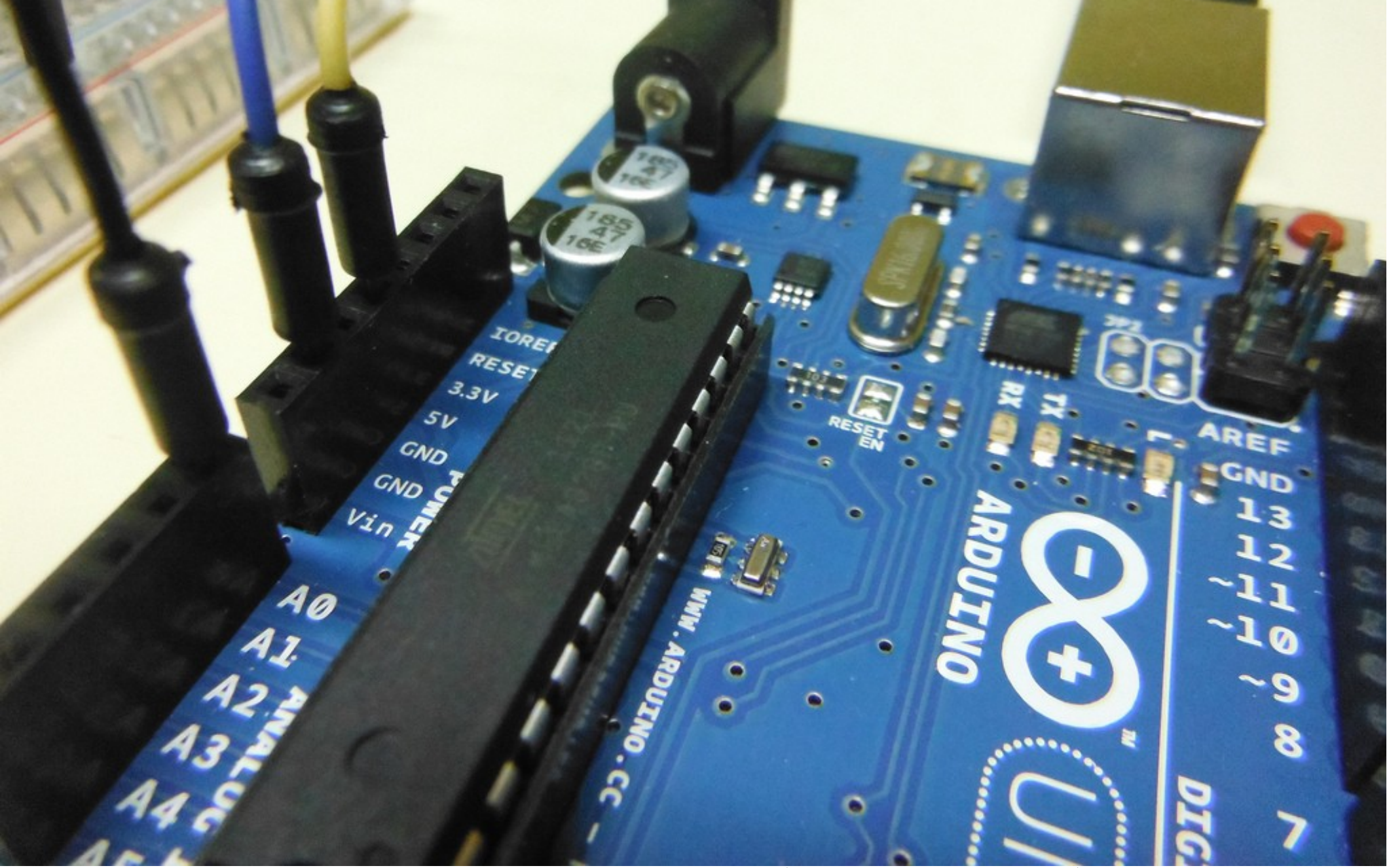


Robert Mibus (@mibus)

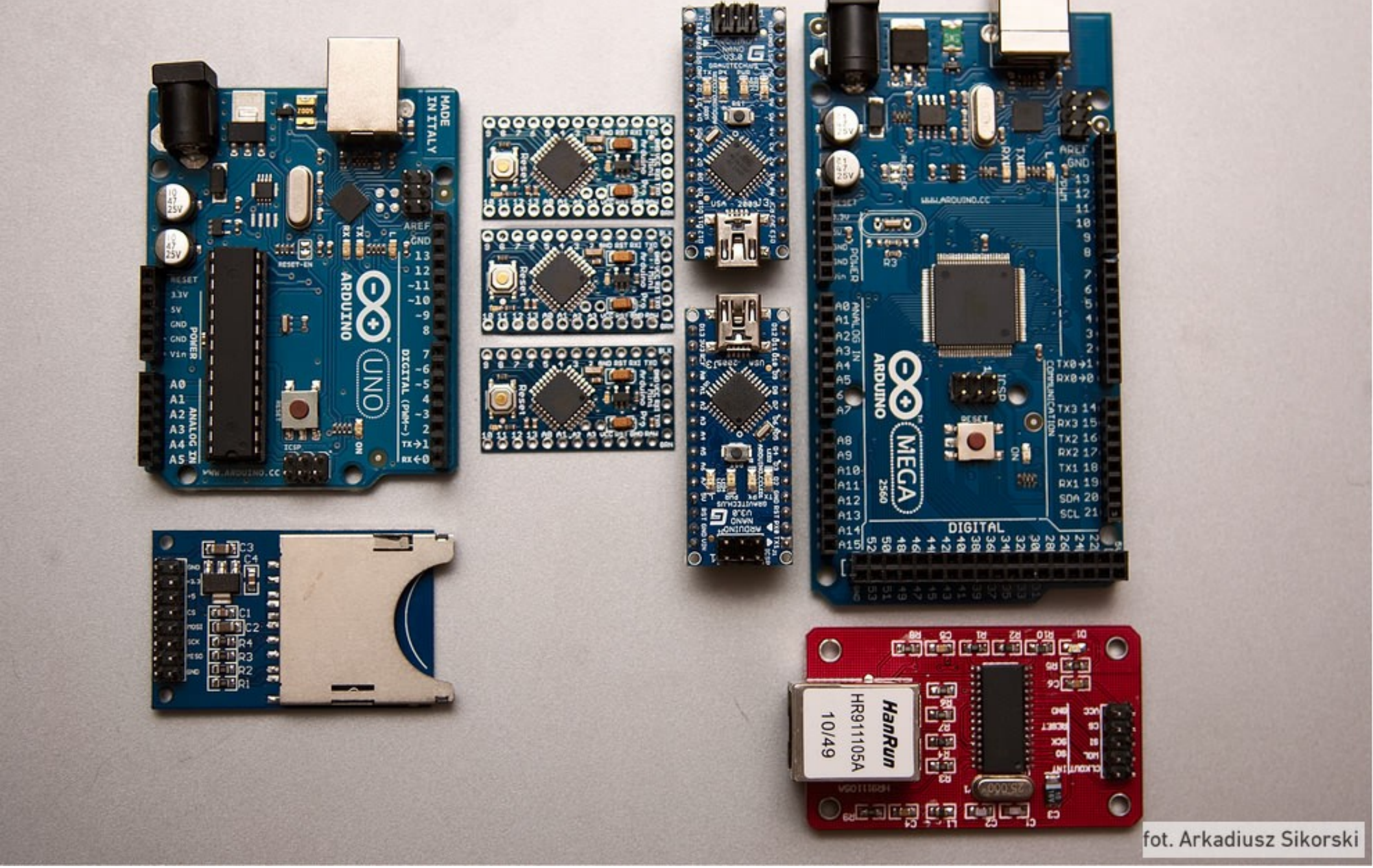
After Arduino

Why?

