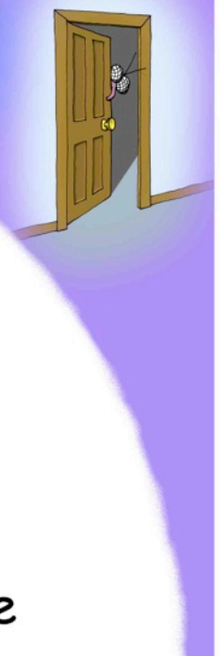
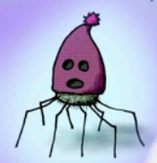
Debugging

Press Play: Interactive Device Design | Aug 01, 2012

DEBUGGING RULES!





Understand the system

Make it fail

Quit thinking and look

Divide and conquer

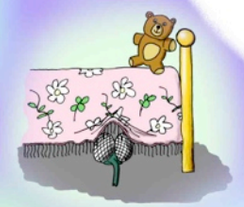
Change one thing at a time

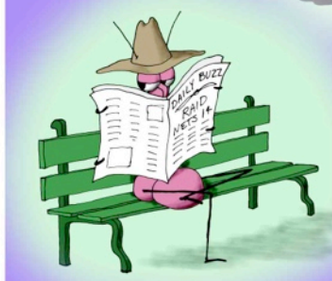
Keep an audit trail

Check the plug

Get a fresh view

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





from Debugging © 2002 by David Agans

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.

