"Arduino" without the Arduino

Tips and Tricks from a Neanderthal

Elliot Williams
elliot@elliotwilliams.org
hackaday: elliotwilliams@hackaday.com
github & twitter: hexagon5un

May 12, 2015

Outline

Outline

I come from the time before Arduino

Biases

- C-centric
- Small programs
- Love embedding logic in stuff
- Hardware-near

Perspective

- AVR chips for 12+ years
- Cool stuff

This Talk

- Outsider's perspective on Arduino
- Tips and tricks from outside the box
- Some peeking behind the curtain
- Doing "Arduino" without Arduino
- Doing Arduino without "Arduino"

Survey

- How many of you have used other microcontrollers?
- Other languages?
- Other development environments?
- Other HALs for micros? (CMSIS for ARM? XMega?)

What is Arduino?

- What do you think of when I say "Arduino"?
- IDE
- Hardware
- "Language" / Libraries
- (The Community)
- Now let's take them all apart!

Outline

What's the Arduino IDE Doing?

Let's Peek

- Turn verbose logging on
- Compile something
- Have a look

"Too much"

- Compiles everything "core" into `core.a` library
- Only really need to recompile "core" when it changes
  Recompiling every time is conservative / safe

"Not Much"

- Just rebuilding core library, linking our code with it, uploading
- `http://www.arduino.cc/en/Hacking/BuildProcess`

Can we do without the IDE?

Yes.

- First the brutal way: reproduce the log file
  build up the `core.a` static library, link against it
- Need to add in anything outside the "core" manually:
  copy code into directory
- Hardware platform dependent files: `pins_arduino.h`
- One should write up a nice Makefile for this

Why would you want to?

- Automation?
- Thirst for knowledge?

Outline

Arduino "Language"

There is no Arduino Language

- It's really C(++)
- The `.ino` files get an include file tacked on, some prototypes declared
- Comments deleted (why?)
- Trivial stuff (why?)
- The real secret sauce is in the libraries

Blink.ino

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

blinkLED.c

```
// ------- Preamble -------- //
#include <avr/io.h>                        /* Defines pins, ports, etc */
#include <util/delay.h>                     /* Functions to waste time */

int main(void) {
  // -------- Inits --------- //
  DDRB |= (1 << PB5);                           /* Enable output on LED */

  // ------ Event loop ------ //
  while (1) {

    PORTB |= (1 << PB5);                              /* Turn on LED pin */
    _delay_ms(1000);                                            /* wait */

    PORTB &= ~(1 << PB5);                               /* Turn off LED */
    _delay_ms(1000);                                            /* wait */

  }                                                   /* End event loop */
  return 0;                                  /* This line is never reached */
}
```

Main.cpp

```
#include <Arduino.h>

//Declared weak in Arduino.h to allow user redefinitions.
int atexit(void (*func)()) { return 0; }

// Weak empty variant initialization function.
// May be redefined by variant files.
void initVariant() __attribute__((weak));
void initVariant() { }

int main(void) {
        init();
        initVariant();
#if defined(USBCON)
        USBDevice.attach();
#endif
        setup();
        for (;;) {
                loop();
                if (serialEventRun) serialEventRun();
        }
        return 0;
}
```

Big Code

Blink in C and .ino

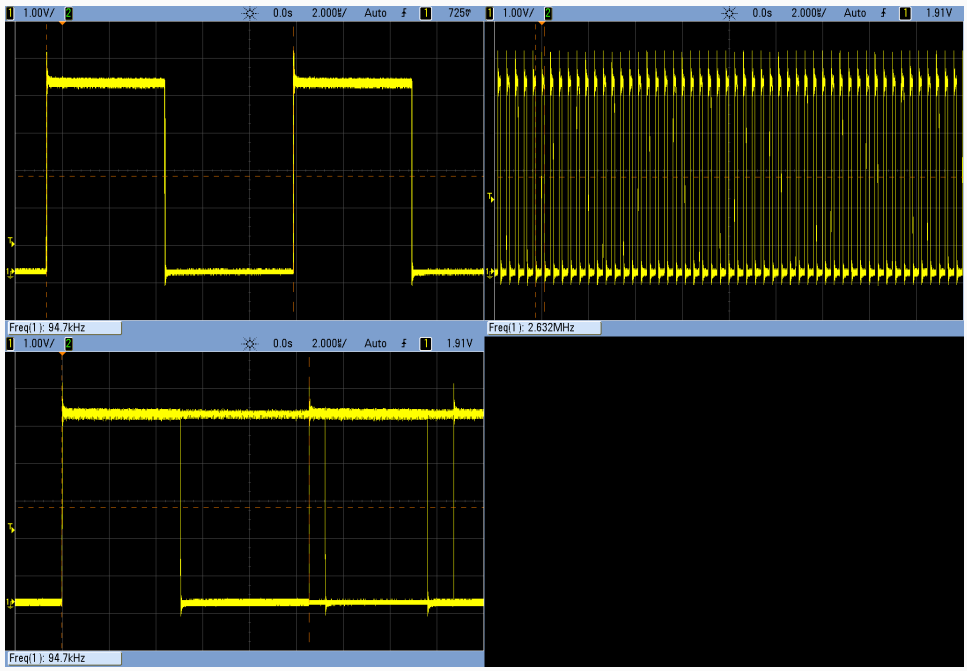
- `Blink.ino` vs `blinkLED.c`
- 1030 bytes vs 178 bytes
- Arduino has a lot of bells and whistles
- milliseconds timer, for instance, is great
- see `init()` in `wiring.c`
- But what if you need it small?
- Edit `Main.cpp` by hand, removing `init()` or write your own `main()`...