- Cost
- Features (even in Arduino projects)
- Sheer Nerdiness

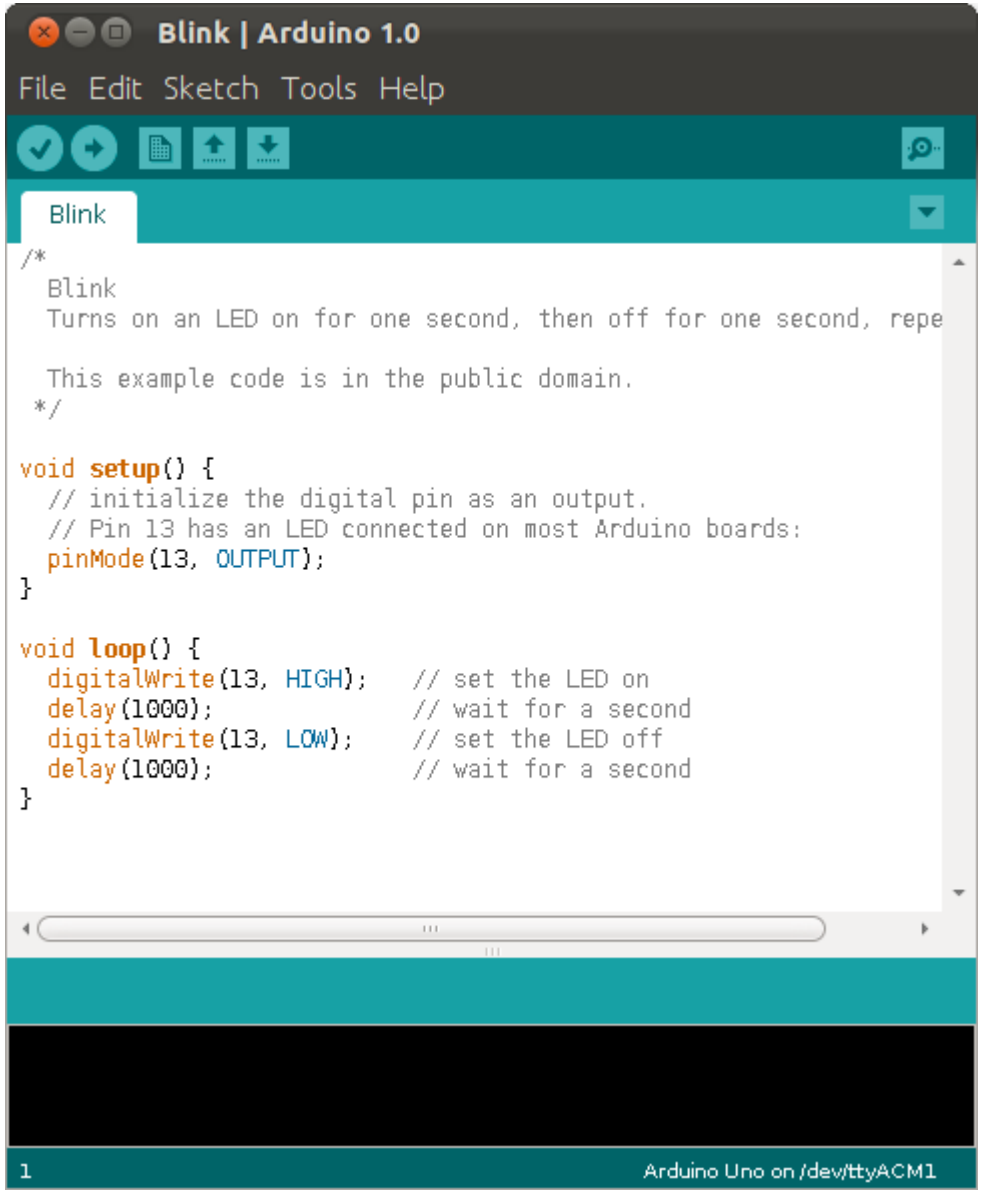
Why not?

