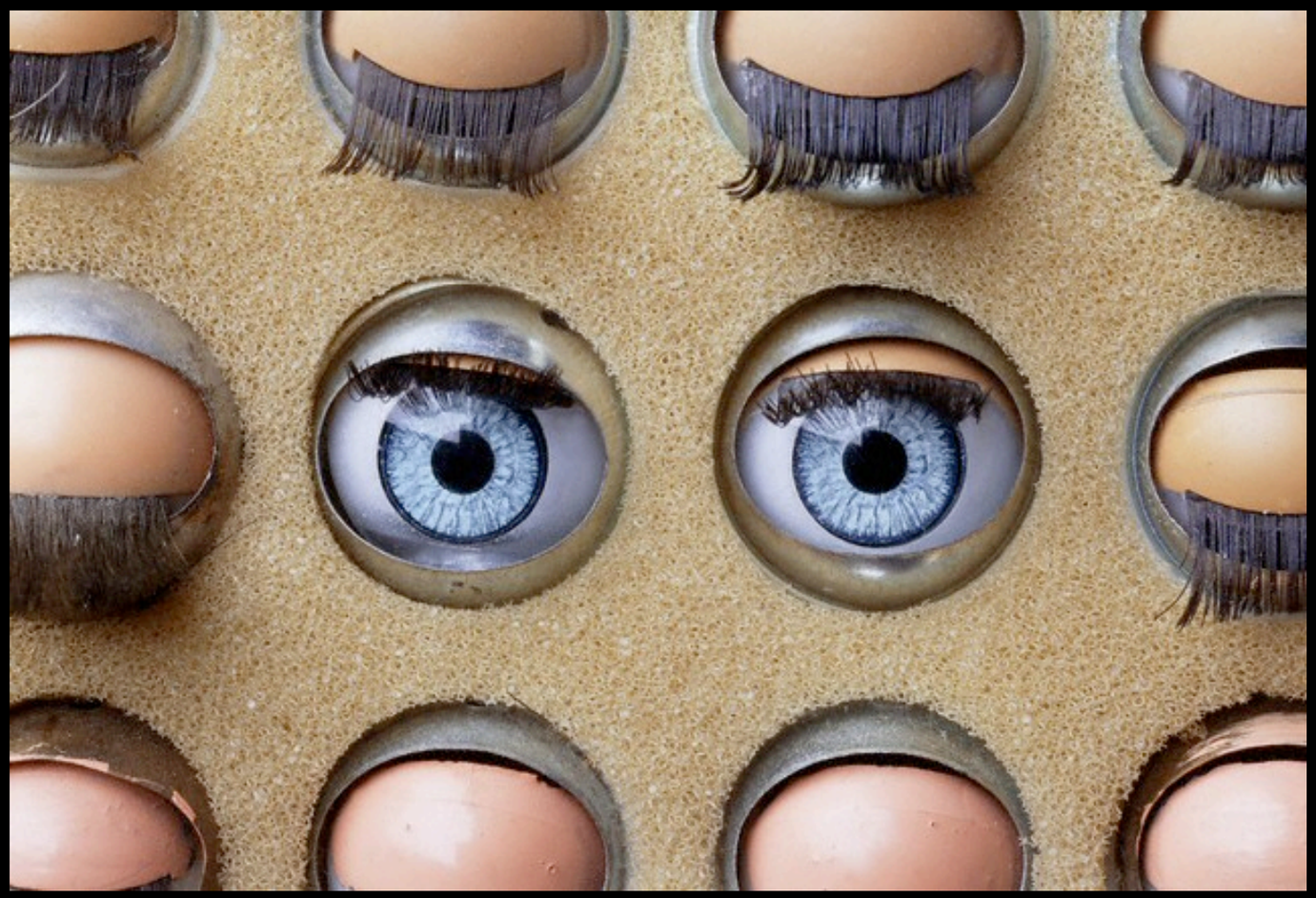


*Image from codinghorror.typepad.com*

- Get up and take a walk.
- Have someone else look at the problem with you.
- Work on another part of the problem for a while.
- Look online and see if anyone else has had the same problem!

