

Why Arduino?

~

An Introduction

Ed Nisley • KE4ZNU
softsolder.wordpress.com

~
MHV Linux Users Group
February 2010



Upcoming Events

- Why You Should Care
- Where It All Began
- Microcontrollers
- Programming Model
- Programming Gotchas
- Commercial Shields
- DIY Gadgetry
- Show-n-Tell



Proposed Arduino Tower
(No Relation...)
Milan Italy

Why You Should Care

- ♦ Typical programming excludes The Real World
 - ♦ Virtualization is the (Old) New Thing
 - ♦ Hardware is (allegedly) fungible
- ♦ **Everything** depends on Other People's Code
 - ♦ Hidden complexity remains complex
 - ♦ Debugging / isolation becomes difficult
- ♦ The Newbie problem
 - ♦ Where do they start?
 - ♦ Why should they care?

The Newbie Problem

hello, world

<!--

This is the source file for the "Hello World of AJAX" tutorial
Please visit <http://www.DynamicAJAX.com> for more great AJAX
source code and tutorials.
Copyright 2006 Ryan Smith / 345 Technical / 345 Group.

-->

<html>

<head>

<title>The Hello World of AJAX</title>

<script language="JavaScript" type="text/javascript">

//Gets the browser specific XmlHttpRequest Object

function getXmlHttpRequestObject() {

if (window.XMLHttpRequest) {

return new XMLHttpRequest(); //Not IE

} else if(window.ActiveXObject) {

return new ActiveXObject("Microsoft.XMLHTTP"); //IE

} else {

//Display your error message here.

//and inform the user they might want to upgrade

//their browser.

alert("Your browser doesn't support the XmlHttpRequest object. Better upgrade to Firefox.");

}

}

//Get our browser specific XmlHttpRequest object.

var receiveReq = getXmlHttpRequestObject();

//Initiate the asynchronous request.

function sayHello() {

//If our XmlHttpRequest object is not in the middle of a request, start the new asynchronous call.

if (receiveReq.readyState == 4 || receiveReq.readyState == 0) {

//Setup the connection as a GET call to SayHello.html.

//True explicitly sets the request to asynchronous (default).

receiveReq.open("GET", 'SayHello.html', true);

//Set the function that will be called when the XmlHttpRequest objects state changes.

receiveReq.onreadystatechange = handleSayHello;

//Make the actual request.

receiveReq.send(null);

}

}

//Called every time our XmlHttpRequest objects state changes.

function handleSayHello() {

//Check to see if the XmlHttpRequests state is finished.

if (receiveReq.readyState == 4) {

//Set the contents of our span element to the result of the asynchronous call.

document.getElementById('span_result').innerHTML = receiveReq.responseText;

}

}

</script>

</head>

<body>

<!-- Clicking this link initiates the asynchronous request -->

Say Hello

<!-- used to display the results of the asynchronous request -->

</body>

</html>

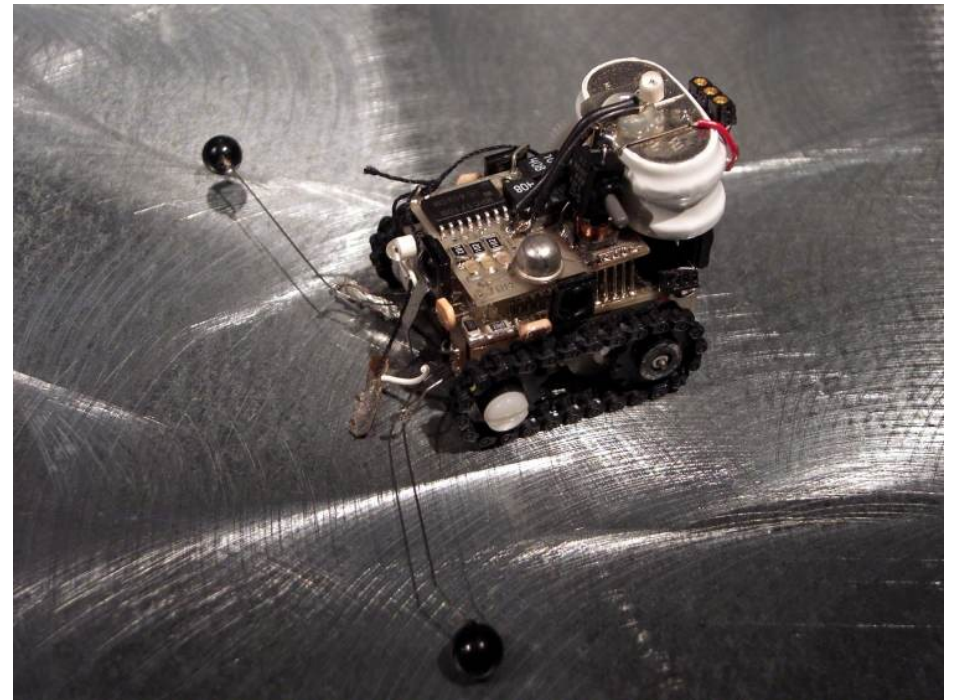
Ah-hem...

The Newbie Problem

- ♦ “All you need to do is ...”
 - ♦ Nested abstractions
 - ♦ Many simultaneous learning curves
- ♦ When it breaks, it's *really* broken
 - ♦ Too many moving parts
- ♦ No visceral reward
 - ♦ It's just some pixels
 - ♦ Crisis of rising expectations

Where It All Began

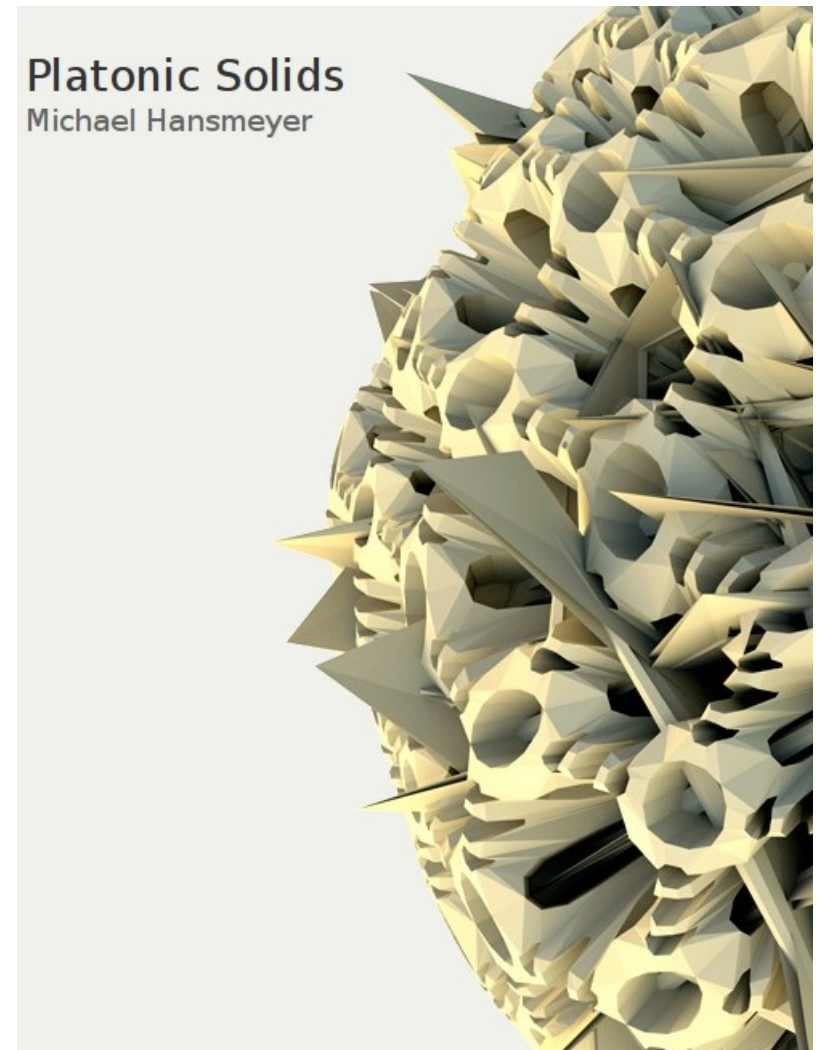
- ♦ MIT AI Lab – '70s
 - ♦ Advanced widgetry
- ♦ MIT Media Lab – '80s
 - ♦ Human accessible
 - ♦ Direct manipulation
 - ♦ Programmable *stuff*
 - ♦ Rapid prototyping
 - ♦ LEGO® Mindstorms®, et al



Processing

Processing was founded by Ben Fry and Casey Reas in 2001 while both were John Maeda's students at the MIT Media Lab.

www.processing.org/about



Processing

- ♦ *Processing is*
 - ♦ *a simple programming environment*
 - ♦ *that was created to*
 - ♦ *make it easier to*
 - ♦ *develop visually oriented applications*
 - ♦ *with an emphasis on animation and*
 - ♦ *providing users with instant feedback*
 - ♦ *through interaction.*

