Step *Away* from the Arduino!

Dr. Eric Ayars

Professor of Physics
Official Advanced Lab Guy
Unofficial Department Hacker
Curmudgeon in Training

California State University, Chico

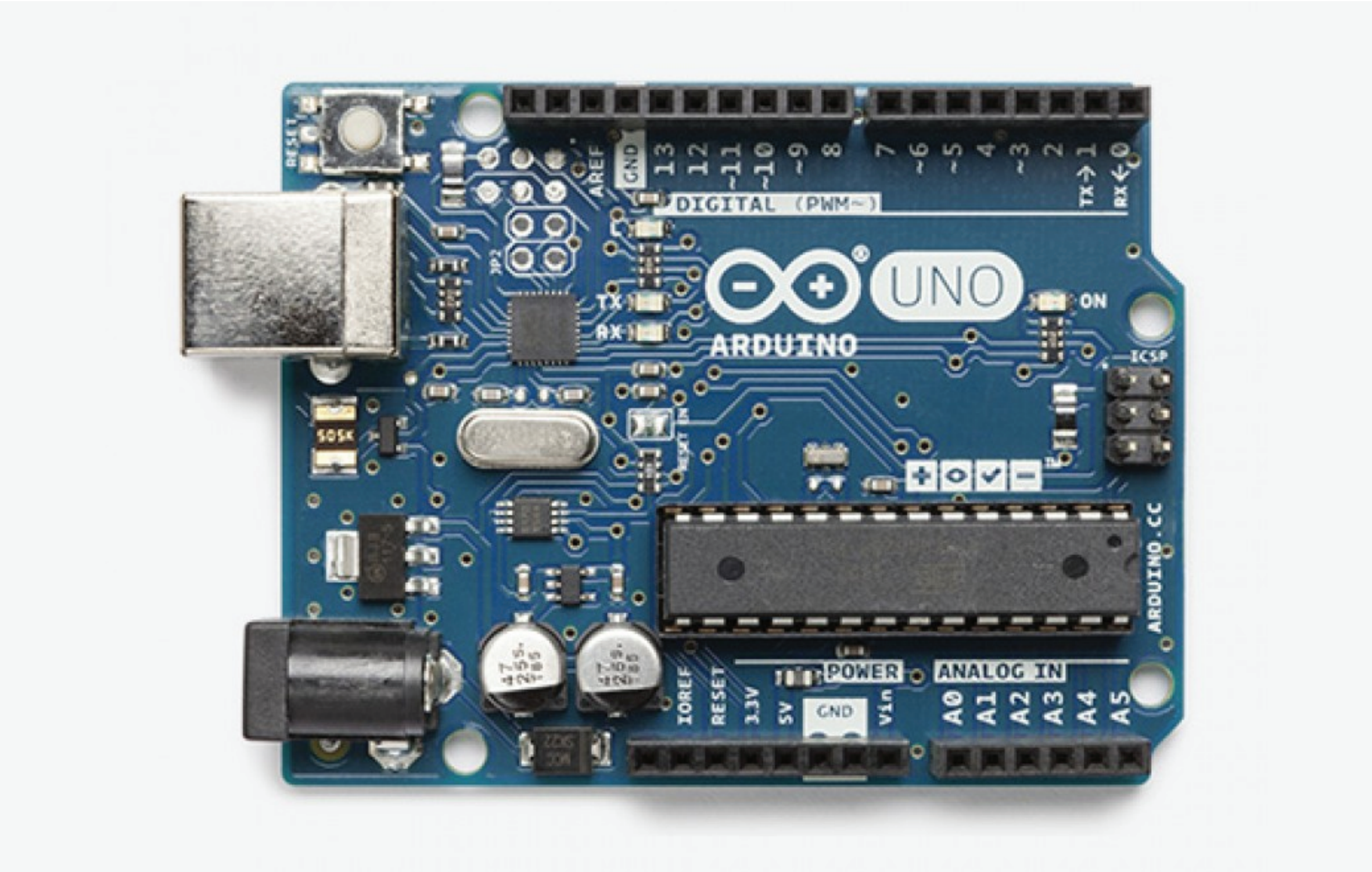
On the one hand, Arduino has done more to get people involved in electronics than any organization since Heathkit. Artists, inventors, grade-school children, and yes, even physics students and instructors; all varieties of amateurs have benefitted from the Arduino development platform. This is a *good thing*, and it's why I'm a fan.

On the other hand, if you're using a $24 Arduino Uno to turn on an LED when your experiment temperature gets too high, you're doing it wrong. And if you're using an 8-bit microcontroller with 32k of program memory to synthesize a drive waveform while tracking a high-resolution quadrature encoder and sending position/velocity data to a computer, you're unlikely to be doing it at all.

