Arduino Comparison Guide a [learn.sparkfun.com tutorial](#)

Available online at: http://sfe.io/t122

Contents

Introduction

Let's face it, there are a a lot of different Arduino boards out there. How do you decide which one you need for your project? In this tutorial, we'll take a look at the diverse world of Arduino boards. We'll begin with a tabular overview of the features each board has. Then we'll delve deeper into each board, examining the pros, cons, and example use-cases.
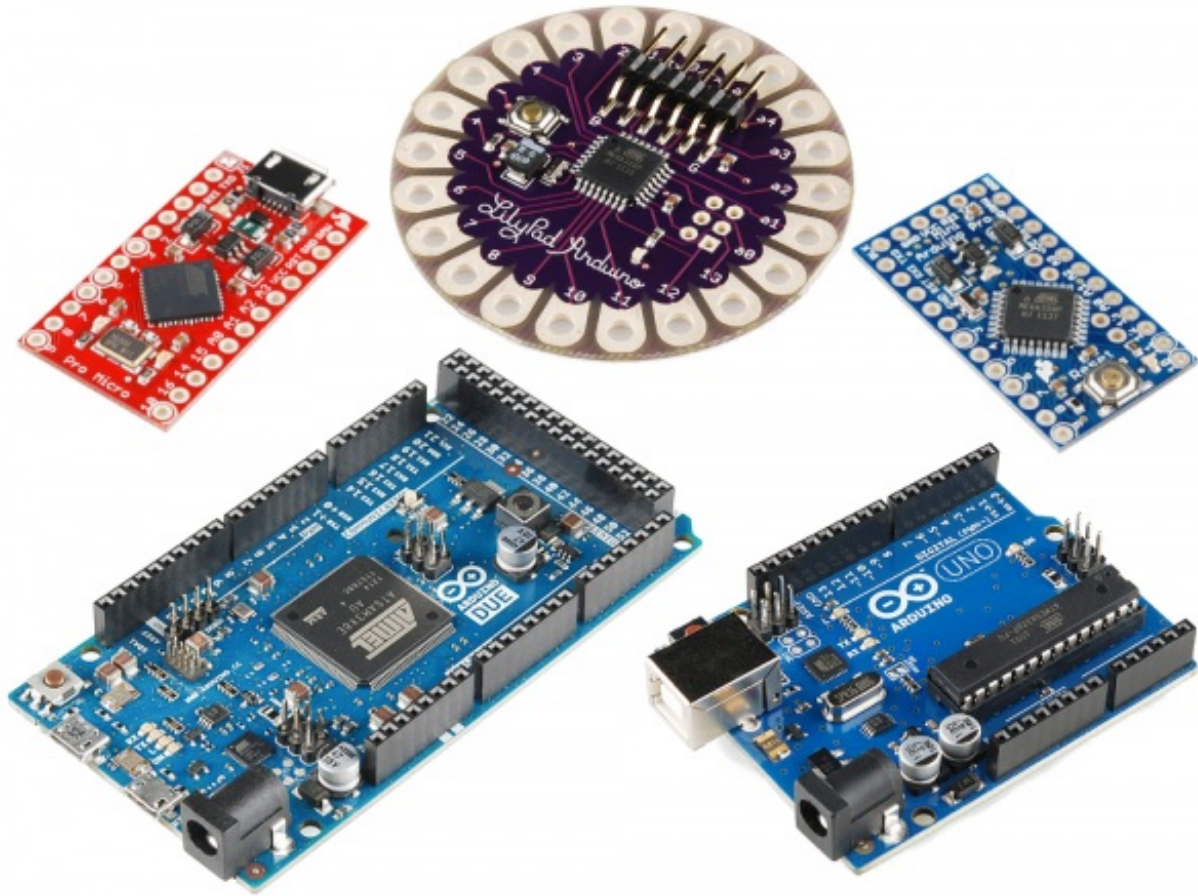
[Arduino](#) is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments. Or more simply, you load on some code and it can read sensors, perform actions based on inputs from buttons, control motors, and accept [shields](#) to further expand it's capabilities. Really, you can do almost anything.

All Arduino boards have one thing in common: they are programmed through the [Arduino IDE](#). This is the software that allows you to write and upload code. Beyond that, there can be a lot of differences. The number of inputs and outputs (how many sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, and form factor are just a few of the variables. Some boards are designed to be embedded and have no programming interface (hardware) which you would need to buy separately. Some can run directly from a 3.7V battery, others need at least 5V. Check the chart on the next page to find the right Arduino for your project.

Suggested Reading

- If you don't know what Arduino is but found yourself here, you may want to start with our [What is an Arduino tutorial](#).
- You should also have a good understanding of the Arduino IDE. If you need help installing it, visit [this tutorial](#).

Totally Tabular

For a quick cross-comparison of the available Arduinos, here's a (totally tubular) tabular look at the variety of boards. The boards are sorted by their main microcontroller, which is what defines most of the characteristics for each of them.