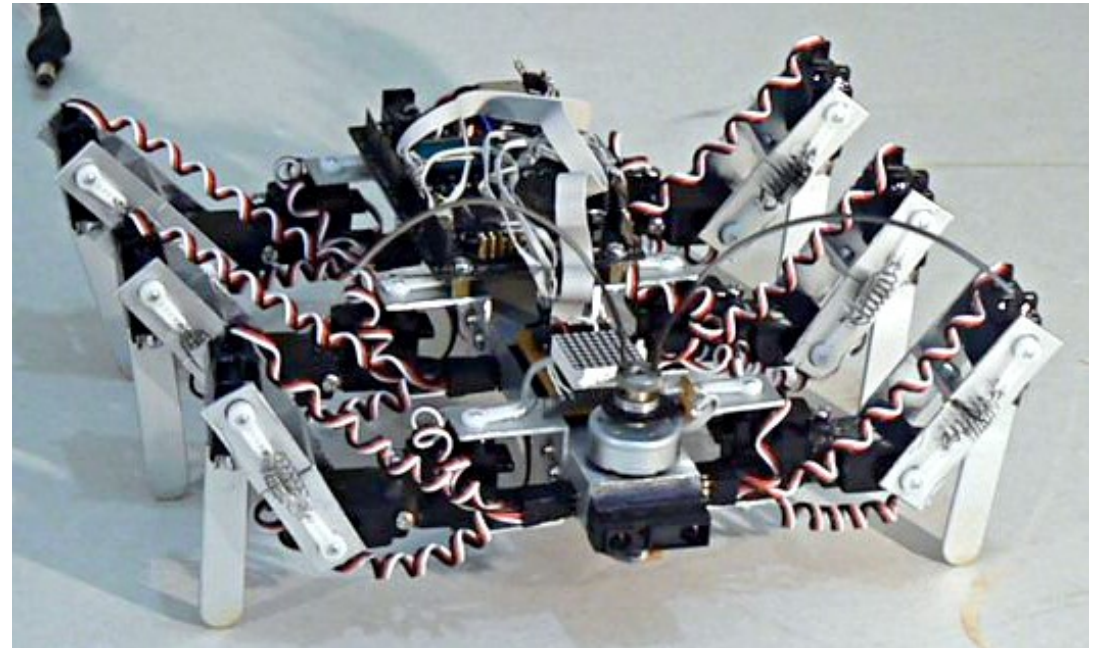
<http://processing.org/learning/gettingstarted>

Wiring

*Hernando
[Barragán] started
Wiring in **Summer
2003** and he has
been developing it
since.*

***Wiring** builds on
Processing ...*

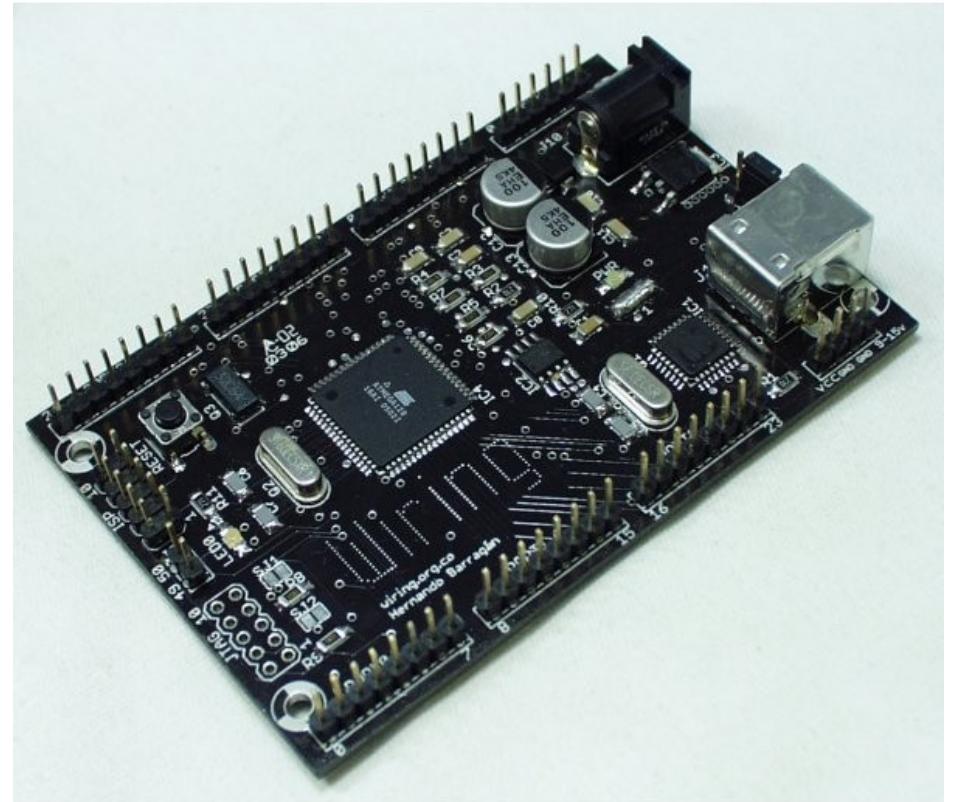
www.wiring.org.co/about.html



www.pablogindel.com/2009/07/samsa-el-hexapodo

Wiring

- ♦ *Wiring is*
 - ♦ *both a programming environment and*
 - ♦ *an electronics prototyping input/output board*
- ♦ *for exploring the electronic arts and tangible media.*

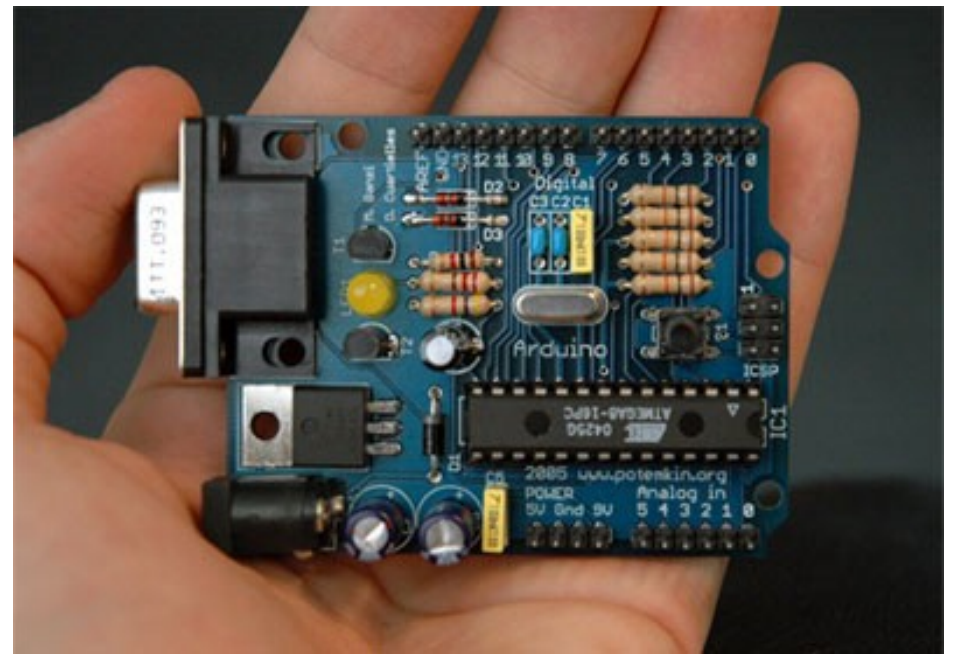


www.wiring.org.co/about.html

www.sparkfun.com/commerce/product_info.php?products_id=744

Arduino

- ♦ In *winter 2005*, [Massimo] Banzi was discussing the problem with David Cuartielles ...
- ♦ [H]is students ... couldn't find an *inexpensive*, powerful microcontroller to drive their *arty robotic projects*

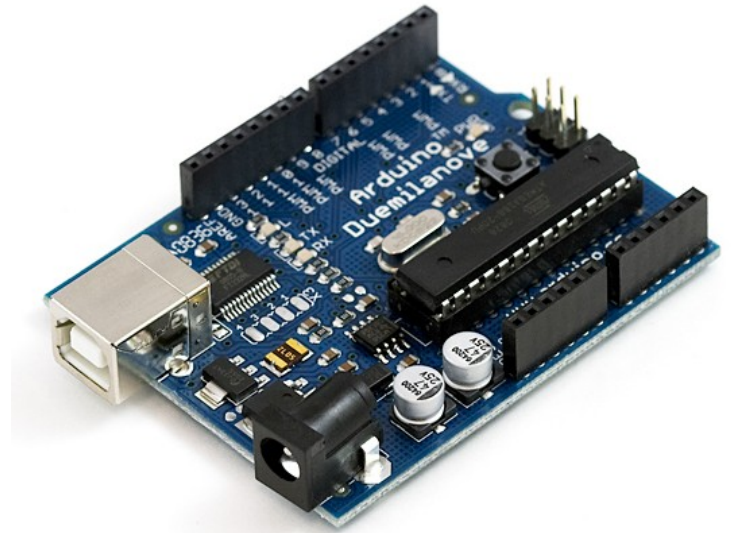


<http://arduino.cc/en/Main/ArduinoBoardSerial>

Arduino

- ♦ The *microcontroller* ... is programmed using
 - ♦ the Arduino programming language (*based on Wiring*) and
 - ♦ the Arduino development environment (*based on Processing*)