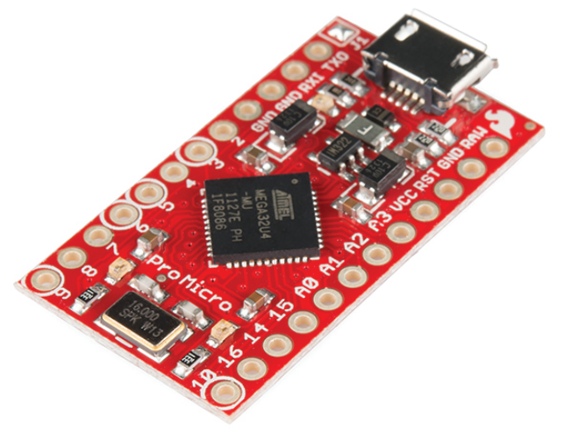
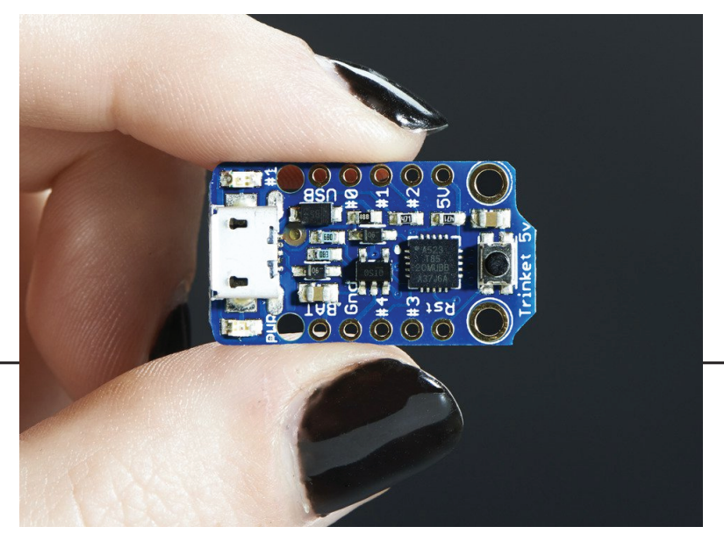
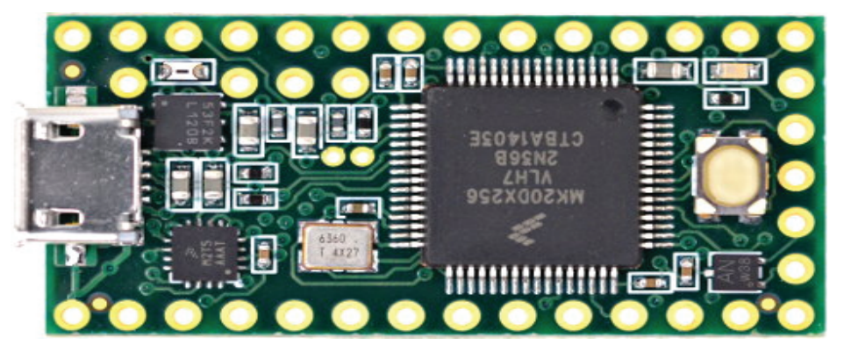
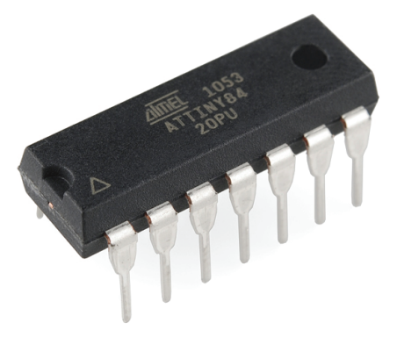
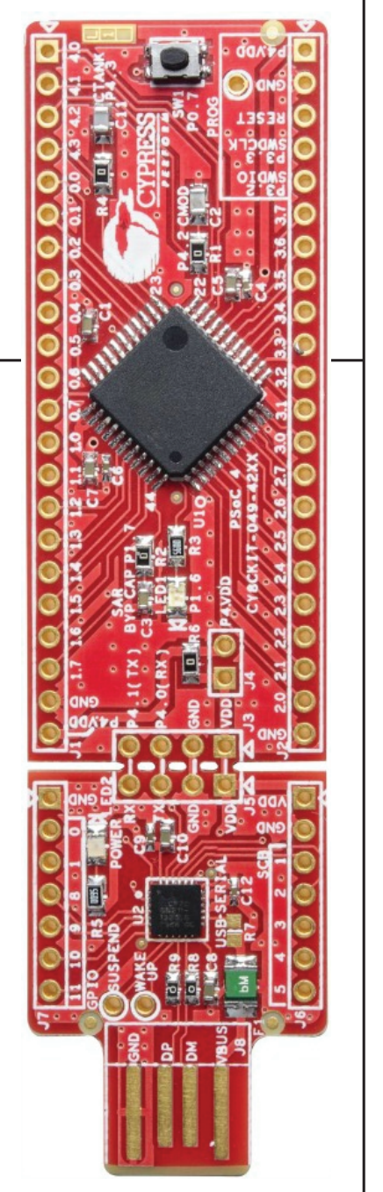
So here are some other options. Most use the Arduino Integrated Development Environment (IDE), so programming them is the same as programming an Arduino Uno. All of them are breadboard compatible, so they can be easily incorporated into projects either temporarily or permanently. And all of them are less expensive than an Arduino Uno.

Here's the baseline: the Arduino Uno.