Slow Code

Direct Pin Access

- `digitalWrite()` is basically horrible.
- Pin-toggling race
- `digitalWrite()`: 95 kHz
  Set/clear pins in C: 2.27 MHz
  Optimized C: 3.85 MHz
- 24-40x faster to access pins directly
- 4, 7, or 168 clock cycles
- Why? Arduino decodes the pins for you
  there's `if()` statements running in real time

The Shootout



The Gain

Why would they code up `digitalWrite()`?

- AVR (all 8-bit chips?) arrange pins in banks of 8.
- You need to know which bank the pin is in
- That's why AVR pins named like "PB3" and "PC0"
- It's a minor hassle.
- Is it worth 160 clock cycles?

Bit Twiddling

They really slowed down bit manipulations by 24x for that?

- Cross-hardware compatibility
- The real hassle is bit manipulation in C
- `PORTB &= ~(1 << PB5);`
- That's hard to read at first
- It's an old idiom in microcontroller C
- But it's completely avoidable

Arduino.h

```
/*
  Arduino.h - Main include file for the Arduino SDK
  ... snip ...
*/

#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <math.h>

#include <avr/pgmspace.h>
#include <avr/io.h>
#include <avr/interrupt.h>

#define bitRead(value, bit) (((value) >> (bit)) & 0x01)
#define bitSet(value, bit) ((value) |= (1UL << (bit)))
#define bitClear(value, bit) ((value) &= ~(1UL << (bit)))
#define bitWrite(value, bit, bitvalue) (bitvalue ? bitSet(value, bit) : bitClear(value, bit))
```