www.arduino.cc



<http://static.sparkfun.com/images/products/00666-03-L.jpg>

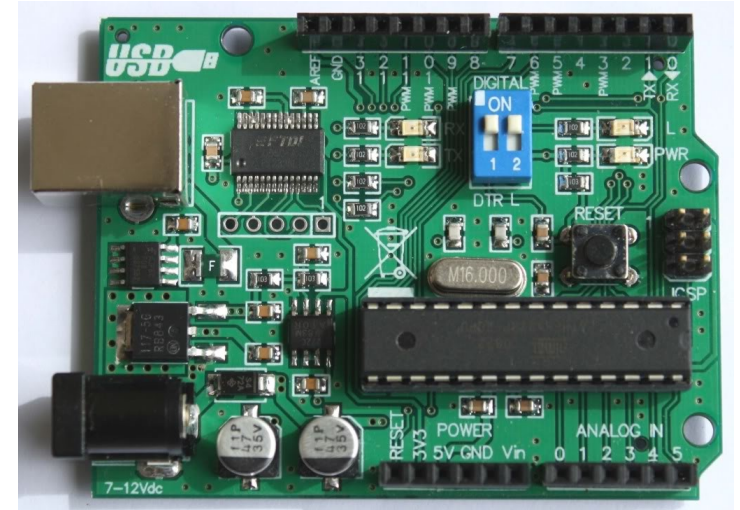
Be Careful Where You Sit



<http://www.gulliverproducts.com/Art/D23.html>

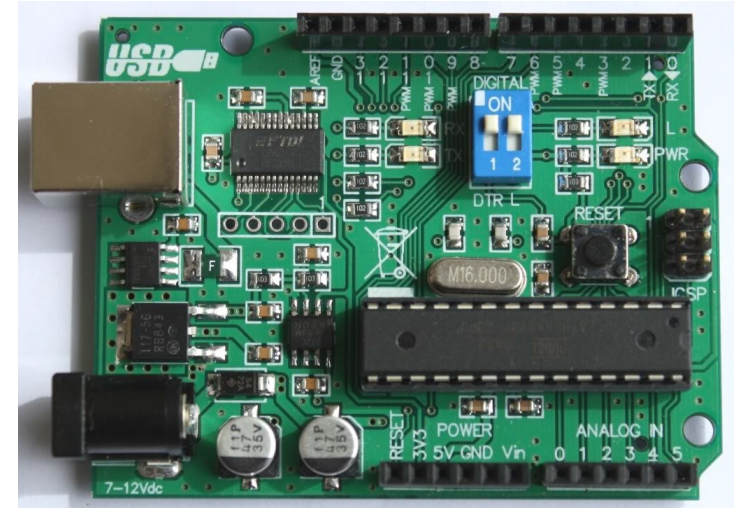
Open-Source Hardware

- ◆ *Arduino* is a TM
 - ◆ **duino* is not
- ◆ Commercial copies
 - ◆ Licensing fees
 - Except in slack IP regimes
- ◆ Derivatives are fine
 - ◆ Use a different name



Open-Source Hardware

- ◆ *Fino328* (top)
 - ◆ \$1.50 + \$28 ship – Thailand
- ◆ *ARDUINO 2009 Duemilanove* (bottom)
 - ◆ “manufactured based on aduino duemilanove reference design”
 - ◆ “Will Not Ship to Italy”
 - ◆ \$27 + \$0 – Hong Kong
- ◆ Sparkfun genuine licensed
 - ◆ \$30 + \$7 s/h – Boulder CO



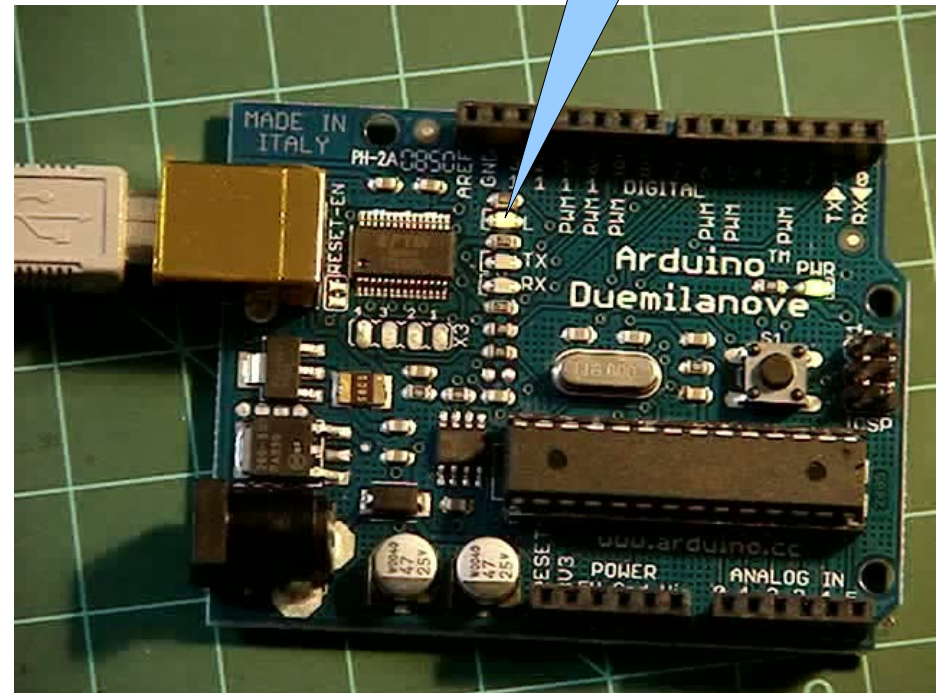
Hello World

```
int ledPin = 13;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

... blink ...



Microcontrollers

This. Is. Not. A. PC.

ATmega168

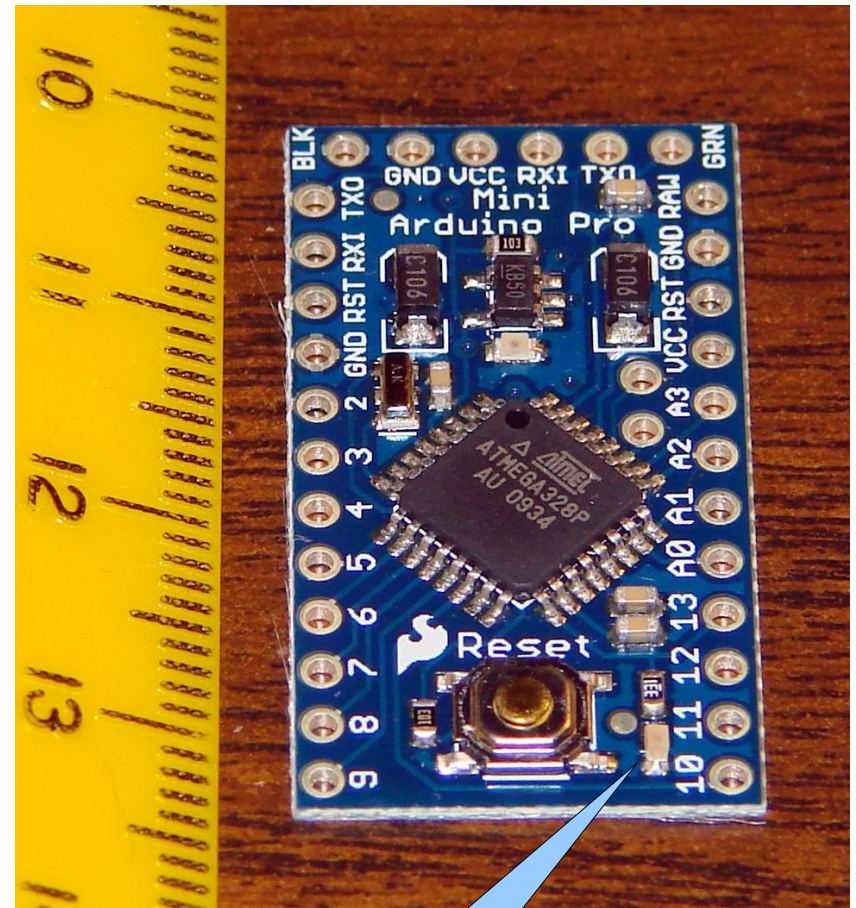
- ♦ 16 kB program memory
 - ♦ 14 kB available for user code
 - ♦ 2 kB for bootloader / self-programmer
- ♦ 1 kB RAM (8-bit data path)
 - ♦ Volatile data memory
 - ♦ Variables + Stack + Heap...
- ♦ 512 B EEPROM
 - ♦ Nonvolatile *write-occasionally* data memory
 - ♦ 100 k write cycles (≈ 200 seconds at full throttle)

ATmega328

- ♦ 32 kB program memory
 - ♦ 30 kB available for user code
 - ♦ 2 kB for bootloader / self-programmer
- ♦ 2 kB RAM (still 8-bit data path)
 - ♦ Volatile data memory
 - ♦ Variables + Stack + Heap...
- ♦ 1 kB EEPROM
 - ♦ Nonvolatile *write-occasionally* data memory
 - ♦ 100 k write cycles (\approx 200 seconds at full throttle)

Hardware I/O

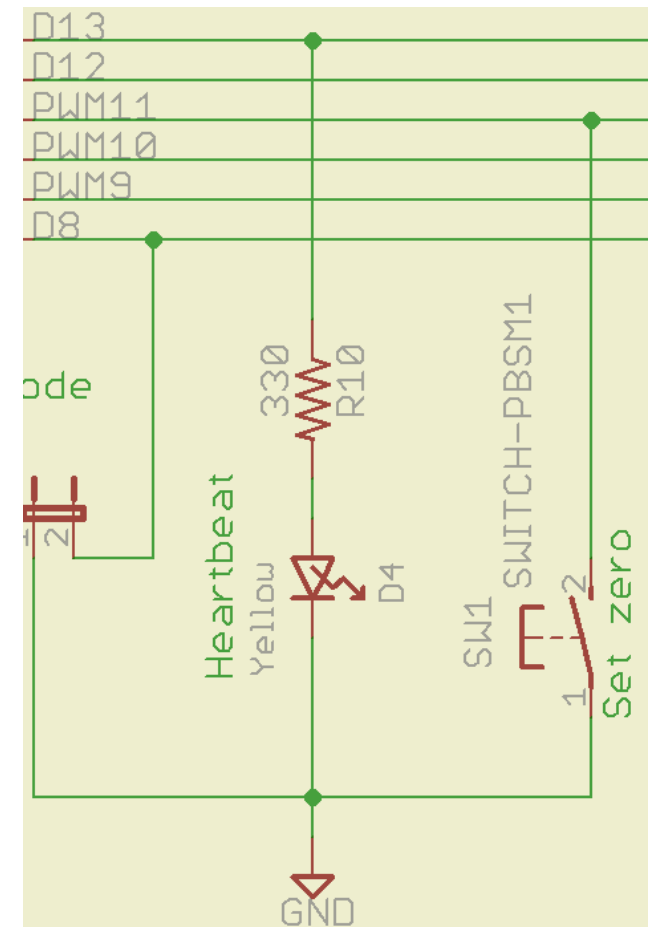
- ♦ 13 Digital I/O bits
 - ♦ 2 = serial TxD/RxD
 - ♦ Direct LED drive
 - One LED...
- ♦ 6 “Analog” outputs
 - ♦ Pulse-width modulated
 - ♦ Choose digital **or** PWM
 - ♦ 8-bit resolution
- ♦ 6 Analog inputs
 - ♦ 10-bit resolution **at best**



... blink ...

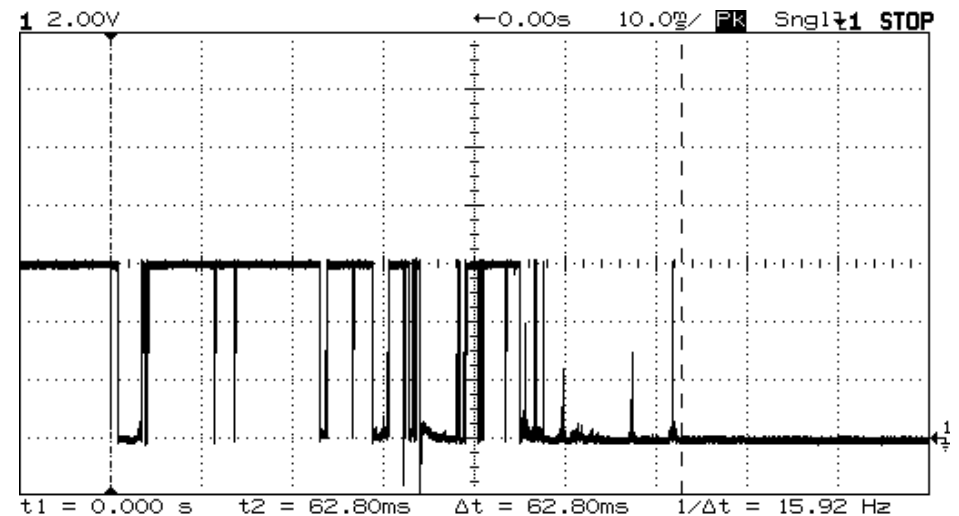
Programming

- ◆ Bare Silicon Techniques
 - ◆ Direct I/O bit manipulation
 - ◆ Interrupt handlers (experts only!)
 - ◆ Very limited code portability
- ◆ Eschew Obfuscation
 - ◆ **Hard** memory limits
 - ◆ Count the cycles!
- ◆ It's surprisingly fast
 - ◆ 16 MHz clock \approx 16 MIPS



How Fast Is It?

- ♦ Switch Bounce
 - ♦ 62.8 ms
 - ♦ ≈ 1 million instructions
 - ♦ That's a lot ...
 - Even in C
- ♦ Users are surprised
 - ♦ They're not engineers
 - That's OK ...
 - ♦ It's still embedded programming



Other People's Code

- ♦ Many libraries!
 - ♦ Most work well
- ♦ Many functions!
 - ♦ Most are useful
- ♦ Many bytes!
 - ♦ Most are necessary
- ♦ Many authors!
 - ♦ Most write good code...

Bounce is a library for Arduino ... It debounces digital inputs and more.
<http://www.arduino.cc/playground/Code/Bounce>