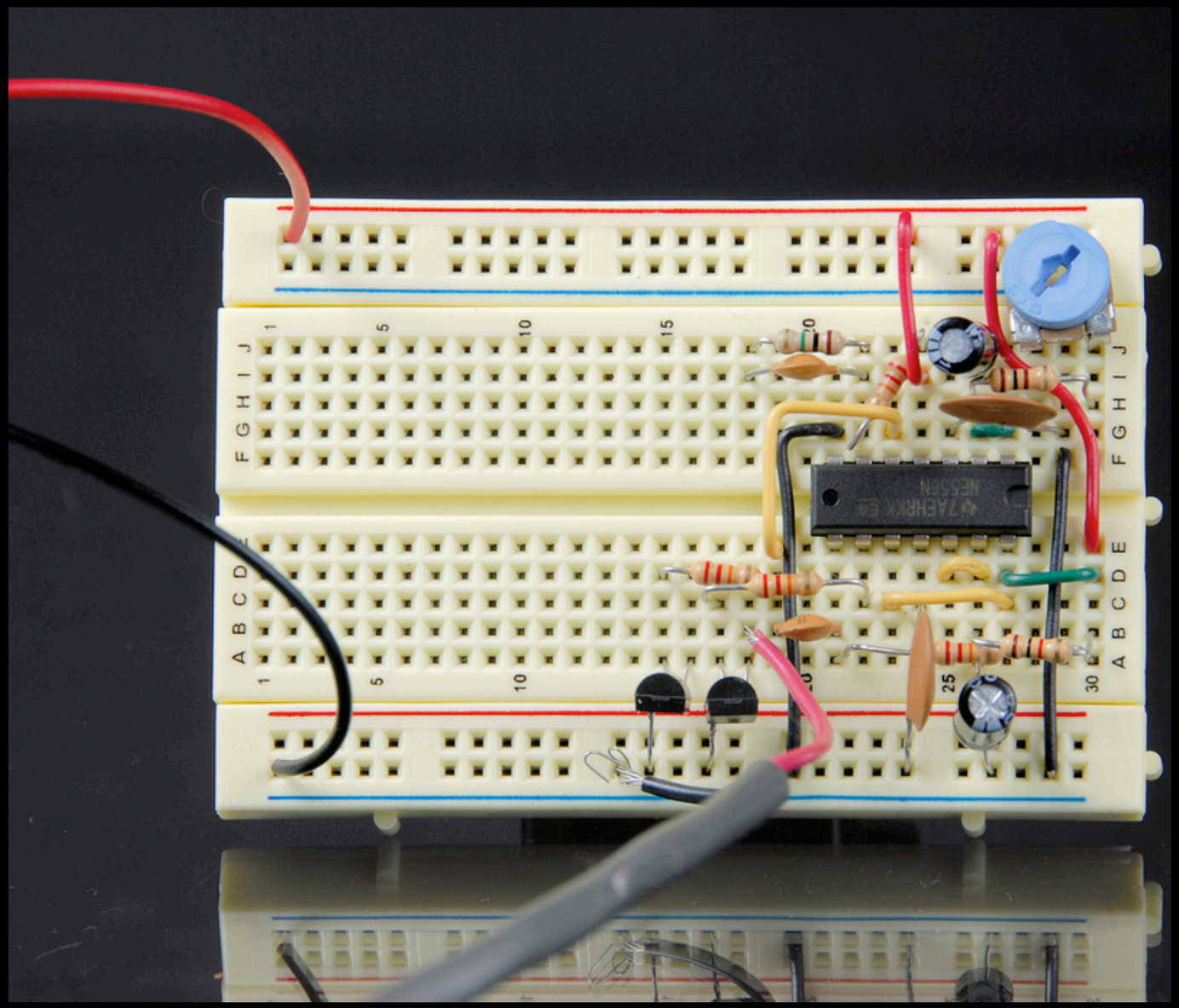


Image from flickr: leprechaun947

Debugging a Problem

Do a Quick Route-Trace

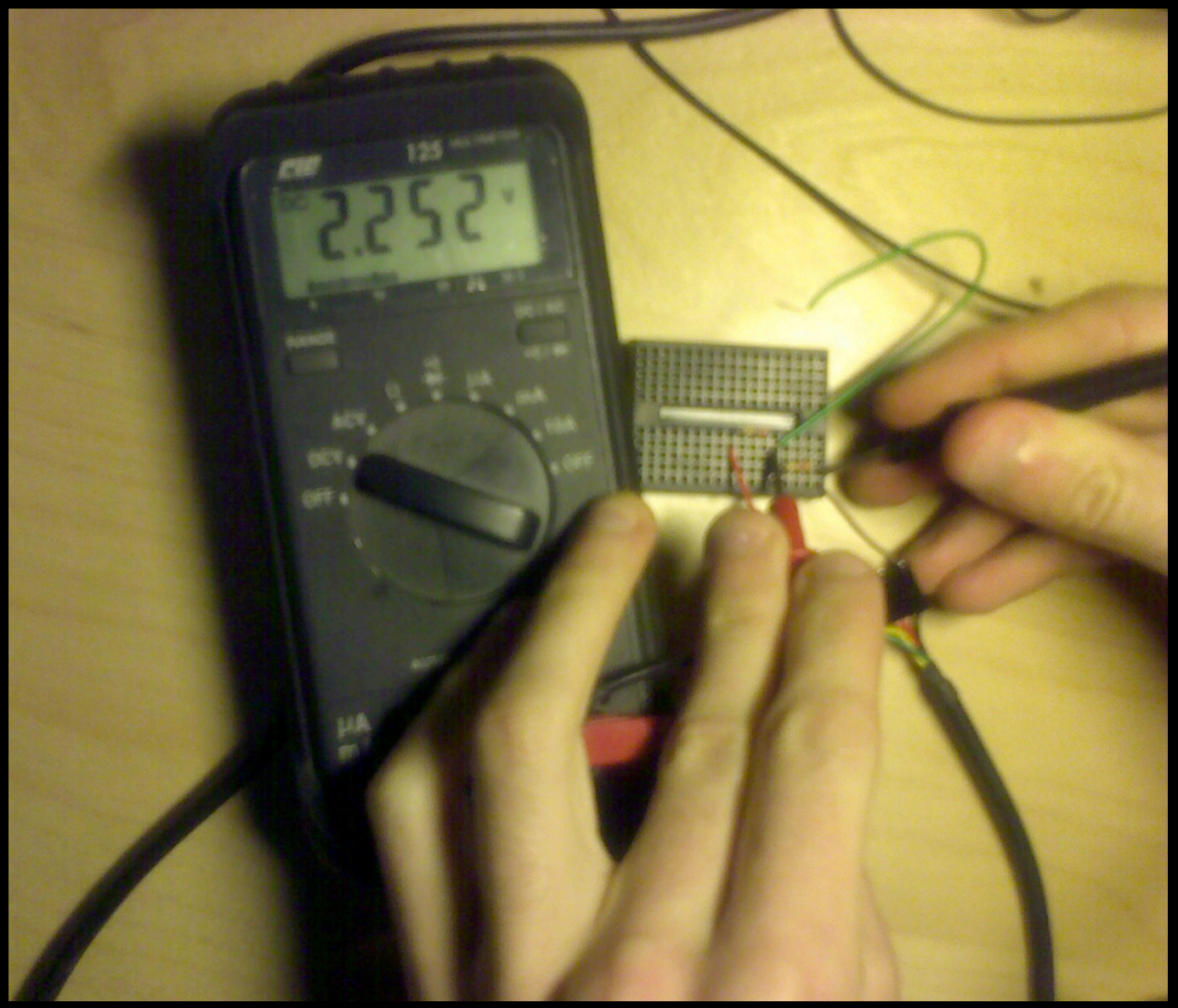


Image from flickr: deadhacker

- 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?