- Complexity
- I don't know what I'm talking about

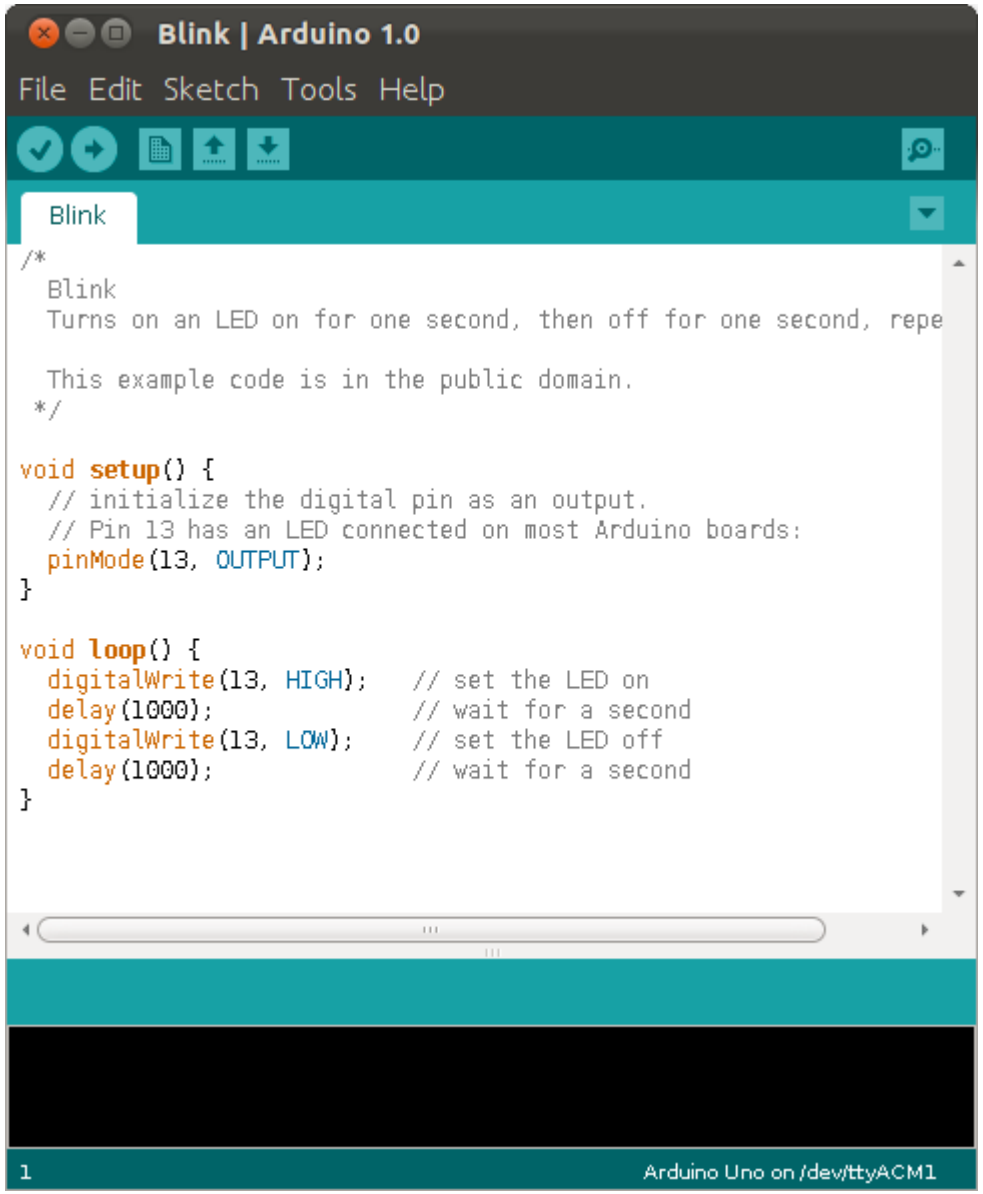


fot. Arkadiusz Sikorski

Arduino is: Hardware



Arduino is: An IDE



Arduino is: Embedded Software (Bootloader, libraries)

Main Site Blog Playground **Forum** Labs Store Help | Sign in or Register

ARDUINO FORUM

Search

Welcome, **Guest**. Please login or register.
January 19, 2013, 08:26:51 PM

| Arduino Forum

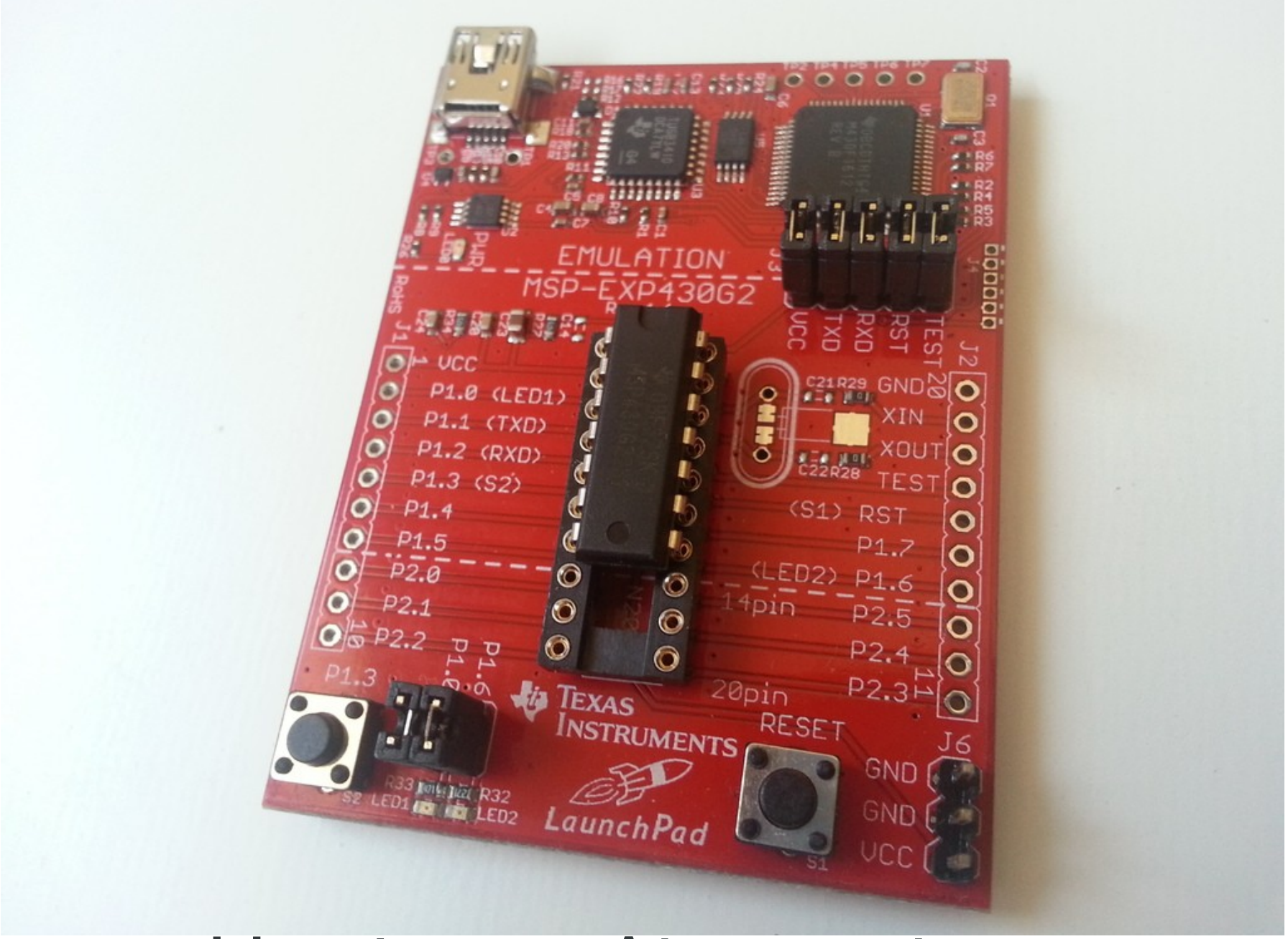
Using Arduino

	Installation & Troubleshooting For problems with Arduino itself, NOT your project **Last post:** Re: Burning a bootloader... by supton on **Today** at 07:51:02 PM	23099 Posts	4698 Topics
	Project Guidance Advice on general approaches or feasibility **Last post:** Re: Need some advice to ... by PeterH on **Today** at 08:22:32 PM	85481 Posts	11456 Topics
	Programming Questions Understanding the language, error messages, etc. **Last post:** Could use some guidance ... by Unhinged_Reprisal on **Today** at 08:19:10 PM	122845 Posts	14345 Topics
	General Electronics Resistors, capacitors, breadboards, soldering, etc.	44459	4901

Arduino is: Community

Hardware

- Easily recognisable
- Development board...