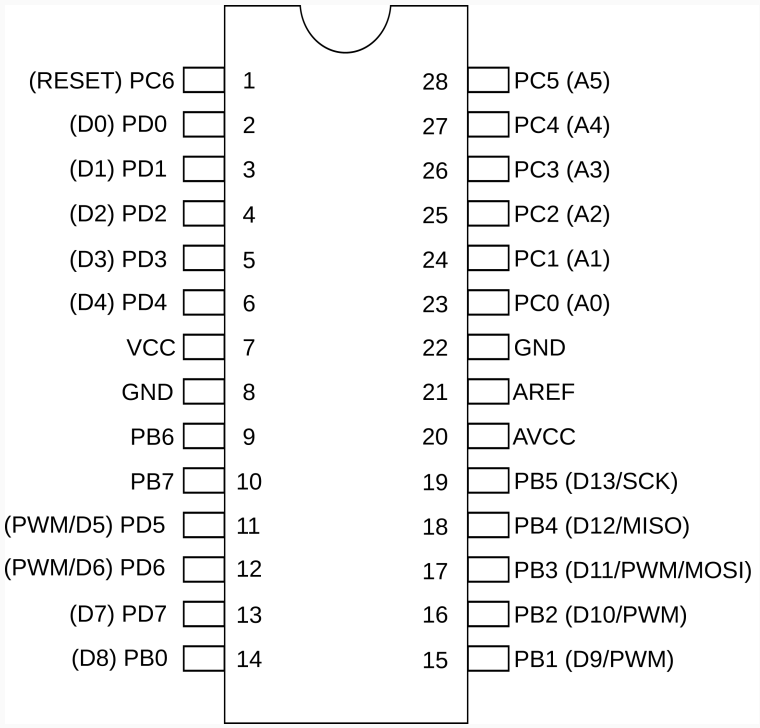
BlinkDirect.ino

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  // digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  bitSet(PORTB, PB5);
  delay(1000);                 // wait for a second
  // digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  bitClear(PORTB, PB5);
  delay(1000);                 // wait for a second
}
```

Pin Lookup



Forsaking the Sweet Arduino Libraries

What do you gain by writing lower-level code?

- More control over timings
- Manage resource conflicts
- Memory efficient, faster

But what did people do before Arduino?

- AVR standard libs: `http://www.nongnu.org/avr-libc/`
- USART: `http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/Der_UART`
- Copy your own previous work
- Menu-driven code generator: `http://www.dioda.ro/avrwiz/index.html`

Write C in Arduino IDE

- What's with all this `.ino` stuff anyway?
- You're already writing C/C++
- Just ignore the `.ino` file.
- Procedure:
  Open up a blank project.
  "Save as" it.
  Open up `whatever.c` and start coding.
  Compile, profit.

Live Demo

Outline

What's in an Arduino?

Arduino Uno: When life was simple

- AVR microcontroller
- Serial/USB converter
- voltage regulator
- crystals
- capacitors
- reset button
- LEDs

"Arduino" on Other Hardware

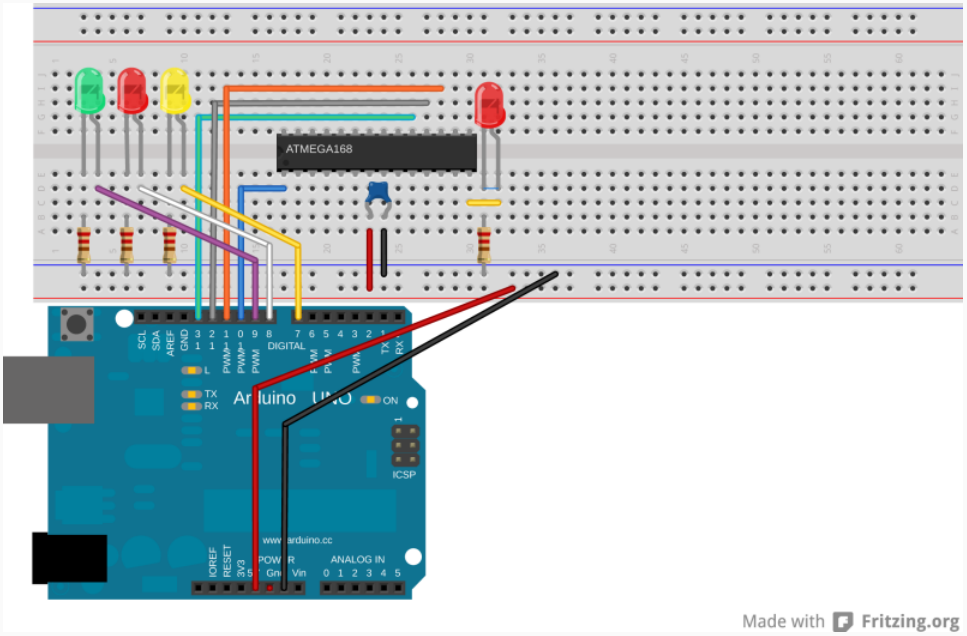
- The community has been doing this for a while:
  bare AVR ATmega chips
  ports to ATtiny85 (Gemma, Digispark, etc)
- Chinese Clones, *duinos, PJRC Teensy
- Arduino Due: Arduino on an ARM platform
- STM32 Nucleo (ARM)
- Intel Galileo
- There's really no hacking to be done here.

Arduino on Bare AVR

But how to get the code in?

- Turn your Arduino into an AVR programmer
- Flash ArduinoISP project
- Hookup to breadboard
- Tools → Programmer → Arduino as ISP
- Tools → Board → Arduino Pro Mini (8MHz)
- *Shift-click* Upload

ArduinoISP Demo



Or Go Completely Rogue

Drop the Hardware, IDE, and Libraries:

- Write your code using whatever editor you'd like
- Compile directly (Makefile for help)
- upload using ArduinoISP outside of the IDE
- `avrdude -p m168p -c avrisp -b 19200 -P /dev/ttyACM0`
  or `COM5` or `/dev/tty.usbserialxxxxxxx` or whatever

Wrapup

- "Writing in Arduino is like knitting with boxing gloves on."
- If you just want control over the hardware,
  the Arduino ecosystem can get in your way.
- But you can take the gloves off
- Or convert them into fingerless boxing gloves
- Ditch the IDE,
  ditch the libraries in whole or in part,
  ditch the hardware.

The End

Outline