# Design for Debug

## Yes, You Can Do This!

- One of the secrets of debugging is not to write too many bugs in the first place!
- Here's some tips for how.


*Image from moderndesignblog.com*

# Design for Debug

## Actually Design Your System

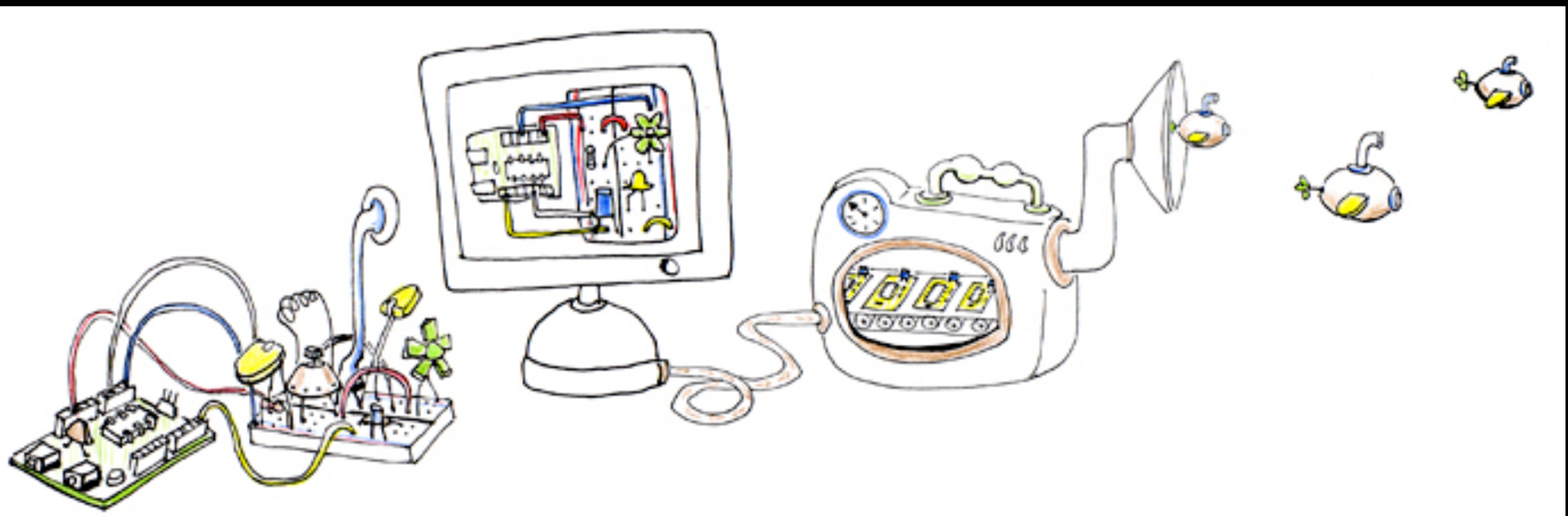


*Image from fritzing.org*

If you just throw stuff together, it’ll be a miracle if it really works.

# Design for Debug

## Actually Design Your System

- Take the time to draw sketches and schematics, both for hardware and software.
- Move from high-level to low-level in your design.
- Write pseudo-code first!