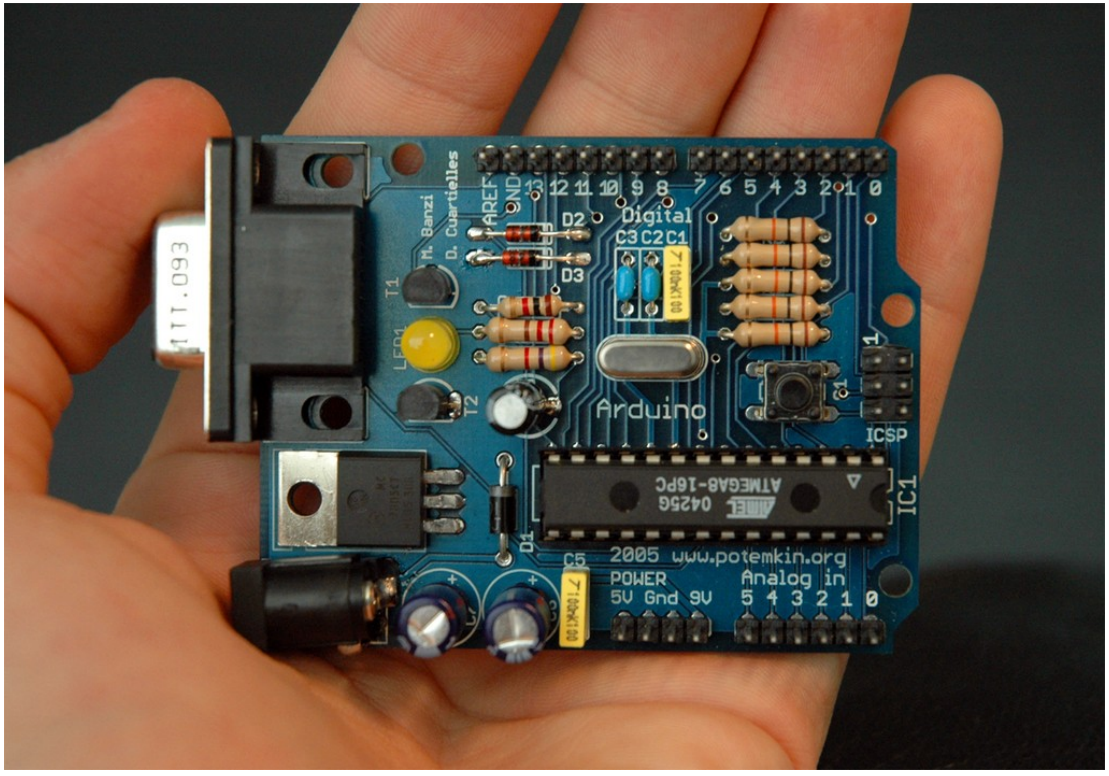


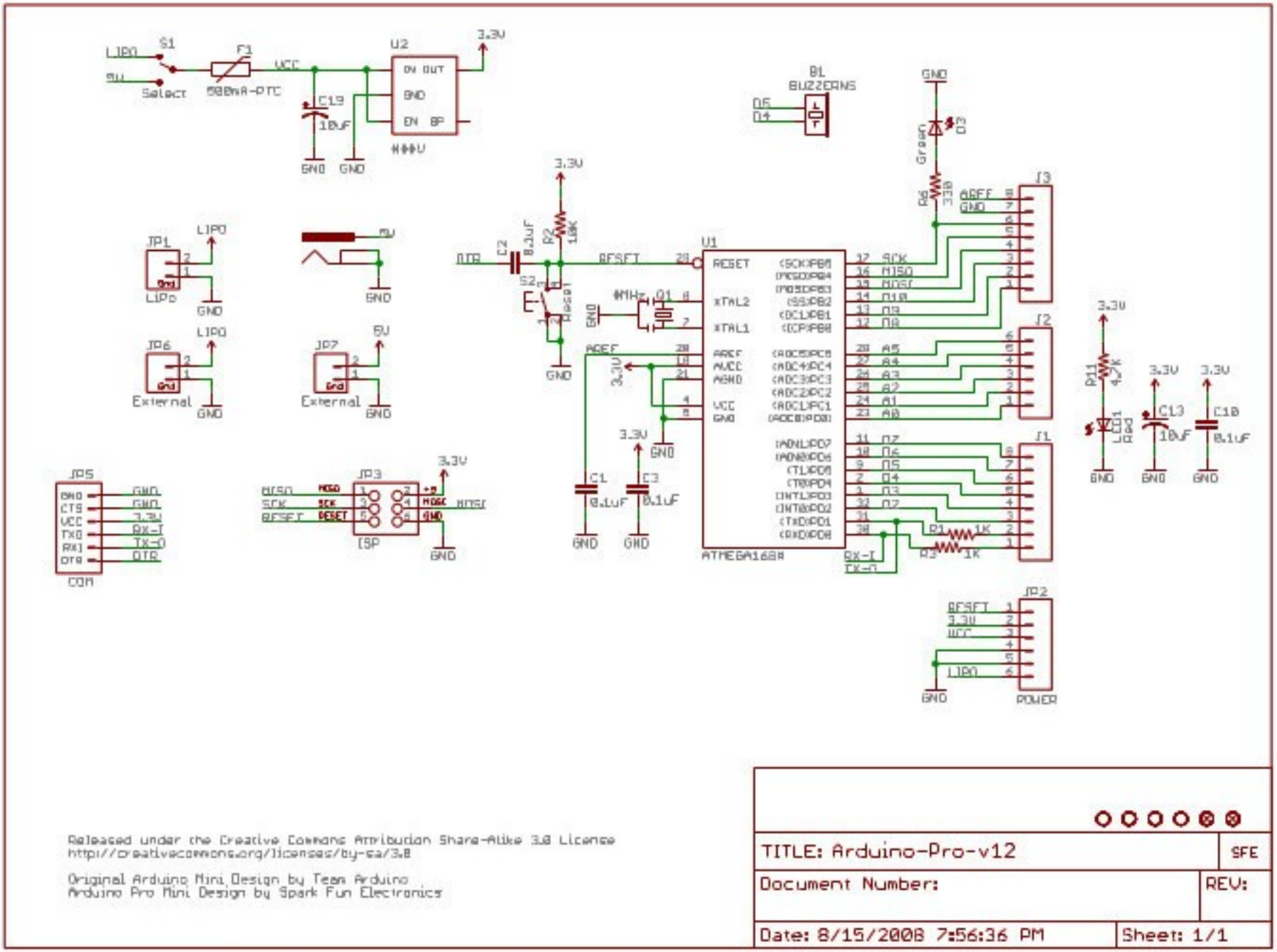
Hardware Alternatives:

eg. TI Launchpad MSP430

Basic Arduino(ish) boards

- Arduino Pro
- Veroduino
- Slowduino
- Breadboard Arduino





But wait, it's open!