32k program memory
16MHz 8-bit processor
Built-in serial communications via USB
6 10-bit A/D inputs (using one A/D converter)
14 digital input/output lines
    6 of which can be used for "analog" PWM output
On-board power management
    runs on 5-15V DC
$24. (Cheaper if you buy Chinese clones, which work about 3/4 of the time.)



Easier/simpler replacements ← → **More complicated/powerful replacements**

	Arduino Pro Micro	Trinket	Teensy 3.2	Bare Atmel Chips	Cypress PSoC
What	An Arduino Uno in small breadboard-friendly format, $20. 	A tiny Arduino-compatible device with just a few i/o ports and not much memory, $7 	Arduino on steroids, $20 	Just the microcontroller, add your own bells and whistles. $1 - $3 	Programmable System on Chip: Analog & digital & FPGA logic & op-amps & DACs & ADCs and a Cortex MCU all on one device, $4. 
Why	• Nearly a direct Uno replacement • Easy to incorporate into a breadboard test circuit	• Amazingly compact • Cheap • Runs on 5-16V with on-board power regulation	• 6x the speed, 8x the memory. • 12-bit analog output (not just PWM!) • dual 12-bit A/D converters • 32-bit Cortex M4 processor • CAN bus, Capsense, multiple serial/SPI/I2C busses, etc. • Real-time clock (just add 32.768 kHz crystal)	• Pick the size and features you need for your project • Options range from the 8-pin Attiny45 to the Atmega328 used by the Uno. • Can be programmed using an Arduino • Can work with zero external components, if you're willing to run at its internal clock speed	This is a completely different paradigm. Define which (of many) possible features you want to use, define how they're connected internally, write a C program to tie it all together, upload the hardware description, and the chip becomes the hardware you describe.
Why Not	• A few of the Uno pins aren't immediately available • Only 4 analog inputs, 5 PWM outputs • For the same price (and size) you can have a Teensy 3.2	• 5 GPIO pins, only 3 are independent of the USB port • No serial communications: USB port can only be used for programming/power, *not* communications. • Roughly 5kB of program space • Many Trinket-level projects could probably be done with a 555 timer.	• Most Arduino programs run fine, but sometimes you need to do just a bit of tweaking. • Some pins are not breadboard-friendly. • 3.3V device. The Teensy 3.2 is "5V tolerant" on inputs, but will not supply 5V on outputs.	• You'll need to get an AVR programmer, or program an Arduino to be an AVR programmer. (not too hard...) • Not as user-friendly as most other options.	• The learning curve is a step function. • Not open-source (but free software) • Windows-only • Small user base (compared to Arduino) so it's harder to get help.
When/ Where	Any application for which you'd use an Arduino Uno.	• Simple minimalist projects. • Blinky LEDs, simple sensor/warning/interlock systems, and other situations where an Arduino Pro Micro is overkill • 555-type circuits with long time constants	Anywhere you want! Unless you're straight out of grad school, this may be more computational power than was in your first computer, in a device the size of a postage stamp.	• Custom embedded devices • Replacements for logic circuits when speed is not critical • 555-type circuits with more complex behavior or long time constants	Instead of having a microcontroller control external analog and digital circuitry, everything just runs on the PSoC. And if you need to change the hardware you're using, you can just upload a new hardware description. (Up to a point!)
How	https://www.sparkfun.com/products/12640	https://www.adafruit.com/product/1501	https://www.pjrc.com/store/teensy32.html	https://create.arduino.cc/projecthub/arjun/programming-attiny85-with-arduino-uno-afb829	Mark Masters runs an ALPhA Immersion for this one, http://www.advlab.org/imm2017_ipfw_PSoC.html or http://www.cypress.com and search for CY8CKIT.
Similar Options	• Arduino Micro • Arduino Nano	Adafruit.com sells several varieties of these, with alternate chips, better capabilities, and slightly higher cost.	PJRC sells several other varieties; if you need lots of GPIO pins, or a built-in SD card slot, check out the Teensy 3.5 and/or 3.6.	Lots of options: the ones I find most useful are: • Attiny85: 8-pin DIP, 8kB memory • Attiny84: 14-pin DIP, 8kB memory • Atmega328: 28-pin DIP, 32kB memory (Uno's MCU)	The Cypress PSoC 5LE dev board ($10) is even more powerful and includes USB also. This is very useful for instrument development!
Best Practice	Ideal for an electronics class, or for student projects. The small size and low mass make it a great choice for, say, running experiments mounted on drones.	A bare Attiny85 is often a better choice, unless you don't have 5V available in your project in which case the voltage regulator on this board is very handy.	Complex projects, instrumentation. If you're making a lab instrument, this device is more than sufficient to implement an entire GPIB/SCPI interface.	If you're building more than one, you should consider this option just on economic grounds.	This is by far the most powerful device here. The processor is very capable and the ability to implement all the external analog and/or logic hardware on one chip is a game-changer. For example, instead of writing a program to carefully track the position of a quadrature encoder, just define part of the hardware to be a counter that tracks posititon automatically and have the program just ask the counter for the current position when necessary.

No mastodons were harmed in the making of this poster.