```
Done uploading.  
Binary sketch size: 896 bytes (of a 14336 byte maximum)
```

Blink: 896 bytes

```
Done uploading.  
Binary sketch size: 1444 bytes (of a 14336 byte maximum)
```

+ Bounce: 1444 bytes

ATmega168

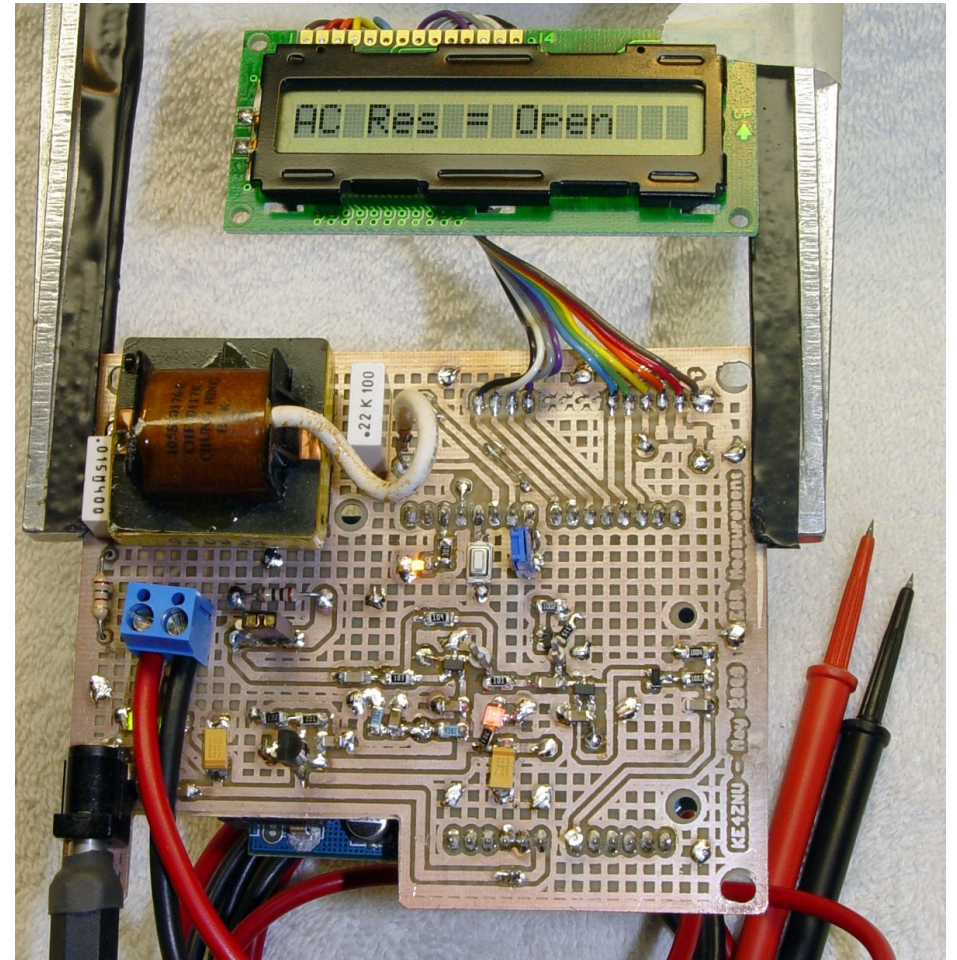
- ♦ 16 kB program memory
 - ♦ 14 kB available for user code
 - ♦ 2 kB for bootloader / programmer
- ♦ 1 kB RAM (8-bit data path)
 - ♦ Volatile data memory
 - ♦ Variables + Stack + Heap...
- ♦ 512 B EEPROM
 - ♦ Nonvolatile *write-occasionally* data memory
 - ♦ 100 k write cycles (≈200 seconds at full throttle)

Microcontrollers

This. Is. Not. A. PC.

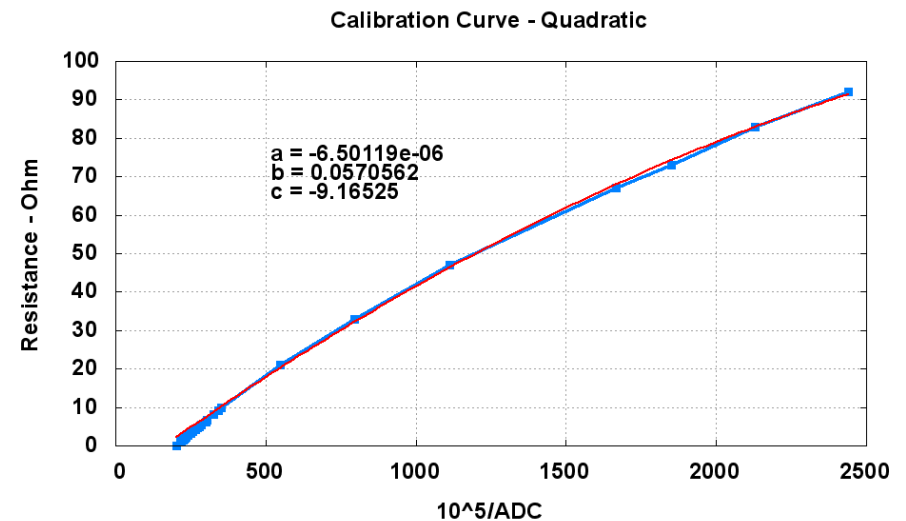
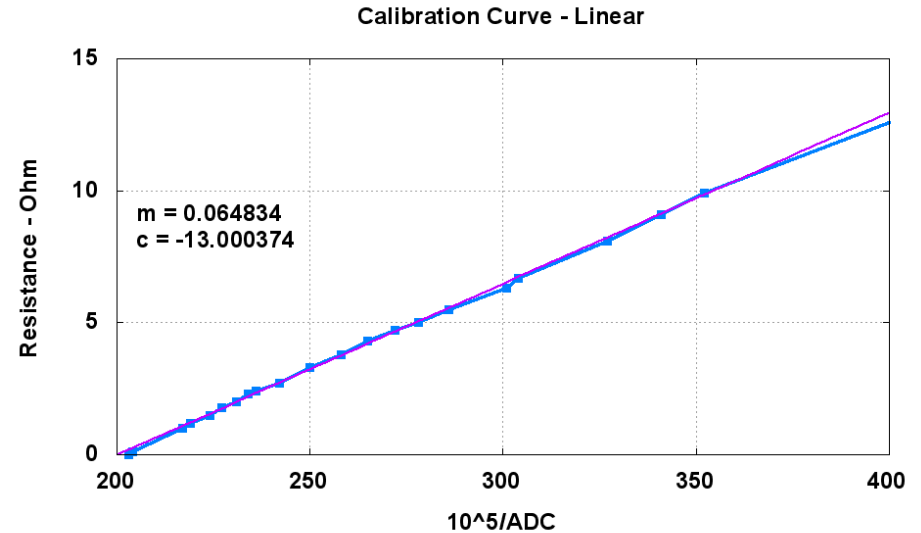
Programming Model

- ♦ Small projects
 - ♦ There Is No ROM
- ♦ Single-threaded loop()
 - ♦ Plus IRQ handlers...
- ♦ Not much I/O
 - ♦ There Is No Bus
 - ♦ SPI & I²C gadgets
- ♦ Not much data
 - ♦ There Is No RAM



Programming Model

- ◆ It's BASICally C
 - ◆ Or C++ for Dummies
- ◆ In the KISS style
 - ◆ Don't be *too* clever
 - ◆ Don't be *too* abstract
 - ◆ Don't be *too* general
- ◆ No place for Git
 - ◆ One source file
 - You can have more...



Programming Model

All the usual suspects