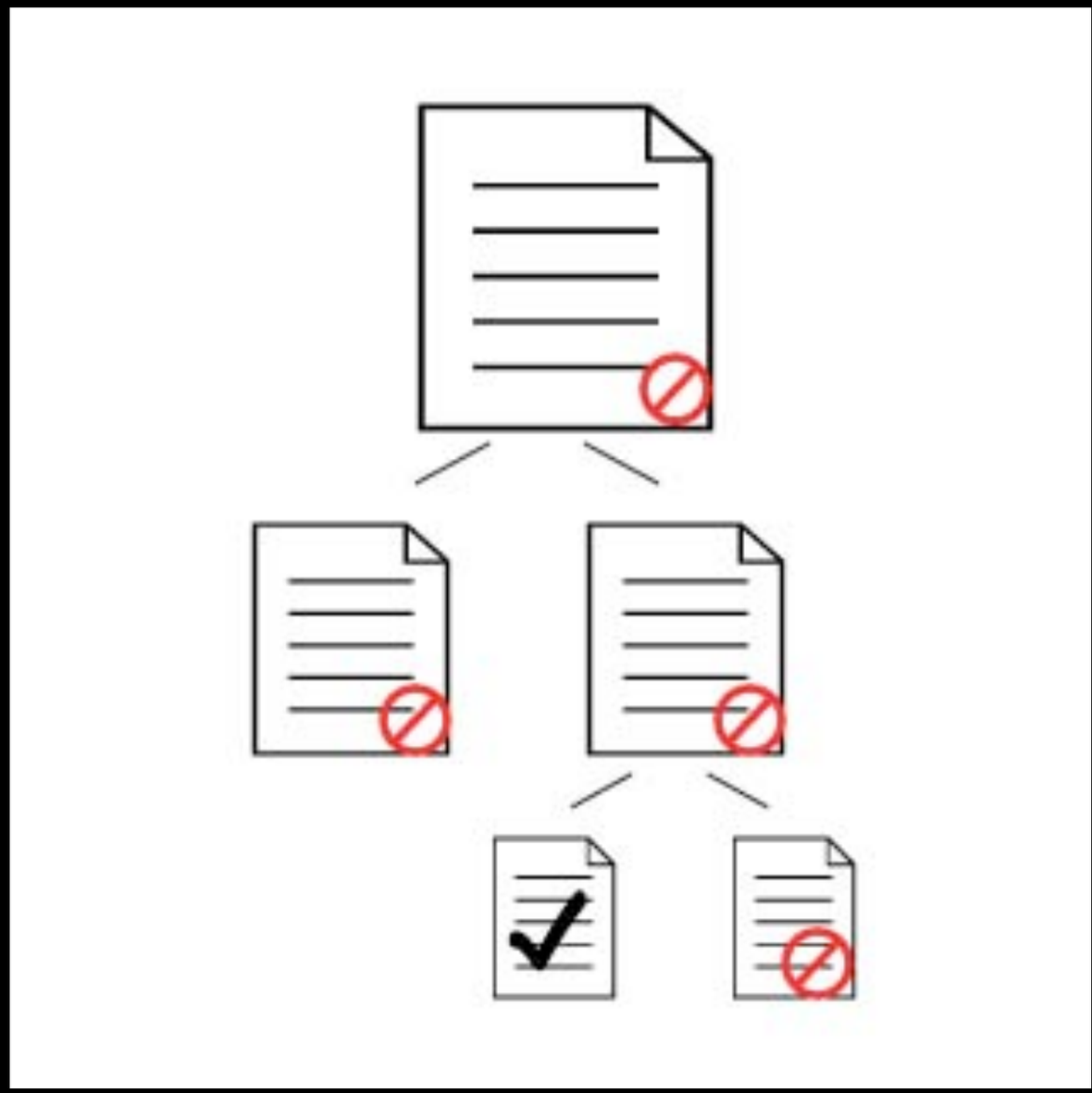


Image from technochakra.com

Debugging a Problem

Get a Fresh Perspective

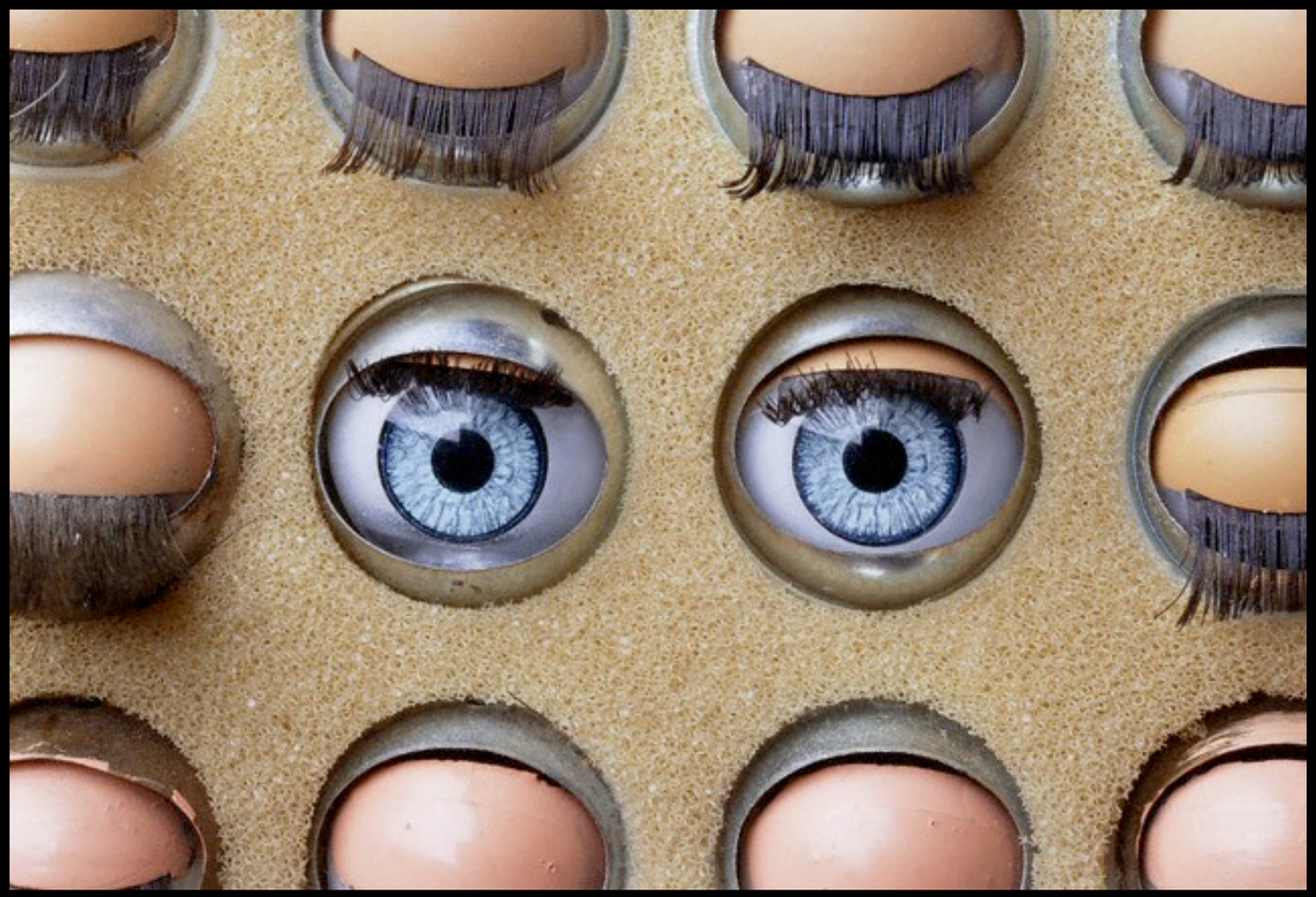


Image from codinghorror.typepad.com