Hardware – Components

- ATmega168/328
- Power regulator (pfft)
- Power smoothing capacitors (double pfft)
- LEDs, reset buttons (optional)
- Crystal Oscillator (red “go faster” paint)
- FTDI/USB connectivity (very optional)

Breadboard Arduino

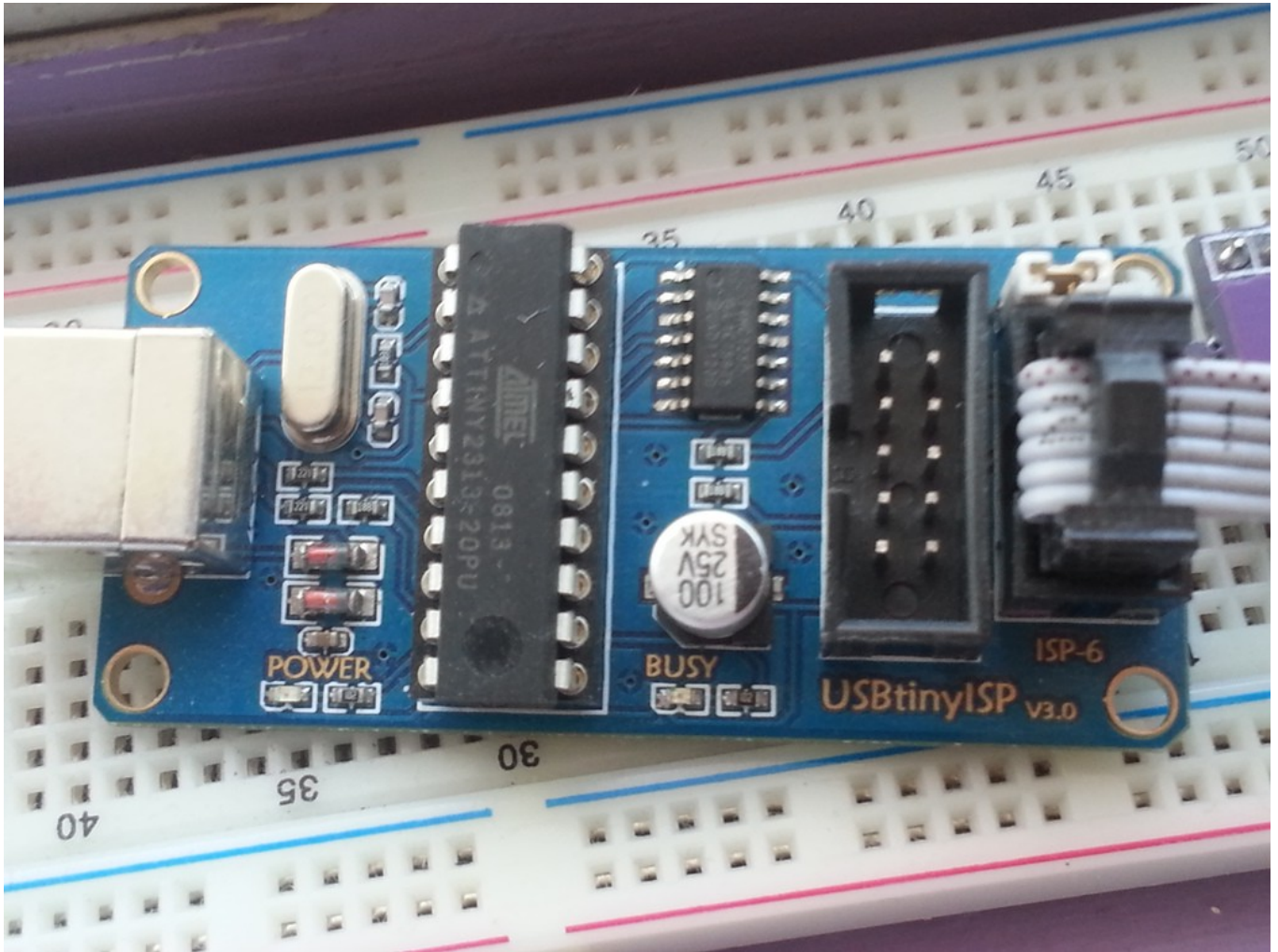
- http://arduino.cc/en/Tutorial/ArduinoToBreadboard
- Also has a “minimal” configuration with just an ATmega328
- Needs custom “device” support in the Arduino software