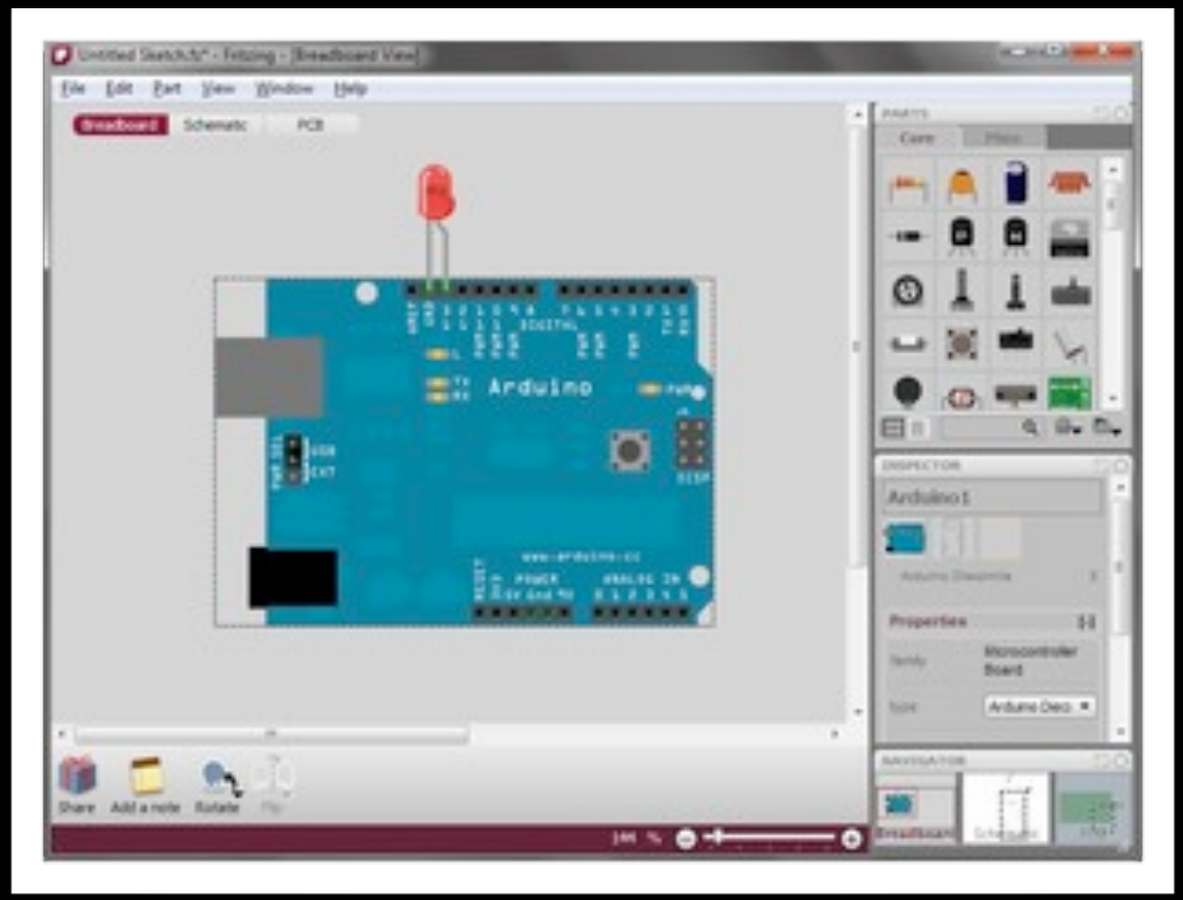



*Image from fritzing.org*

# Design for Debug

## Make One Change at a Time

- ...And make sure it works!
  Keep your tests around.
- In computer science, this is known as unit testing.
- This makes it easier to revert to a “known good” system and to divide-and-conquer later.