- ☐ 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

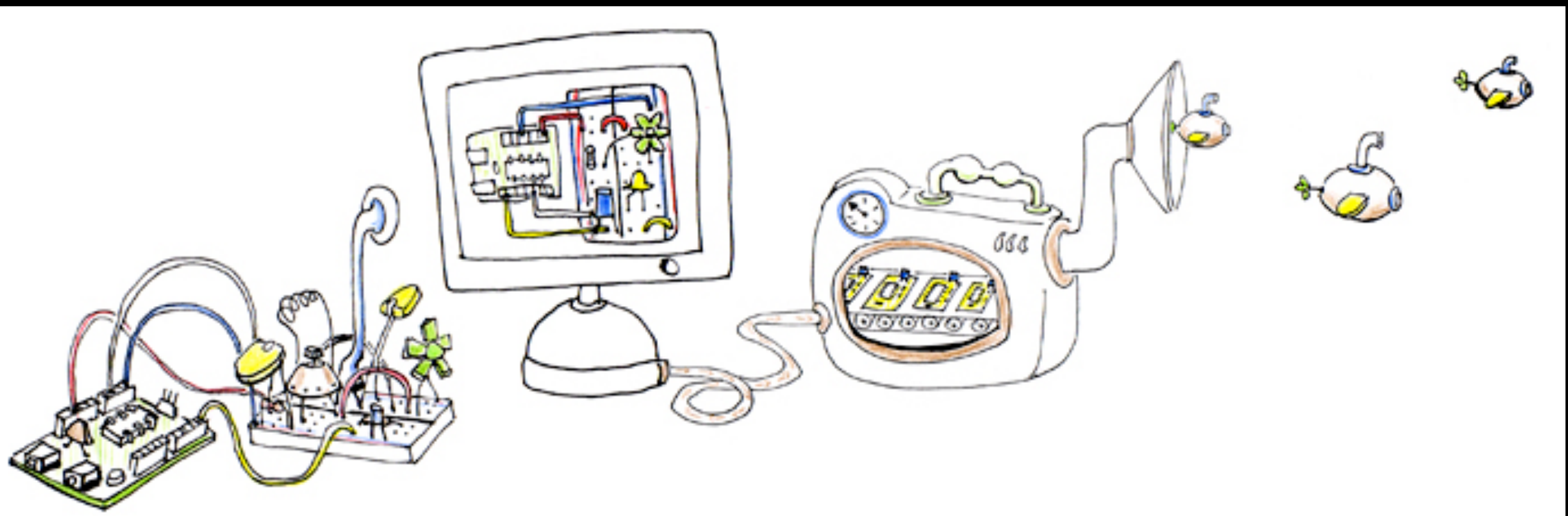


Image from fritzing.org

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!