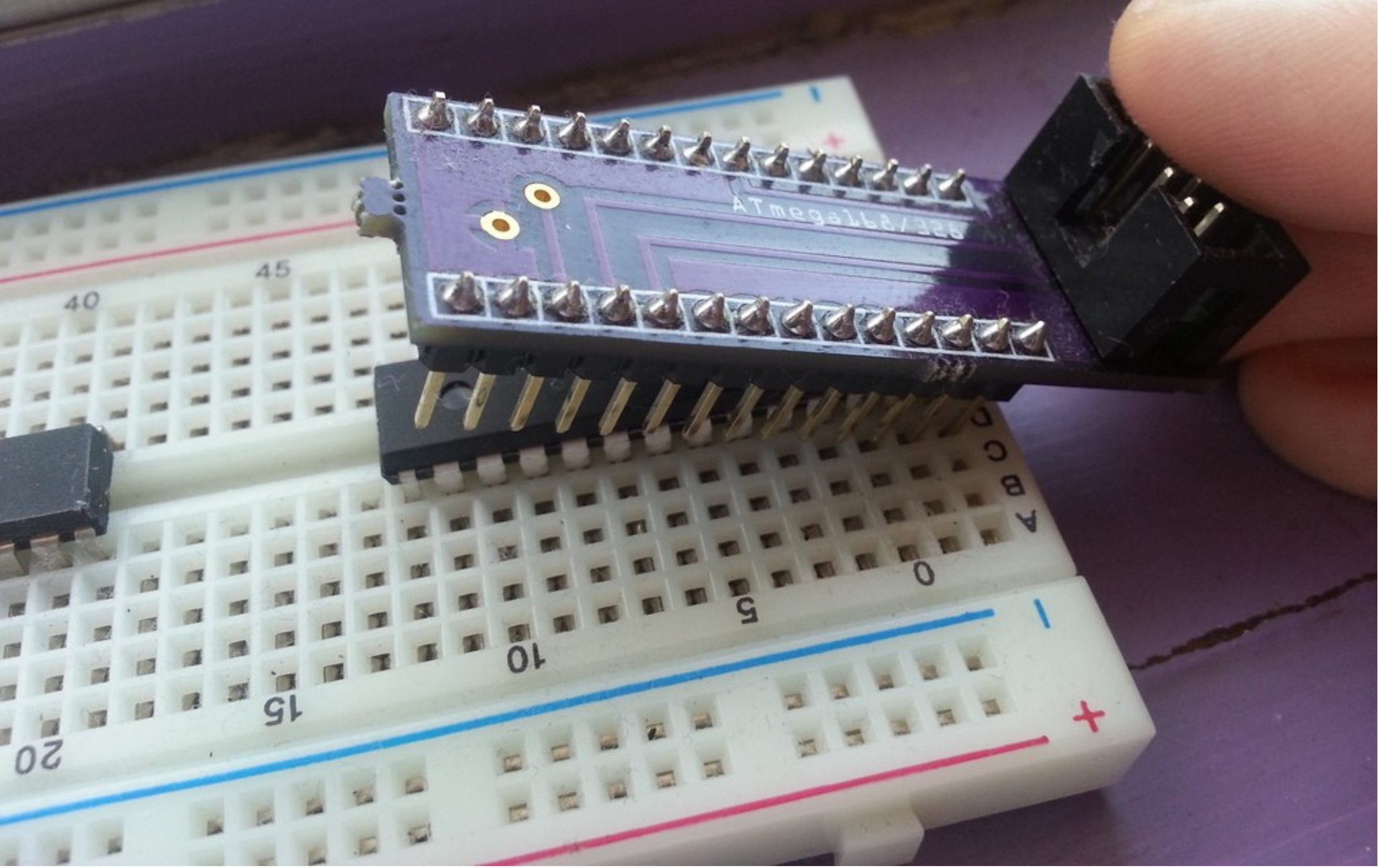
ISP, ICSP, SPI, WTF

- In-System Programming
- In-Circuit Serial Programming
- Serial Peripheral Interface

AVR ISP Devices

- Arduino
- USBtinyISP
- USBasp

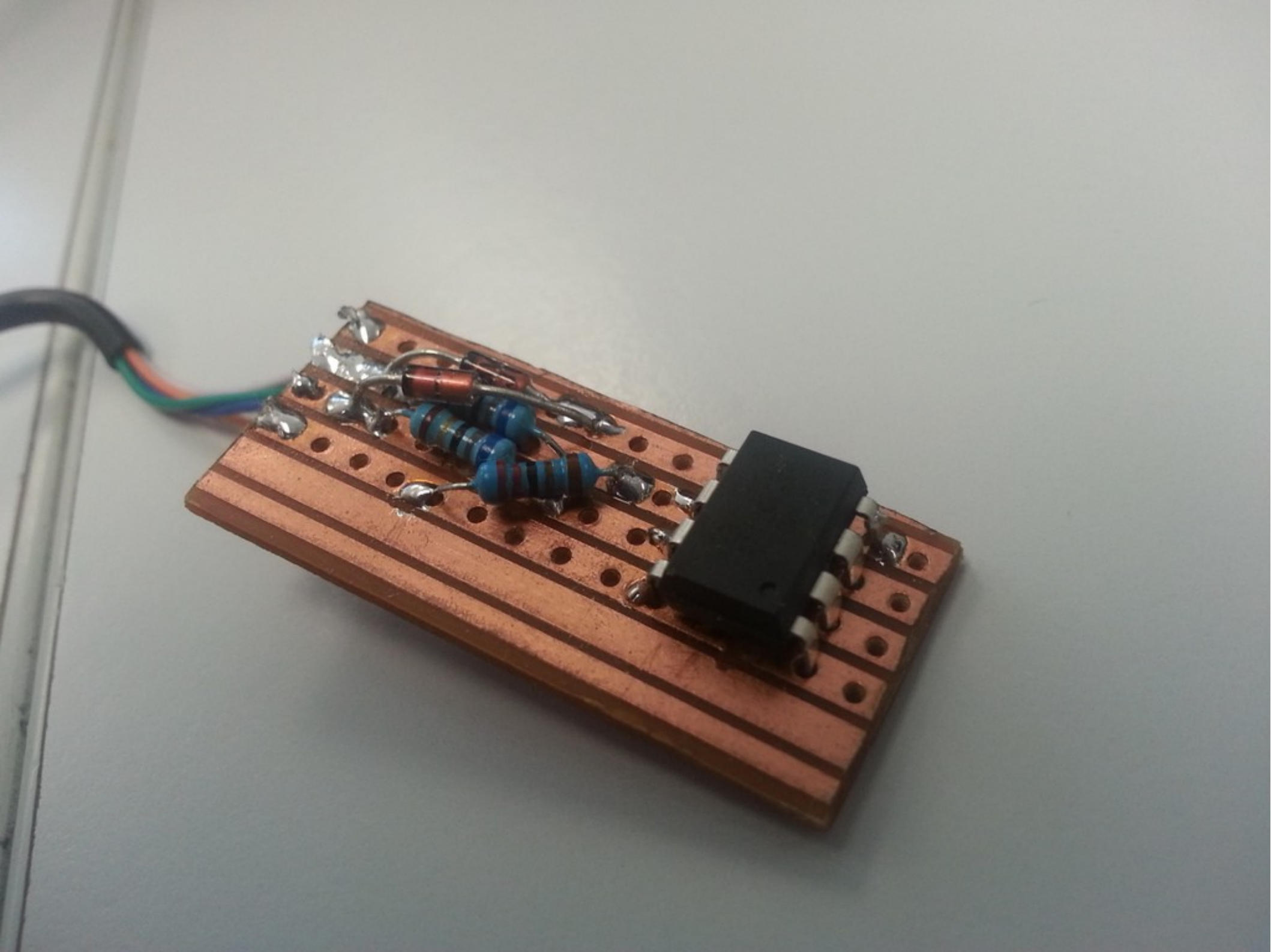




Breadboard ISP adapter

Custom devices, eh?

- ATtiny45/85, ATtiny44/84
- “High-Low Tech” group at MIT
- http://hlt.media.mit.edu/?p=1695
- Burn new fuses (8MHz)
- No bootloader == ISP upload only



Software

- IDE
- gcc
- avrdude
- Embedded bootloader
- Embedded libraries

Software – IDE

- Lacks advanced features
- vim vs emacs.
- (vim is better! :-)