```
void loop() {
```

```
int RawSense;  
int Resistance;  
float FloatRes;  
float x;
```

Hardware I/O

```
RawSense = analogRead(PIN_USENSE); // 0..1023 counts  
if (RawSense <= 1) { // very low values mean open circuit  
    Resistance = RESISTANCE_BAD;  
}  
else { // some sort of resistance is connected  
    Resistance = LinNumerator / RawSense + LinOffset; // try linear fit  
    if (Resistance > RESISTANCE_BREAK) { // out of linear range, use quadratic  
        x = ScaleADC / RawSense; // get plotting-scale 1/ADC values  
        FloatRes = x * (CoeffA * x + CoeffB) + CoeffC; // floating point value  
        Resistance = ScaleResistance * FloatRes + 0.5; // back to integer  
    }  
    if ((Resistance > RESISTANCE_BAD) != (Resistance == 0) ) { // out-of-range?  
        Resistance = RESISTANCE_BAD;  
    }  
}
```

Fancy computations are OK
but can be achingly slow

Programming Model

```
TCCR1B = 0x00;                // stop Timer1 clock for register updates

// Clear OC1A on match, P-F Corr PWM Mode: lower WGM1x = 00

TCCR1A = 0x80 | 0x00;

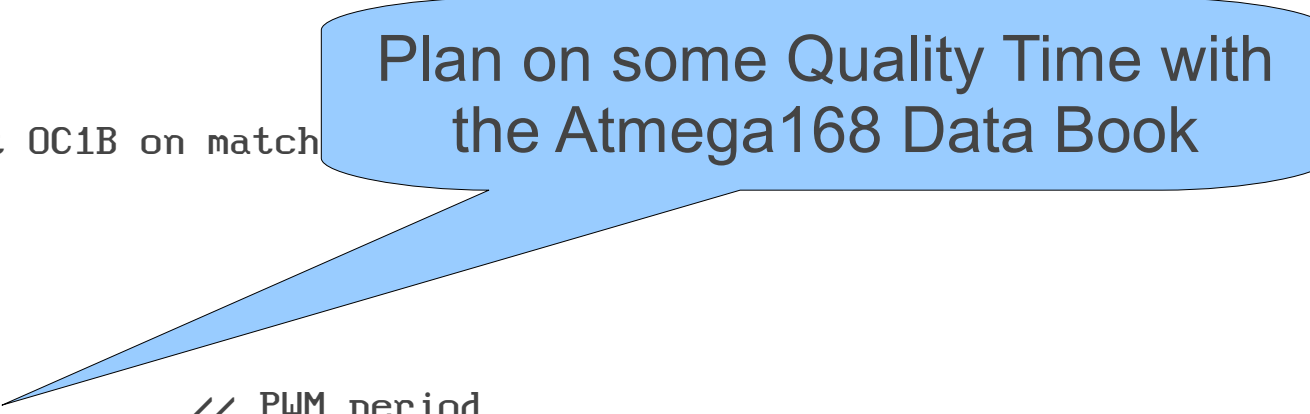
// If push-pull drive, set OC1B on match

#if PUSH_PULL
    TCCR1A |= 0x30;
#endif

ICR1 = PERIOD_TICKS;          // PWM period
OCR1A = PERIOD_TICKS / 2;      // ON duration = drive pulse width = 50% DC
OCR1B = PERIOD_TICKS / 2;      // ditto - use separate load due to temp buffreg
TCNT1 = OCR1A - 1;             // force immediate OCR1x compare on next tick

// upper WGM1x = 10, Clock Sel = prescaler, start Timer 1 running

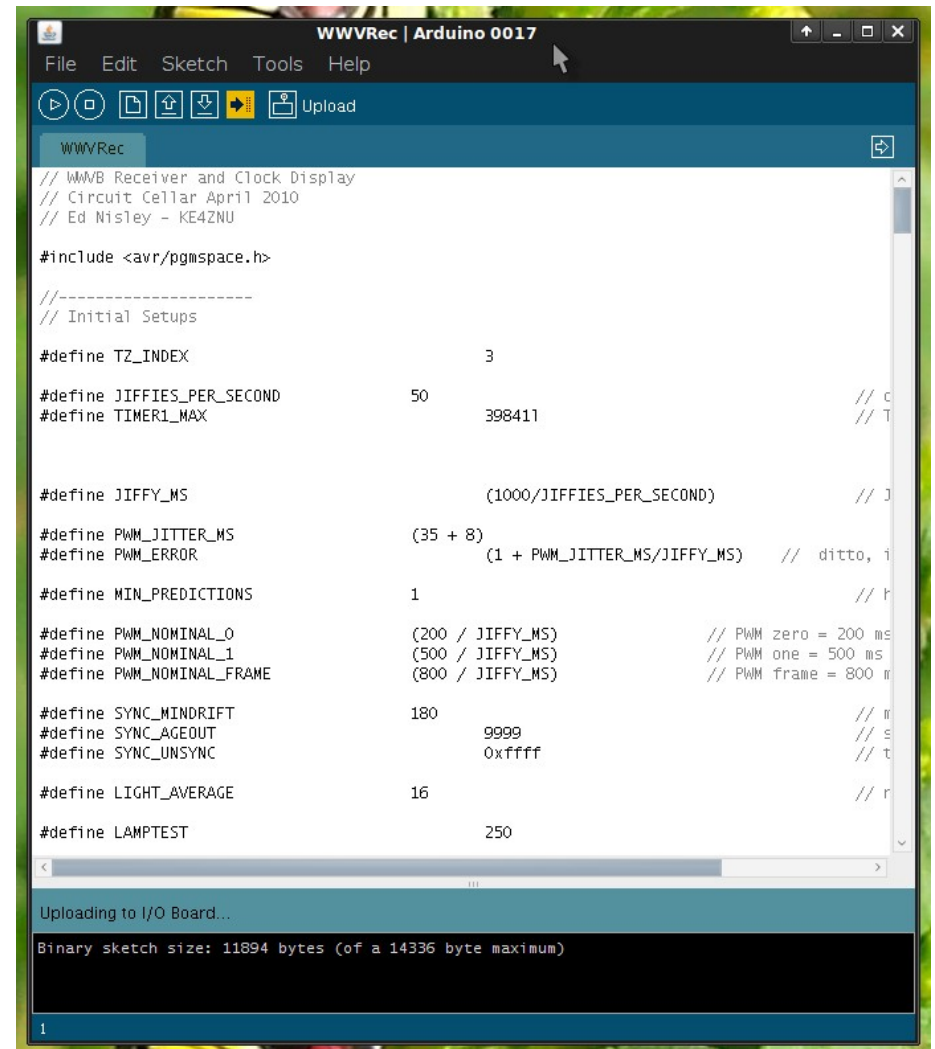
TCCR1B = 0x10 | TCCR1B_CS20;
```



Plan on some Quality Time with
the Atmega168 Data Book

Arduino IDE

- ◆ Yet Another IDE
 - ◆ Very few features
- ◆ Simple programs
 - ◆ Exactly the *use case*
- ◆ Written in Java
 - ◆ ‘Nuff said
- ◆ “Use External Editor”
 - ◆ Click to upload



Development Cycle

- ♦ Write some code
- ♦ Compile & **upload** it
 - ♦ USB/serial link
- ♦ Find errors
 - ♦ Where?
 - ♦ How?
- ♦ Iterate
 - ♦ Hidden errors...



Programming Gotchas

- ♦ Don't be too clever
 - ♦ Compiler errors...
- ♦ Use OPC with care
 - ♦ Some: way clever
- ♦ Rapid prototyping
 - ♦ Hardware, too
- ♦ KISS!
 - ♦ With a vengeance...

The error message lump resulting from compiling that looks like:

```
In function 'void setup()':  
    error: 'OUTPUT' was not  
declared in this scope  
In function 'void loop()':  
In function 'int main()':
```

Really?

Programming Gotchas

*Although the Arduino language looks like C and parses like C and runs like C,
it's not really C.*

*It's a specialized language gnawed on
by a vast conversion monster and
eventually spat out as executable
ATmega-style instructions.*

With That In Mind...

Simple projects work perfectly

~ because ~

that's what Arduino is all about

Arduino Application Model

Hardware equivalent of
Unix-oid filter program

~

Do one thing, do it well

~

VI

Not Emacs

Hardware Shields

Zowie!

- ◆ Other People's Hardware
 - ◆ Commercial
 - ◆ DIY kits & plans
- ◆ Large building blocks
 - ◆ Eliminates most soldering
 - ◆ Economics?
- ◆ One shield per project
 - ◆ There Is No Bus
- ◆ One CPU per project

Results 1 - 18 of about 23,000 (0.21 seconds)



Arduino
468 x 293 -
55k - jpg
bridell.com



first Arduino
500 x 375 -
64k - jpg
[blog...](#)



Arduino
800 x 600 -
48k - jpg
watterott.com



Arduino
640 x 427 -
171k - jpg
[thebasementsc...](#)



for a Arduino
500 x 381 -
50k - jpg
ladyada.net



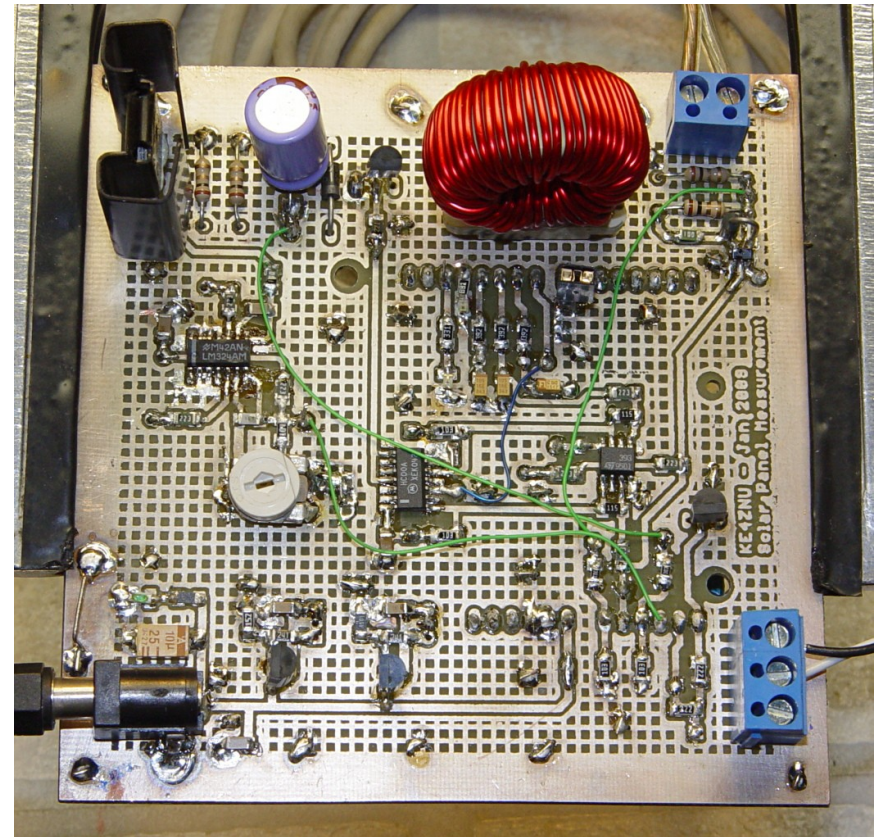
Arduino
480 x 360 -
34k - jpg
[electronics...](#)

Solar Power Data Logger

Achtung!

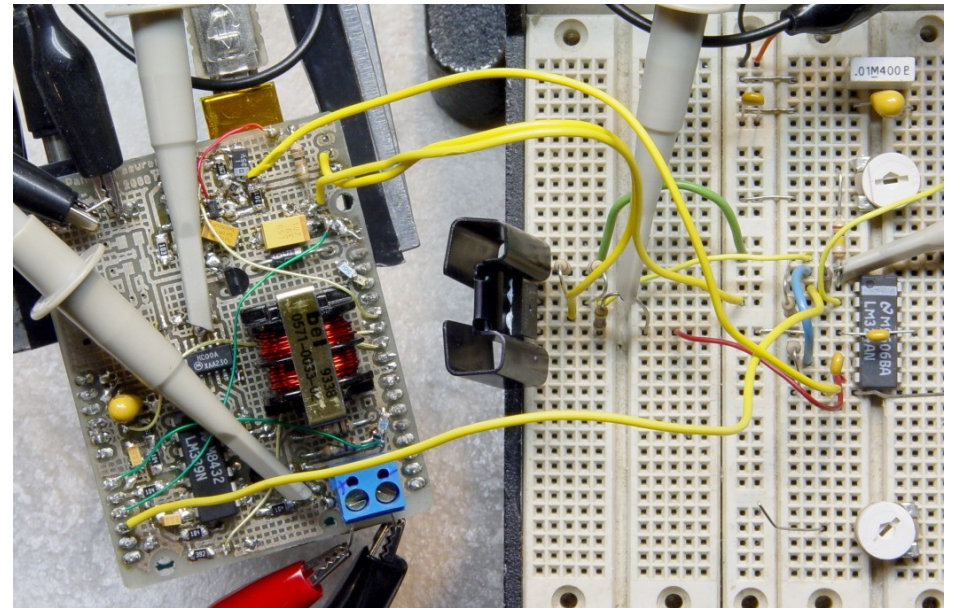
5.5 kB code

- ♦ Optimum Power Point
 - ♦ Vary load current
 - ♦ Measure I, V, W
 - ♦ Dump results to PC
- ♦ Panel voltage booster
 - ♦ Demo circuitry
- ♦ Notice the board size?
- ♦ Circuit Cellar June 09



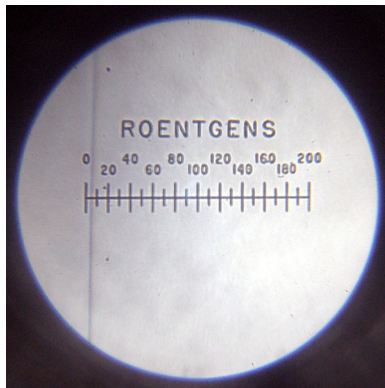
Solar Power Data Logger

- ♦ This one didn't work...
 - ♦ Inductor saturation
 - ♦ Analog noise
- ♦ Notice the board size?
- ♦ Analog vs Digital?
 - ♦ Analog wins!
 - Every damn time
- ♦ Circuit Cellar April 09

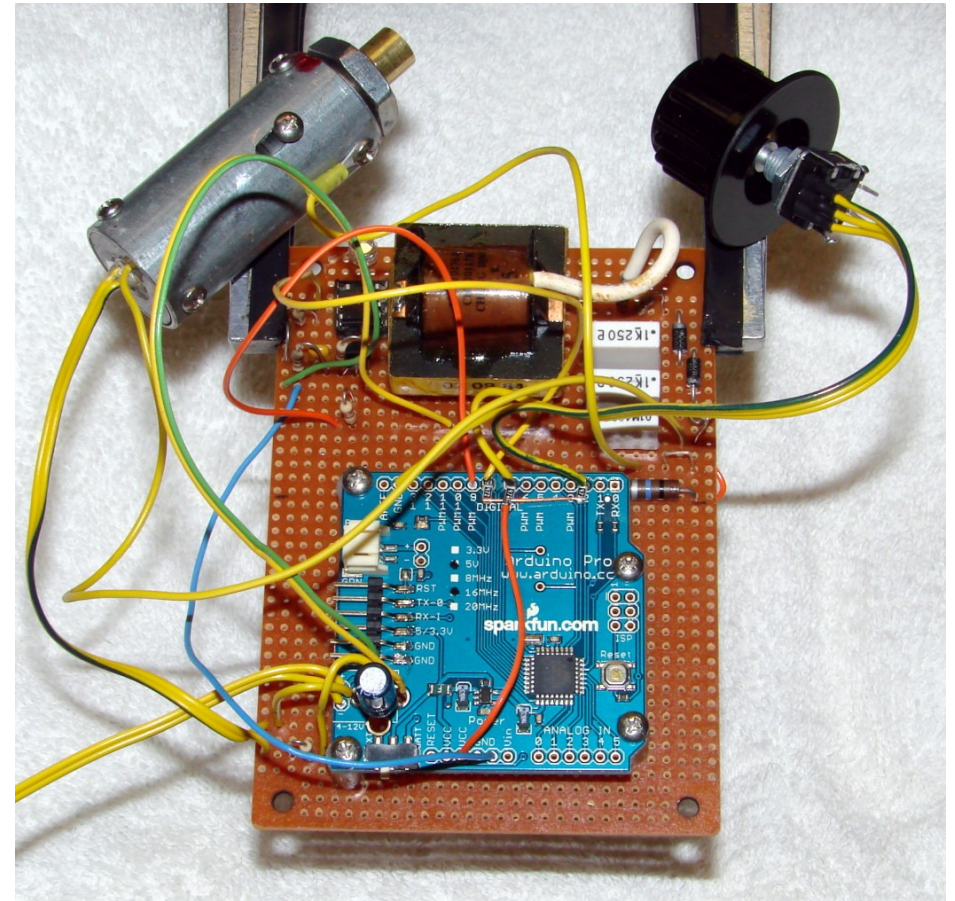