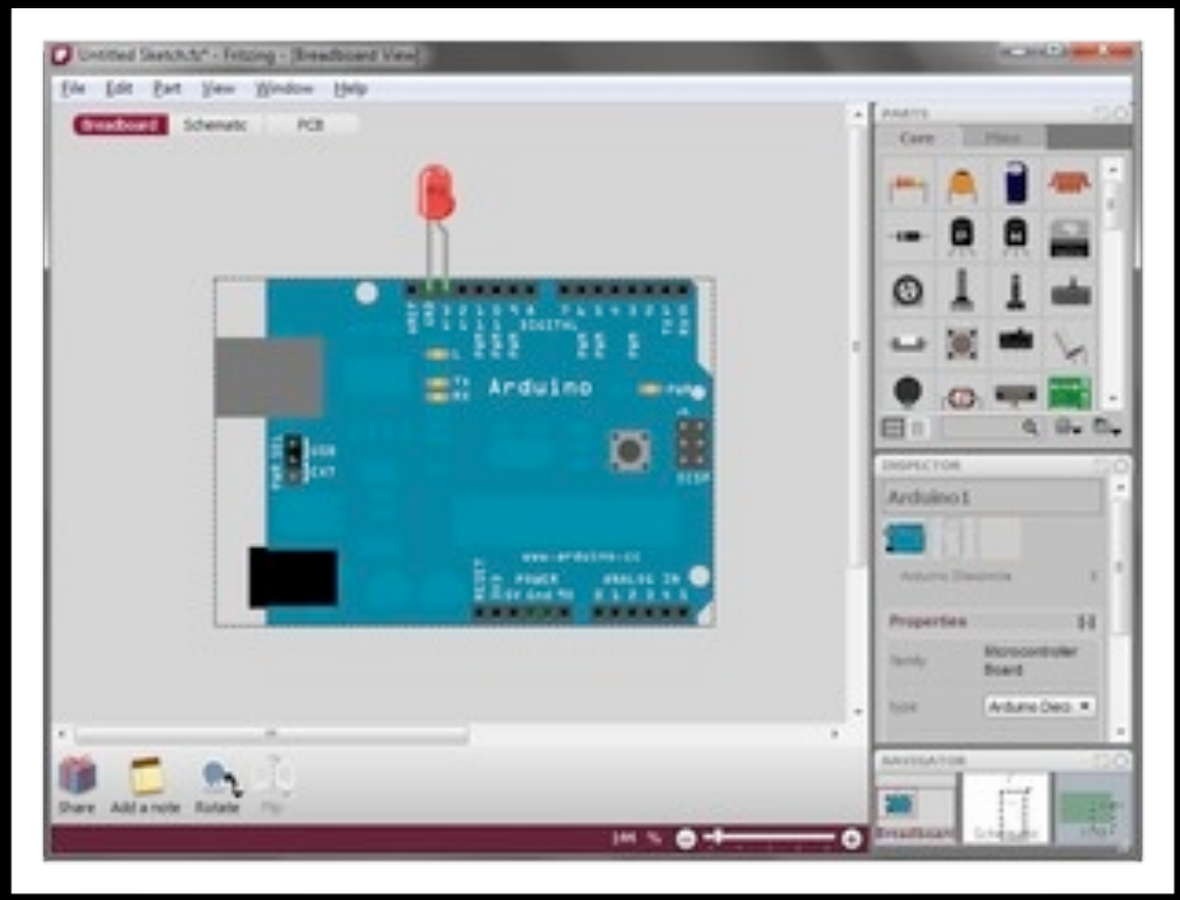


Image from fritzing.org

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

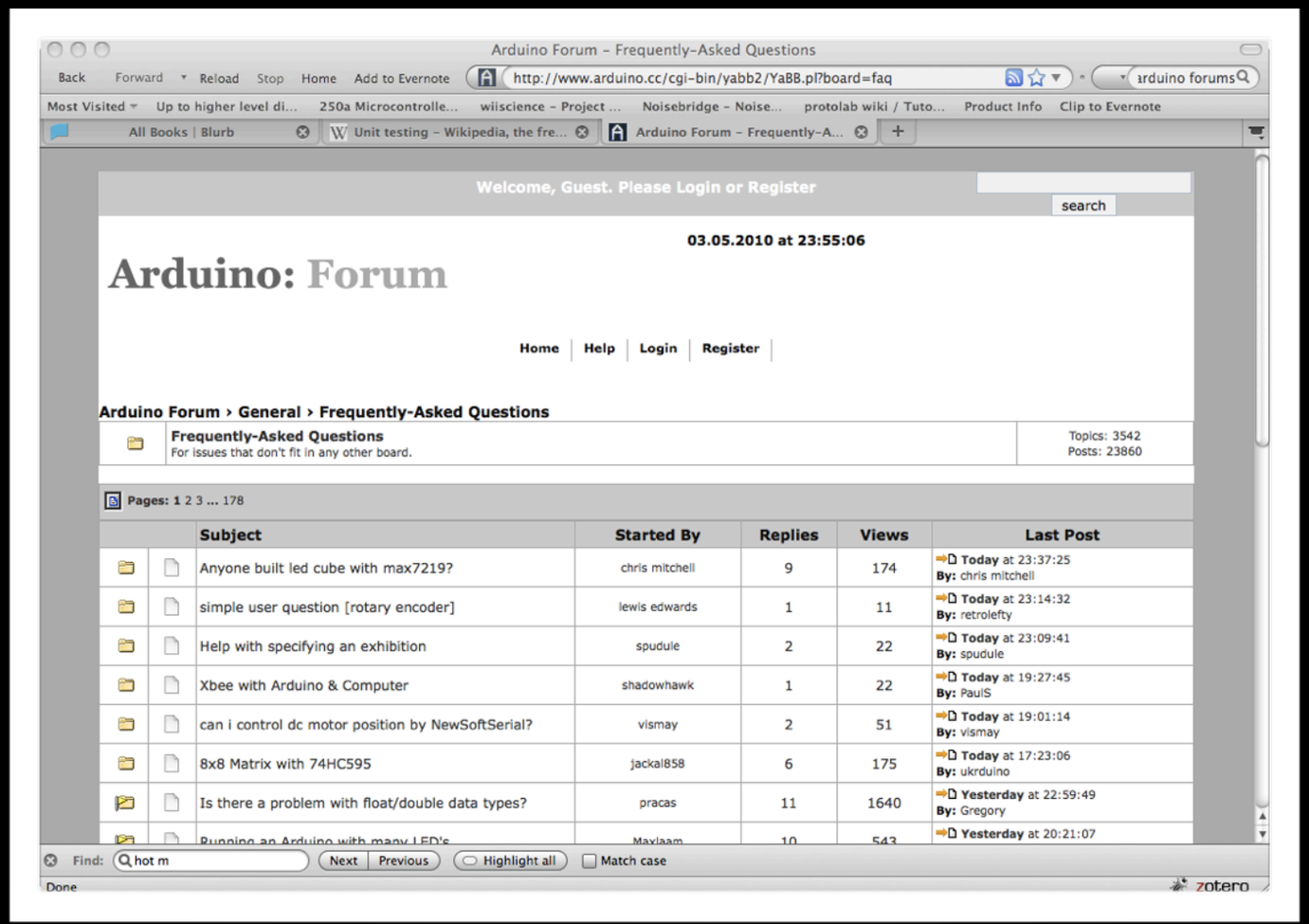


Image from www.allega.com/blog/

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

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?