*Image from sebastian bergmann*

# Design for Debug

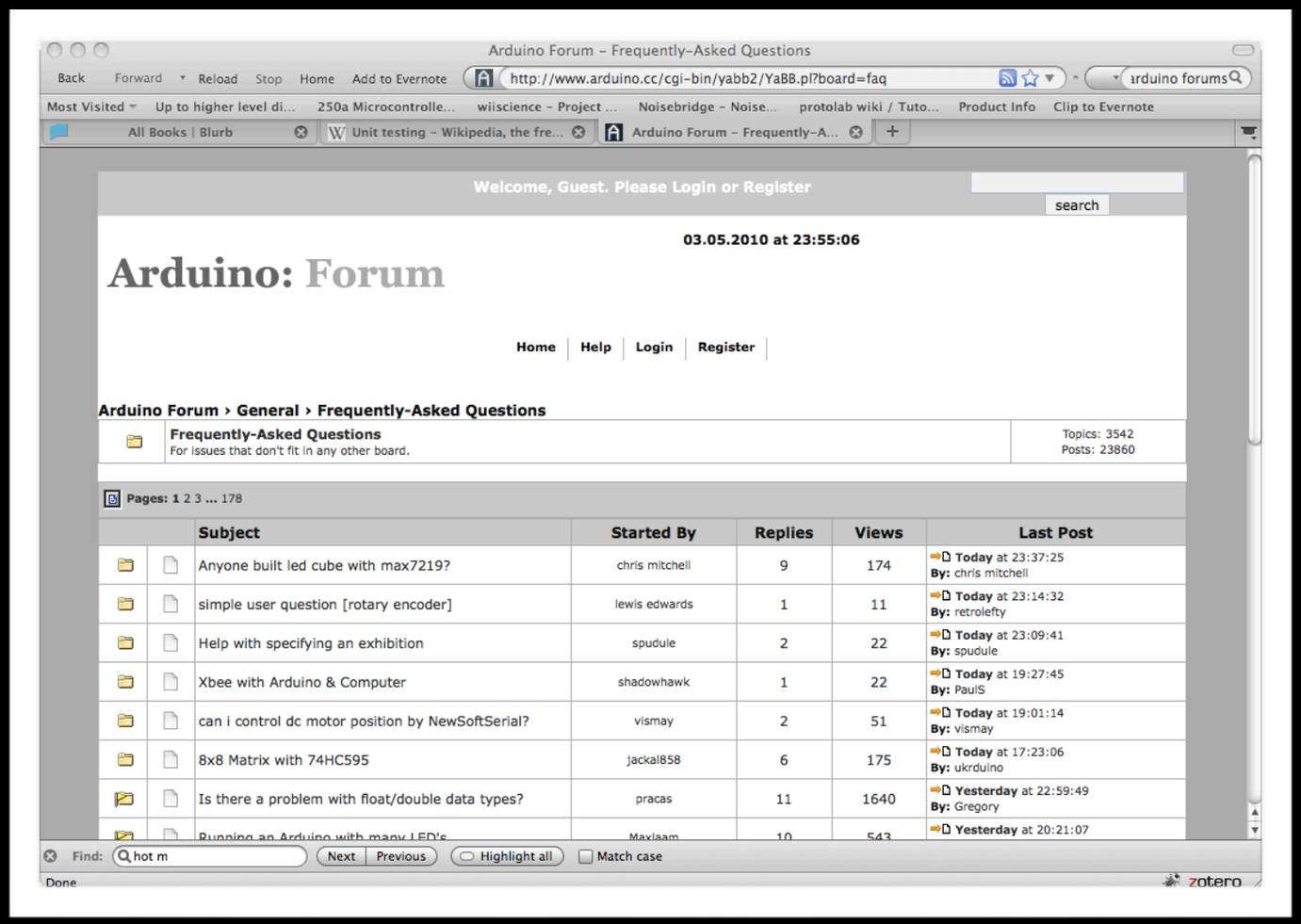
## The Early Bird Gets the Bug

*Image from www.alleba.com/blog/*

- Everyone cuts corners and has difficulty seeing clearly when the dead-line approaches.
- Starting early gives you time to work in a calmer and cleaner manner.

# Design for Debug

## Work Within a Broader Community




- Picking hardware and platforms which are common (and better, open-source) gives you more resources when you do hit the wall.

*Image from www.allega.com/blog/*

# Open Source Hardware

## Resources


Foo & Bar Camps

Ask an Engineer

# Open Source Hardware

## Million Dollar Baby


http://blog.makezine.com/archive/2010/05/million-dollar-baby-businesses-de.html

# In-Class Debug Exercise

What's Wrong with the Circuit and/or Program?

# Homework for Next Week

Deep Dive: Preliminary MP3 Player Designs

Parts List, Interface & Software Design