Dosimeter Charger

- ◆ Quadrature Encoder
 - ◆ Voltage adjustment
- ◆ PWM “analog” output
 - ◆ Hardware PWM
 - ◆ 1:25 step-up → 200 V
- ◆ Circuit Cellar Aug 09



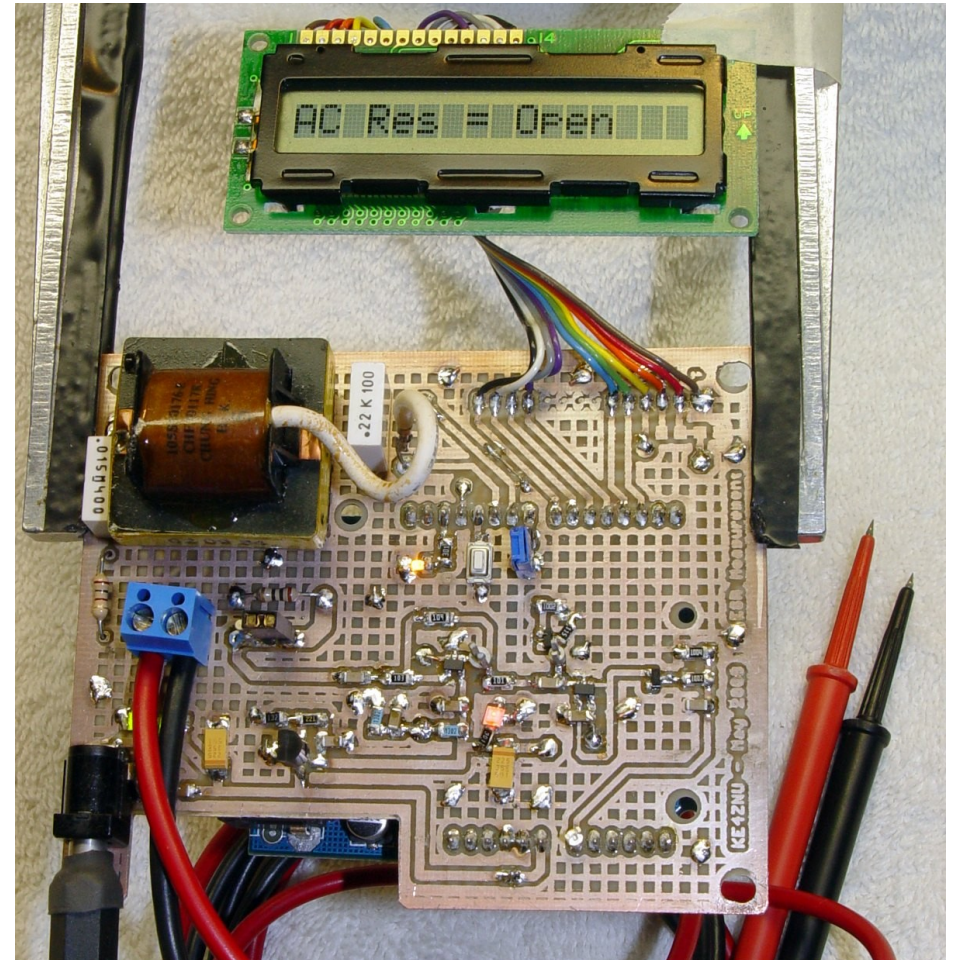
4.4 kB



Capacitor ESR Meter

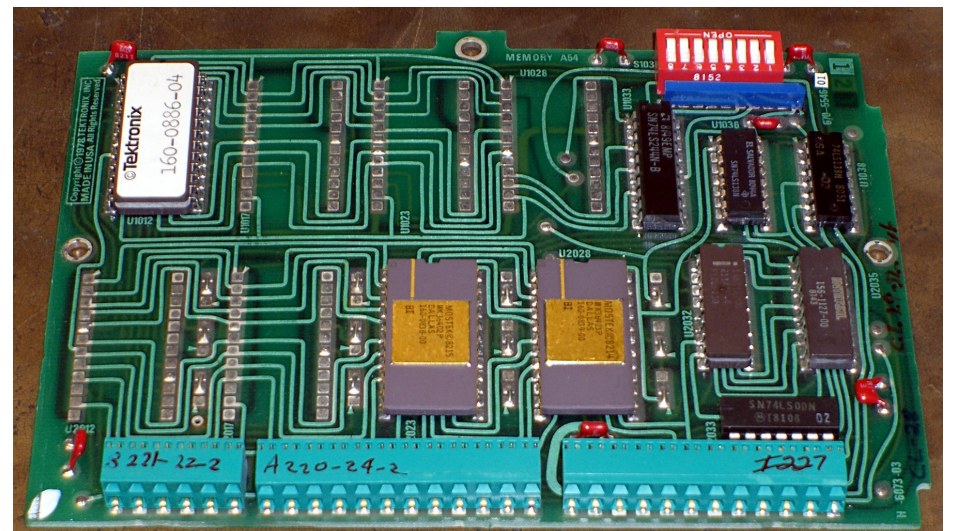
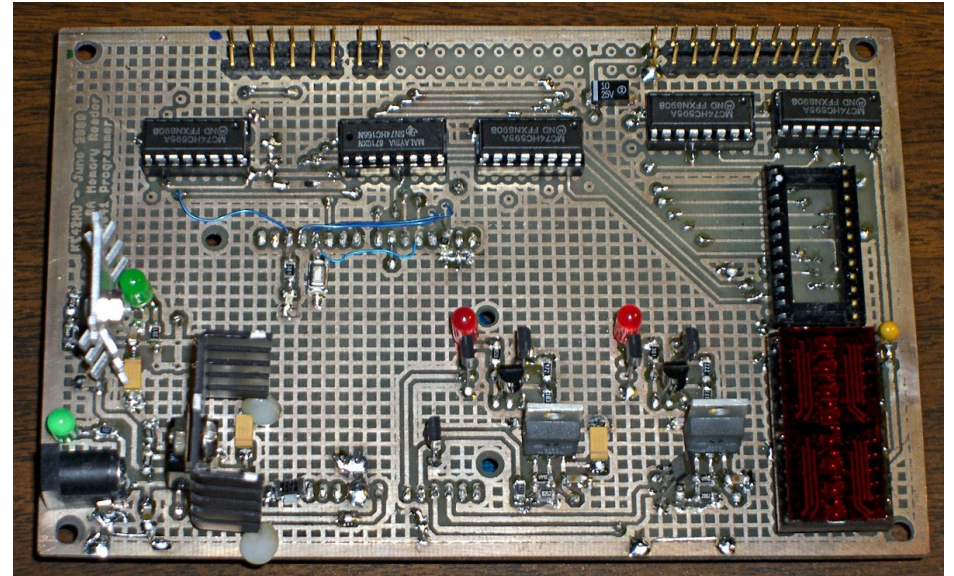
- ◆ Capacitors have R (!)
 - ◆ AC-only property
- ◆ Square-wave drive
 - ◆ Hardware timer
- ◆ Analog input
 - ◆ Hardware ADC
- ◆ LCD output
- ◆ Circuit Cellar Oct 09

4.8 kB



Tek 492 ROM Dump/Burn 7.7 kB

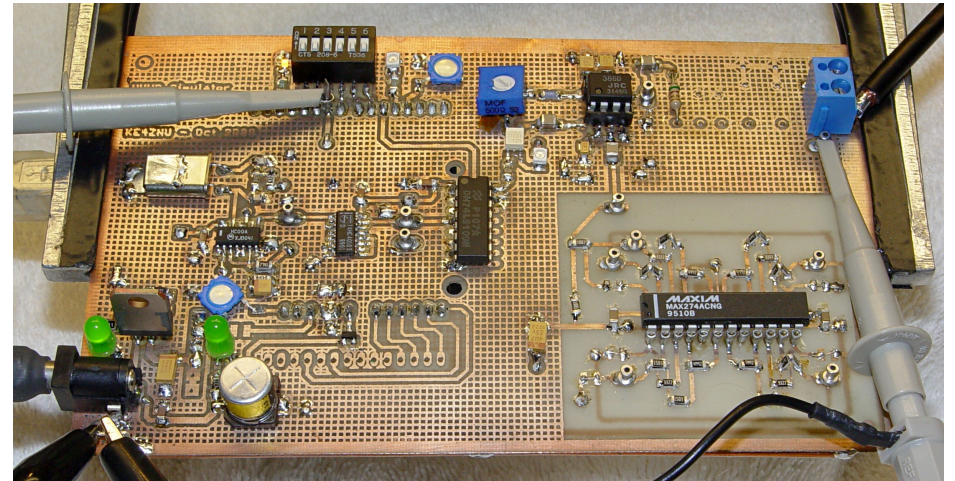
- ◆ Ancient test equipment
 - ◆ ROM bit rot (!)
 - ◆ No commercial value
- ◆ One-off project
 - ◆ Success!
 - From known-good ROM
- ◆ Circuit Cellar Dec 09



WWVB Transmitter Simulator

- WWVB Time Code
 - RF only at night
 - Leap year / second
 - DST/EST changes
- Debugging scaffold
 - Data @ 1 baud (!)
- This. Is. Not. A. Jammer.
 - Really
- Circuit Cellar Feb 10

7.1 kB



Totally Featureless Clock

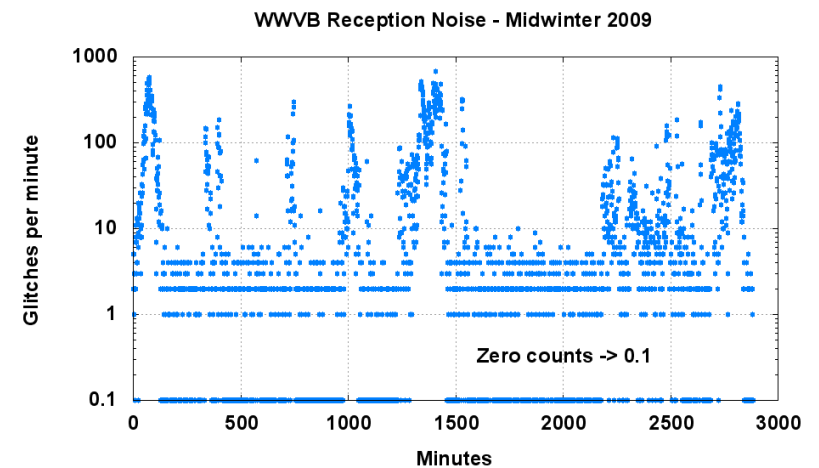
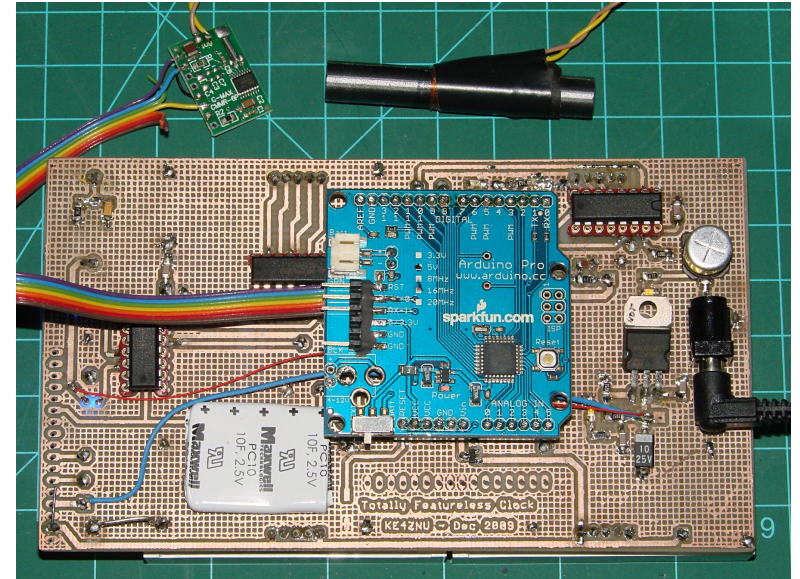
- ♦ All the time
 - ♦ Only the time
 - ♦ Never wrong
 - ♦ No user inputs
- ♦ WWVB parsing
 - ♦ Data @ 1 baud (!)
 - ♦ No error detection bits
- ♦ Trace/logging output
- ♦ Circuit Cellar Apr 10

11.9 kB + 500 B data

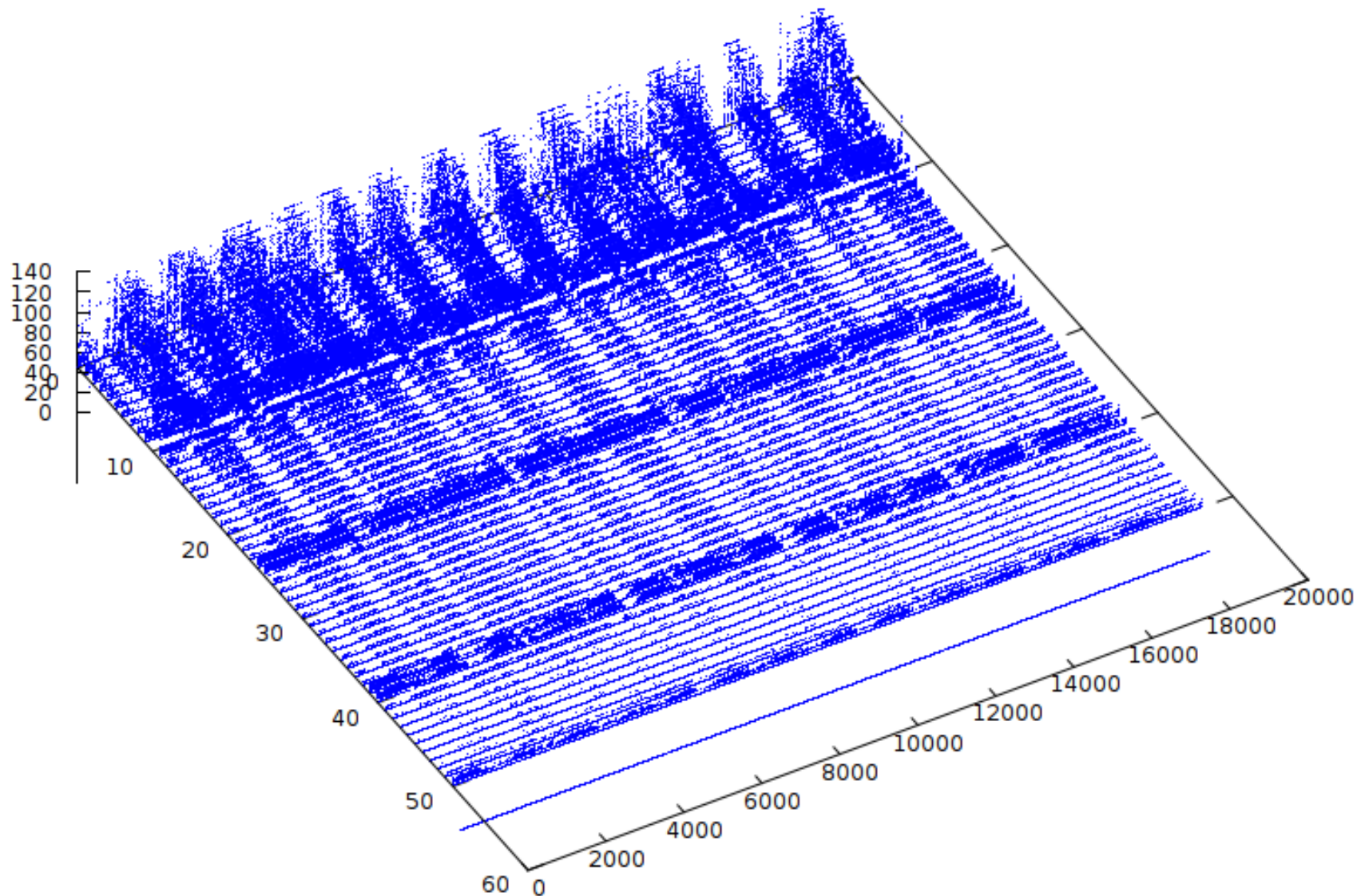


Totally Featureless Clock

- ◆ One input bit
 - ◆ Noisy WWVB Time Code
 - ◆ + a few switch bits
- ◆ One output bit
 - ◆ Serial data → LED digits
 - ◆ + a few debugging outputs
- ◆ Plenty of code in between
 - ◆ Digital Signal Processing
 - ◆ Timekeeping (trivial)



Totally Featureless Clock



Show-n-Tell

Pass around some boards

Admire the listings

Be careful with the clock...

Places To Go

en.wikipedia.org/wiki/Arduino

www.media.mit.edu

www.processing.org

www.wiring.org.co

www.arduino.cc

www.sparkfun.com

softsolder.wordpress.com