Barebones MP3 Player

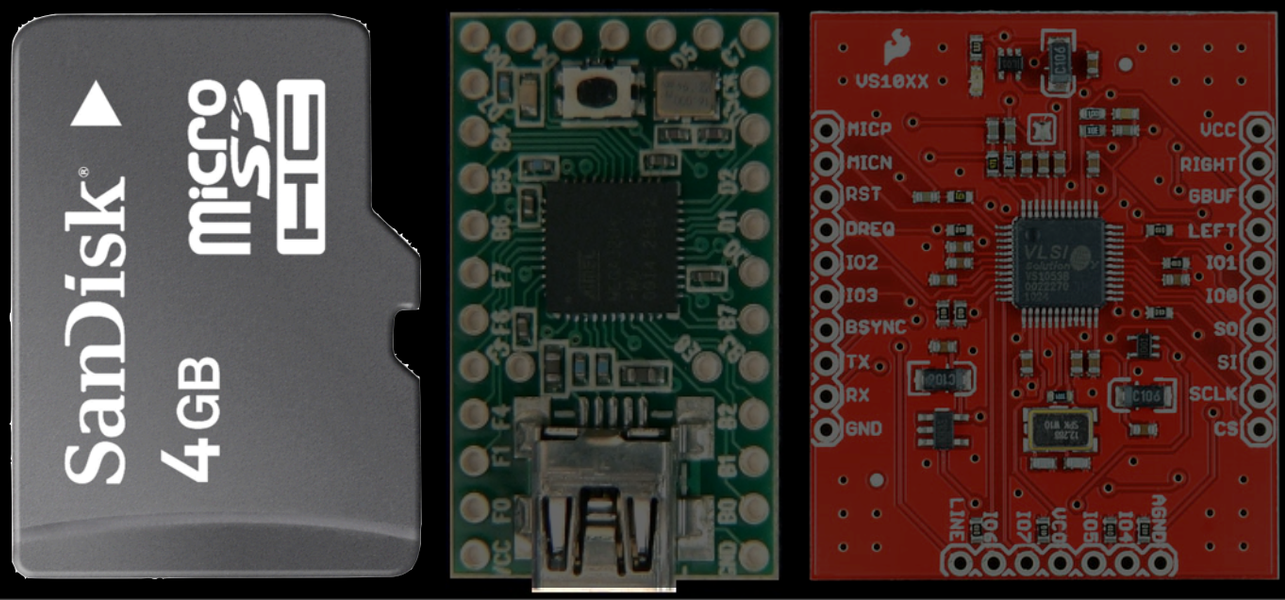
Press Play: Interactive Device Design | Aug 01, 2012

Barebones Overview

How to Play MP3s?

Storing Song Files

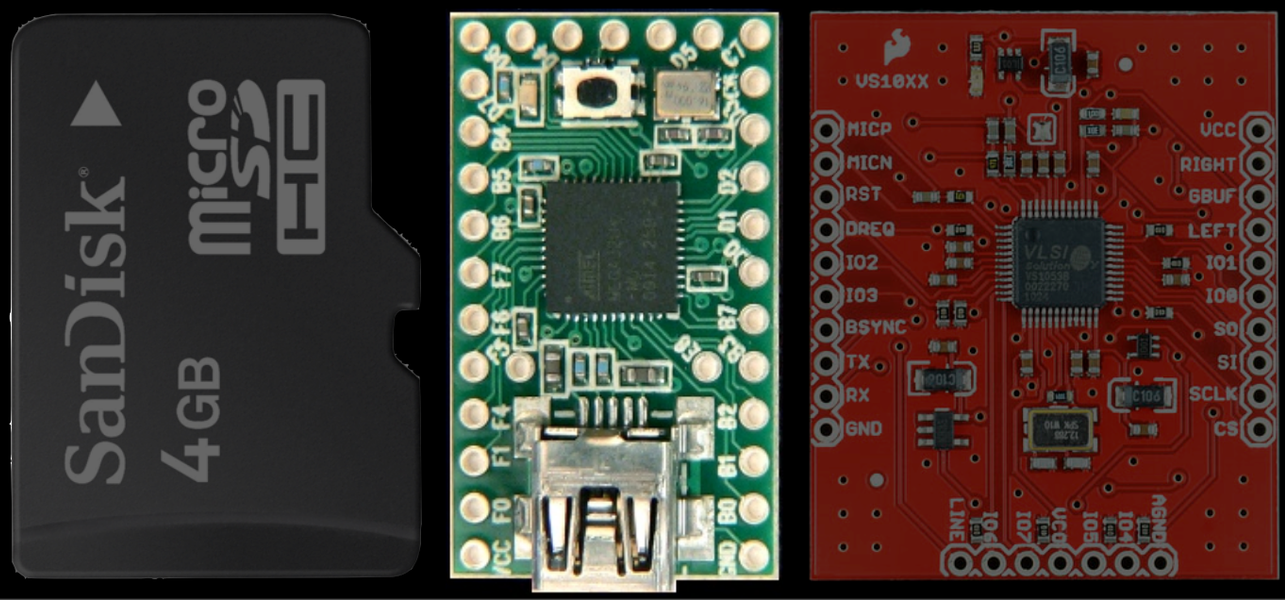
Songs are encoded and stored on microSD (FAT16 filesystem).



Barebones Overview

How to Play MP3s?

- Decode on the Microcontroller?
  1. Decoding MP3s is processor-intensive, and microcontrollers are not very powerful.
  2. Speakers/headphones expect an analog voltage signal, and microcontrollers only provide logic high/low.