Software – gcc

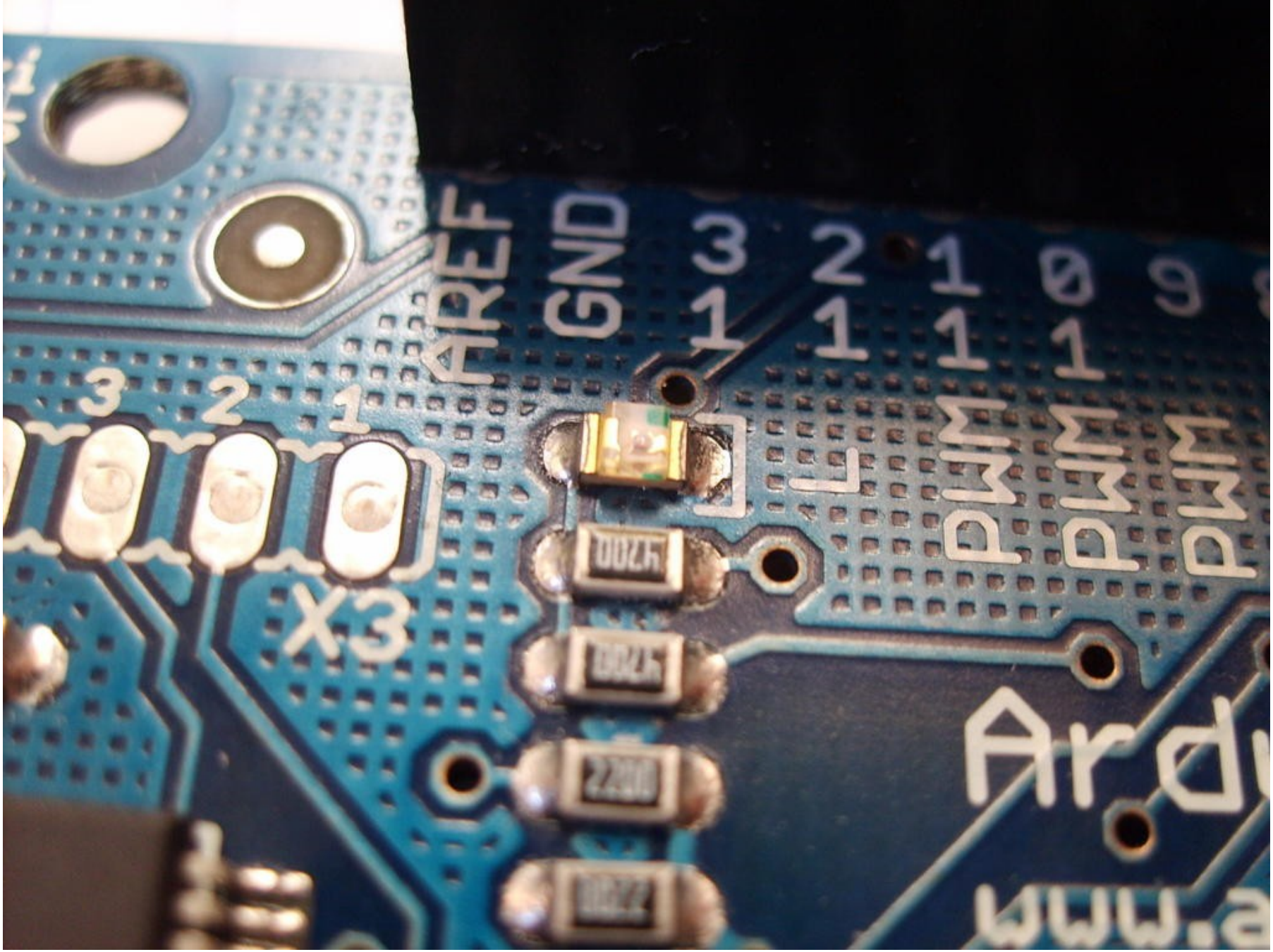
- avr-gcc -g -Os -c -mmcu=attiny13 test.c
- avr-gcc -mmcu=attiny13 test.o -o test.elf
- avr-objcopy -O ihex -R .eeprom test.elf test.hex

Software – avrdude

- Upload via ICSP

Software – Embedded

- Bootloader
- Libraries



Bootloader – Serial Upload

Libraries

- I/O: pinMode() / digitalWrite()
- Interrupts: attachInterrupt()
- Timer: delay() / millis()

9.3 Register Description

9.3.1 MCUCR – MCU Control Register

The External Interrupt Control Register A contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
0x35	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	MCUCR
Read/Write	R	R/W	R/W	R/W	R/W	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 1:0 – ISC0[1:0]: Interrupt Sense Control 0 Bit 1 and Bit 0**

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 9-2. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 9-2. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.

Or: How I learned to stop worrying and love the datasheet.

New Approach

- Same as the old approach
- Selective use of datasheet!
- http://www.atmel.com/devices/atmega328p.aspx?tab=documents

I/O

- Ports – Groups of pins
- Data Direction register
- Input register
- Output register

I/O (includes)

- #include <avr/io.h>

I/O (defines)

- DDRB, PORTB - registers

I/O (example)

- DDRB = 2; // 000010
- PORTB = 2; // 000010
- ((PINB >> 3) & 1) // xx?xxx → 000xx?
  - Is the input from PORTB3
  - Shifted by 3 results in binary value ending in 0 or 1
  - Masked by 1, making it 0 or 1
- Not directly aligned to “Arduino” pin numbers

.....

I/O (defines #2)

- DDB0, DDB1, etc. - bit # in register
- PB0, PB1, etc. - bit # in register

I/O (example #2)

- DDRB = (1 << DDB1);
- PORTB = (1 << PB1);
- ((PINB >> PB3) & 1);

I/O (benefits)

- Can get/set all pins on a port faster
- Can use otherwise-dedicated pins (ie. Crystal)

Interrupts!

poke

9.3 Register Description

9.3.1 MCUCR – MCU Control Register

The External Interrupt Control Register A contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
0x35	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	MCUCR
Read/Write	R	R/W	R/W	R/W	R/W	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 1:0 – ISC0[1:0]: Interrupt Sense Control 0 Bit 1 and Bit 0**

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 9-2. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 9-2. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.

Interrupts

- PCINT – Pin Change Interrupt
- TIMER0_OVF – Timer0 Overflow
- TIMER1_COMPA – Timer1 Compare Match A
- Stored in code as a series of jump instructions

PCINTerrupts

- GIMSK – General Interrupt Mask
  - PCIE – Pin Change Interrupt Enable flag
- PCICR – Pin Change Interrupt Control Register
  - Enables interrupts on a specific Port
  - Different Port → Different ISR
- PCMSK – Pin Change Mask
  - Enables interrupts on a specific pin

Interrupts – Special Macros

- `#include <avr/interrupt.h>`
- `sei() / cli()`
- `ISR(INTERRUPTNAME_vect) { … };`

Interrupts pro/con

- Pro: Can do all pins, not just some
- Con: Can only do “change”, not rising/falling/eaten-by-a-grue, etc.