Copyright-ish Stuff

Plenty of stuff lifted from Wikipedia
GPL Free Doc License 1.2

Other images probably copyrighted, but
shown & attributed here under “fair use”
[whatever that is]

The rest is my own work



This work is licensed under the
Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License.

To view a copy of this license, visit
<http://creativecommons.org/licenses/by-nc-sa/3.0/us/>
or send a letter to

Creative Commons, 543 Howard Street, 5th Floor
San Francisco, California, 94105, USA.



Ed Nisley

Say “NISS-lee”, although we're the half-essed branch of the tree

Engineer (ex PE), Hardware Hacker, Programmer, Author

The Embedded PC's ISA Bus: Firmware, Gadgets, Practical Tricks

Circuit Cellar www.circuitcellar.com

Firmware Furnace (1988-1996) - Nasty, grubby hardware bashing

Above the Ground Plane (2001 ...) - Analog and RF electronics

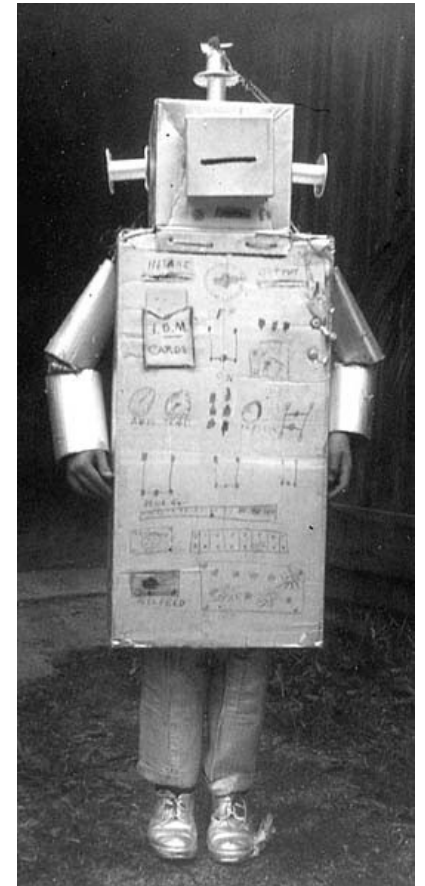
Dr. Dobb's Journal www.ddj.com

Embedded Space (2001-2006) - All things embedded

Nisley's Notebook (2006-2007) - Hardware & software collisions

Digital Machinist www.homeshopmachinist.net

Along the G-Code Way (2008 ...) - G-Code and mathematics



The image features a large black circle centered on a white background. Inside the circle is a dashed black rectangle. The text "If you can't read this then make a new friend 'way up front" is written in blue inside the dashed rectangle. The background is decorated with various colored squares and rectangles, including blue, red, green, yellow, and grey, arranged in a grid-like pattern. A thick black line runs horizontally across the top, and a thick blue line runs horizontally across the bottom. A thick green line runs vertically along the right edge. A thick black line runs vertically along the left edge.

If you
can't read this
then
make a new friend
'way up front