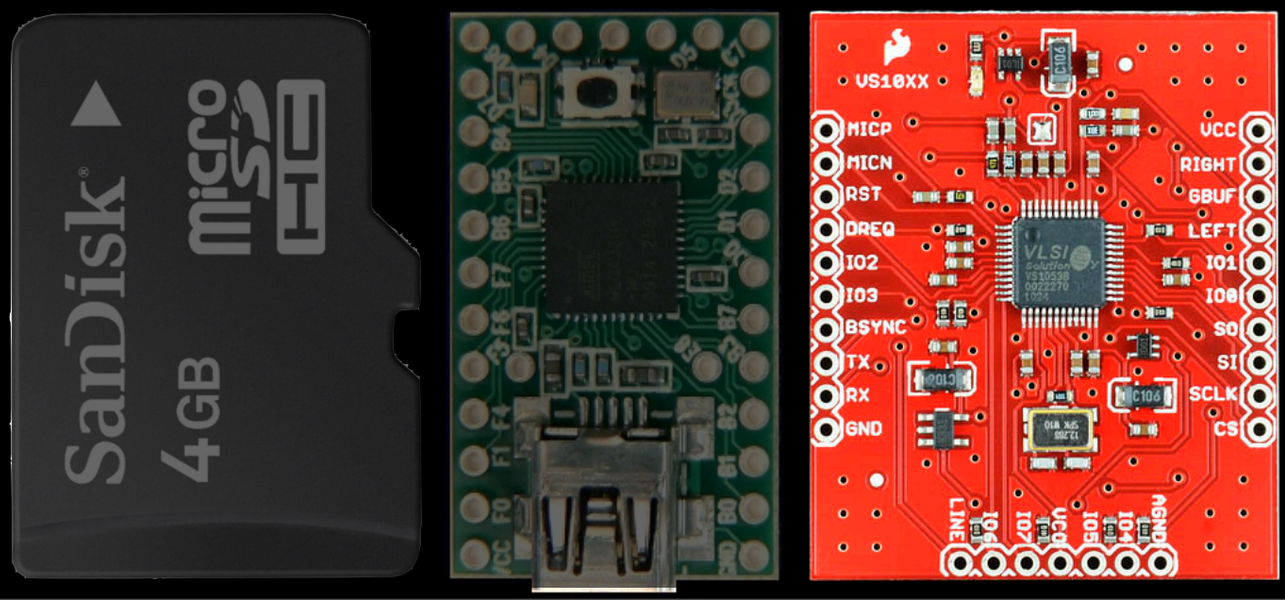
Barebones Overview

How to Play MP3s?

Enter the Decoder

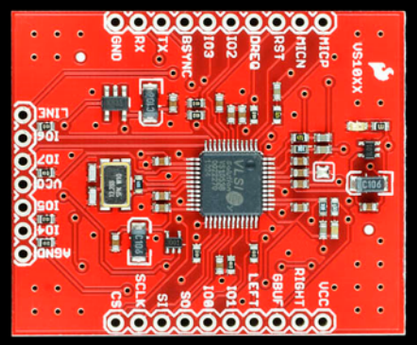
A hardware decoder decodes MP3s (and other encodings), and converts the result to an analog signal.

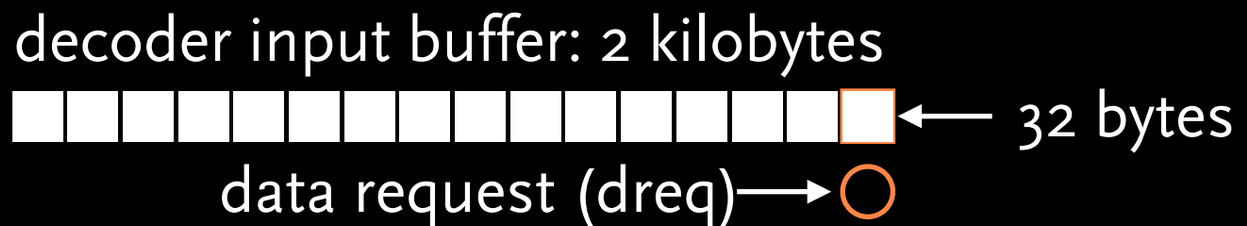
Our decoder (VS1053d) has ADC/DAC, 5.5 Kb SRAM, 8 GPIO pins, built-in MIDI synthesizer, audio amplifier...

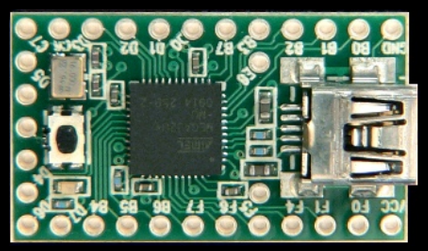


Barebones Overview

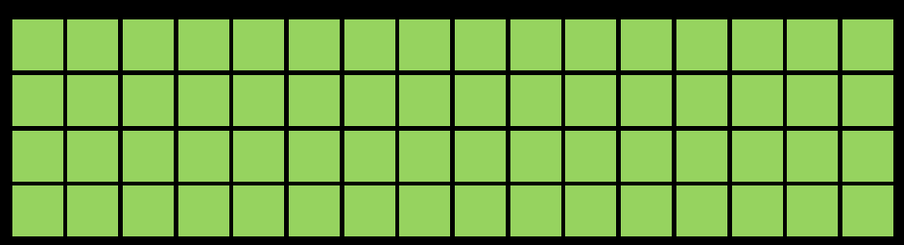
How to Play MP3s?







teensy board at 8 mhz



microSD card: 4 gigabytes or so

Barebones Structure

3 (Initial) Modules

- **Song**

  Routines that control setup and playback (a state machine).