Timers

Timers

- More than just delay() and millis()
- Complicated :-(
- Responsible for PWM!
- Can be a counter for an external clock

Timers (cont)

- An counter (8-bit or 16-bit)
- Can trigger interrupt for overflow, or at a fixed value.
- Can trigger changes on certain pins (PWM!)
- Different counting modes
- Tight control over duty cycles

Timer modes

- From ATtiny85
- Normal
  - Repeatedly counts from 0 to overflow.
- CTC (Clear Timer on Compare)
  - Repeatedly counts from 0 to OCR0A
- Fast PWM
  - Set/unsets output at OCR0A and overflow
- Phase-correct PWM
  - Counts down rather than overflowing

Timer registers

- TCCR0A, TCCR0B – Timer/Counter 0, Control Register A/B
- Waveform generation
- Output controls
- Clock selection / prescaler
- TCNT0 – Counter
- OCR0A – Comparison value
- TIMSK – Timer Interrupt Mask

Tying it together

Sample Code Goes Here

```
// Needed for PORTB and friends
#include <avr/io.h>

// Needed for sei() etc.
#include <avr/interrupt.h>
```

```
int main (void) {
  // Set up Data Direction Register for PORT B
  // These are pins 15 & 16, aka PB1 & PB2
  DDRB = (1<<DDB1) | (1<<DDB2);
-
  PORTB = 0; // Wipe all output of pins on PORT B

  setup_timer(); // My function to enable a timer

  while (1) {} // Infinitely Awesome Loop
}
```

```
int setup_timer (void) {
  // cli();

  // "Normal" waveform generation, and no
  // output/comparison modes
  TCCR1A = 0;

  // No waveform options, clock prescaler of 64
  TCCR1B = (1<<CS11) | (1<<CS10);

  // Set no extra options.
  TCCR1C = 0;

  // Enable interrupt on overflow
  TIMSK1 = (1<<TOIE1);

  sei();
}
```

```
ISR(TIMER1_OVF_vect) {
  // Toggle PORTB pin 1's output
  PORTB ^= (1<<PB1);
}
```

```
MCU=atmega328p
AVRDUDE_MCU=m328p

all: lca2013-mibus.hex

clean:
	rm lca2013-mibus.hex
flash: lca2013-mibus.hex
	avrdude -c usbtiny -p ${AVRDUDE_MCU} -U flash:w:lca2013-mibus.hex

%.o: %.c
	avr-gcc -g -Os -c -mmcu=${MCU} $*.c
%.elf: %.o
	avr-gcc -mmcu=${MCU} $+ -o $@
%.hex: %.elf
	avr-objcopy -O ihex -R .eeprom $+ $@
```

Community

- Arduino official Forum
- Blogs (Hack A Day etc)
- Us
- All still here!

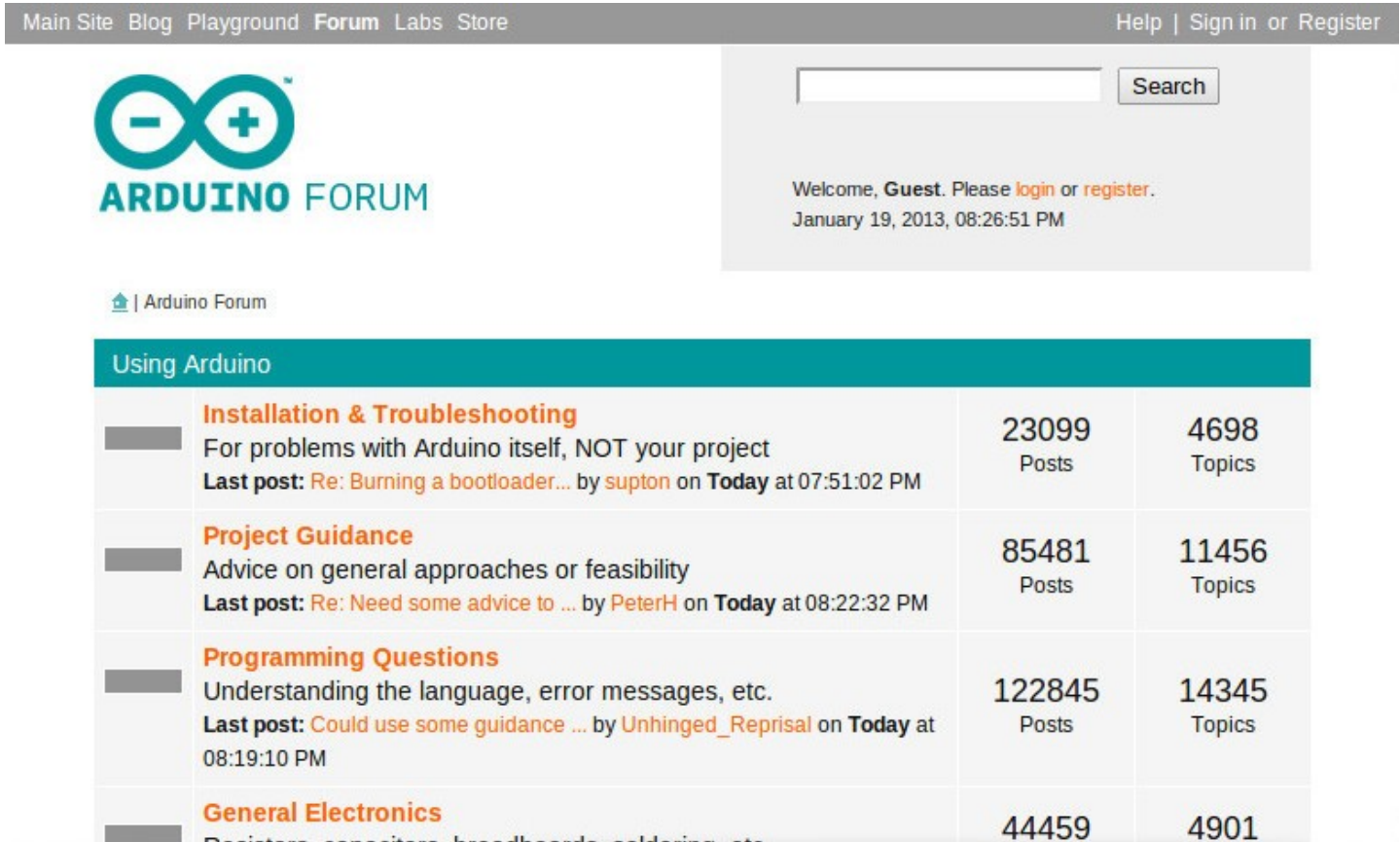


Photo Credits

- http://www.flickr.com/photos/dabidmartinez/8234672297/
- http://www.flickr.com/photos/arakus/8077218145/
- http://en.wikipedia.org/wiki/File:Arduino_1.0_IDE,_Ubuntu_11.10.png
- http://en.wikipedia.org/wiki/File:Arduino316.jpg
- http://en.wikipedia.org/wiki/File:Arduino_led-5.jpg
- http://www.flickr.com/photos/wjlonien/5979084230/
- http://www.flickr.com/photos/jgbarah/3393168616/
- http://www.flickr.com/photos/salim/42670013/

After “After Arduino” by @mibus

- Arduino is:
  - Hardware (ATmega328 “etc”)
  - Software (IDE, gcc, avrdude)
  - Community (you!)
- You can:
  - Use less (or different) hardware
  - Use leaner software
  - Still be part of a “hacker” community
- Code and PCB layouts at:
  - http://github.com/mibus/AfterArduino/