- **Utilities**

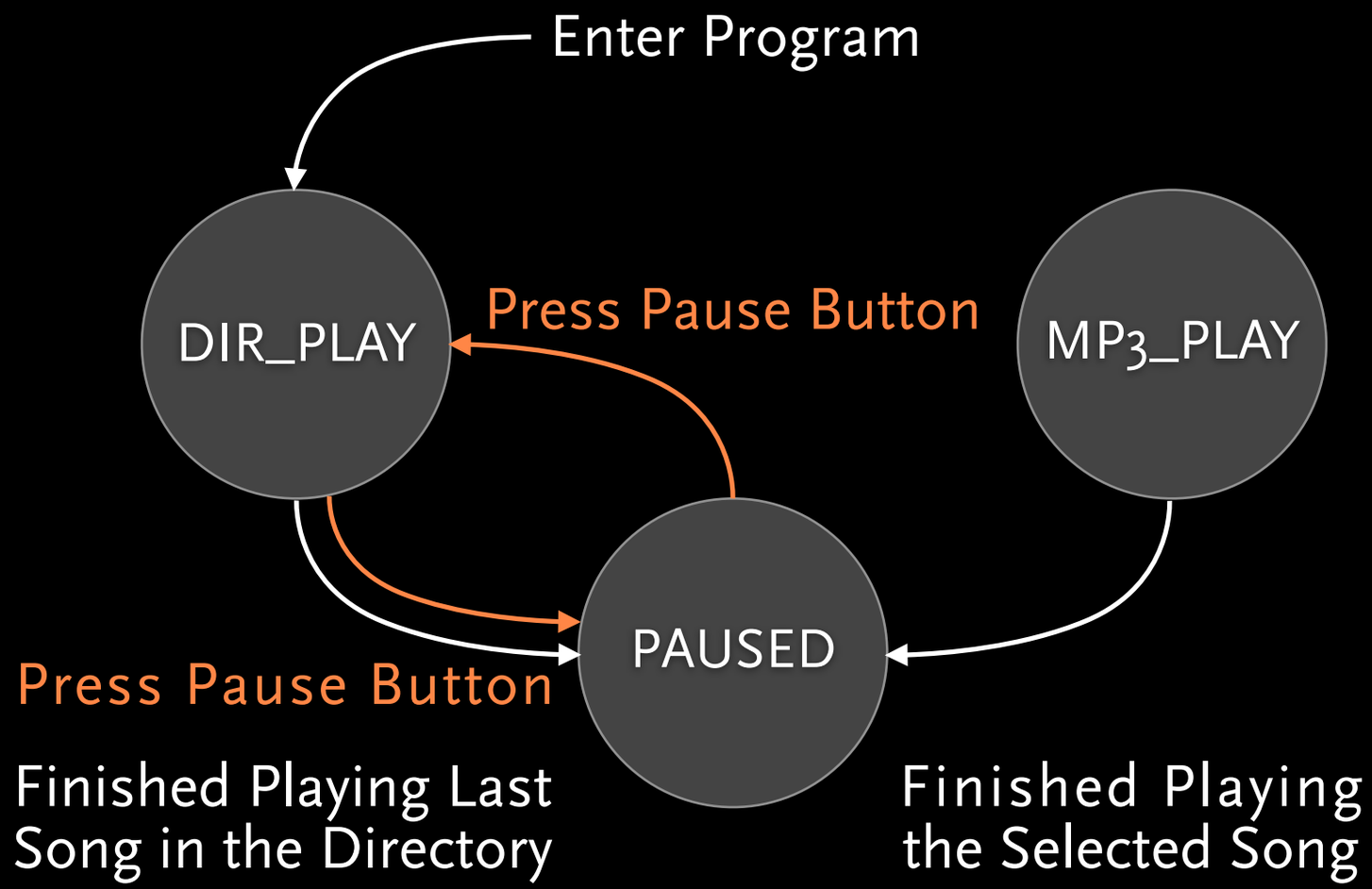
  Initialize microSD, store songs in EEPROM, print debug info.

- **ID3Tag**

  (Attempts to) find song title in ID3 V1 and V2 metadata tags.

Barebones Structure

3 (Initial) States



- These 3 states are coded as an enum of type ‘state’.
- To add a new state just include it in the enum.
- How would you add a PAUSE button?

Barebones Modules

‘Song’

- Where the player’s functionality (state machine) takes place.
- Opens a song file on the microSD card in order to play it.
- Shuttles data between microSD card and decoder input buffer.
- Setup the IO pins, graphic LCD, state enum, state machine.

```
void sd_file_open();

void mp3_play();

void dir_play();
```

Barebones Modules

‘Utilities’

- Where most of the microSD file and directory tasks take place.
- Initializes microSD card, opens root directory, lists files.
- Copies the filenames of song (MP3, WAV) files into EEPROM.
- Associates currently playing song with filename in EEPROM.

```
void sd_card_setup();

void sd_dir_setup();

void map_current_song_to_fn();
```

Barebones Modules

'ID3Tag'

- ID3 tags are metadata stored as plaintext within an MP3 file.
- Each value is in a frame that describes what it is and its size.
- The format and location of metadata has changed over time.
- Our ID3 tag reader focuses on 2 of these formats: v1 and v2.

```
void get_title_from_id3tag();
```

Extending the Player

Build an ‘Interface’ Module

```
void pause_play() {
  if the current state is DIR_PLAY {
    then set the current state to PAUSED
  }
  else {
    set the current state to DIR_PLAY;
  }
}

void skip_backward() {
  if not already at the first song {
    tell decoder to clear its buffer
    close song’s file in the microSD
    set the prior song to be current
    open the file of the current song
  }
}
```

Extending the Player

Build an 'Interface' Module

```
void pause_play() {
  if (current_state == DIR_PLAY) {
    current_state = PAUSED;
  }
  else {
    current_state = DIR_PLAY;
  }
}

void skip_backward() {
  if (current_song != 0) {
    Mp3.cancel_playback();
    sd_file.close();
    current_song--;
    sd_file_open();
  }
}
```

Program Execution

Write Non-Blocking Code!

- **Important Notes:**

  Other than a few very quick functions, the player uses all non-blocking code. That is, there are no routines that 'hold up' other routines from operating as usual.

  If your program has brief pauses between segments of song playback, then you've likely written blocking code (a 'for' or 'while' loop that waits for an event is often the cause).

  Try using 'if' statements or interrupt service routines (ISRs).

Lab #6 Preview

Barebones MP3 Player: Yeah!

First Project Deliverable

A ‘Critical Function’ Prototype