Item	System Voltage	Clock Speed	Digital I/O	Analog Inputs	PWM	UART	Programming Interface	Cost
ATmega328 Boards — 32kB Program Space // 1 UART // 6 PWM // 4-8 Analog Inputs // 9-14 Digital I/O								
Arduino Uno - R3	5V	16MHz	14	6	6	1	USB via ATMega16U2	$29.95
Arduino Uno R3 SMD	5V	16MHz	14	6	6	1	USB via ATMega16U2	$29.95
RedBoard	5V	16MHz	14	6	6	1	USB via FTDI	$24.95
Arduino Pro 3.3V/8MHz	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header	$14.95
Arduino Pro 5V/16MHz	5V	16MHz	14	6	6	1	FTDI-Compatible Header	$14.95

Arduino Mini 05	5V	16MHz	14	8	6	1	FTDI-Compatible Header	$33.95
Arduino Pro Mini 3.3V/8MHz	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header	$9.95
Arduino Pro Mini 5V/16MHz	5V	16MHz	14	8	6	1	FTDI-Compatible Header	$9.95
Arduino Ethernet	5V	16MHz	14	6	6	1	FTDI-Compatible Header	$59.95
Arduino Fio	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header or Wirelessly via XBee[†]	$24.95
LilyPad Arduino 328 Main Board	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header	$21.95

	Voltage	Clock	Digital I/O	Analog Inputs	PWM	UART	Connection	Price
LilyPad Arduino Simple Board	3.3V	8MHz	9	4	5	0[±]	FTDI-Compatible Header	$19.95
ATmega32U4 Boards — 32kB Program Space // 1 UART // 5-7 PWM // 12 Analog Inputs // 9-20 Digital I/O								
Arduino Leonardo	5V	16MHz	20[*]	12	7	1	Native USB	$24.95
Pro Micro 5V/16MHz	5V	16MHz	12	12	5	1	Native USB	$19.95
Pro Micro 3.3V/8MHz	3.3V	8MHz	12	12	5	1	Native USB	$19.95
LilyPad Arduino USB	5V	16MHz	9	12	5	0	Native USB	$24.95
ATmega2560 Arduino Mega's — 256kB Program Space // 4 UARTs // 14 PWM // 16 Analog Inputs // 54 Digital I/O								
Arduino Mega 2560 R3	5V	16MHz	54	16	14	4	USB via ATMega16U2	$58.95

Mega Pro 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header	$44.95
Mega Pro 5V	5V	16MHz	54	16	14	4	FTDI-Compatible Header	$44.95
Mega Pro Mini 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header	$49.95

AT91SAM3X8E Arduino Due — 512kB Program Space // 4 UARTs // 12 PWM (2 DAC) // 12 Analog Input // 54 Digital I/O

Arduino Due	3.3V	84MHz	54	12	12	4	USB native	$49.95

---

*The Arduino Leonardo has the same GPIO pin-count as the other "Uno" style boards, but more of the pins play "double duty" as both analog and digital pins, hence the higher numbers.

†The miniUSB connector on the Arduino Fio is used for battery charging only. An Xbee module is not included with this board.

‡The LilyPad Simple Board does have one UART but the pins aren't broken out to pads. Serial communication can be achieved through the FTDI header.

Glossary of Terms:

Microcontroller (MCU): The microcontroller is the heart (or, more appropriately, the brain) of the Arduino board. The Arduino development board is based on AVR microcontrollers of different types,

each of which have different functions and features.

Input Voltage: This is the suggested input voltage range for the board. The board may be rated for a slightly higher maximum voltage, but this is the safe operating range. A handy thing to keep in mind is that many of the [Li-Po batteries that we carry](#) are 3.7V, meaning that any board with an input voltage including 3.7V can be powered directly from one of our Li-Po battery packs.

System Voltage: This is the system voltage of the board, i.e. the voltage at which the microcontroller is actually running. This is an important factor for shield-compatibility since the logic level is now 3.3V instead of 5V. You always want to be sure that whatever outside system with which you're trying to communicate is able to match the [logic level](#) of your controller.

Clock Speed: This is the operating frequency of the microcontroller and is related to the speed at which it can execute commands. Although there are rare exceptions, most ATmega microcontrollers running at 3V will be clocked at 8MHz, whereas most running at 5V will be clocked at 16MHz. The clock speed of the Arduino can be divided down for power savings with a few tricks if you know what you're doing.

Digital I/O: This is the number of digital input/output (I/O) pins that are broken out on the Arduino board. Each of these can be configured as either an input or an output. Some are capable of PWM, and some double as serial communication pins.

Analog Inputs: This is the number of [analog input](#) pins that are available on the Arduino board. Analog pins are labeled "A" followed by their number, they allow you to read analog values using the analog-to-digital converter (ADC) in the ATMega chip. Analog inputs can also be configured as more digital I/O if you need it!

PWM: This is the number of digital I/O pins that are capable of producing a [Pulse-width modulation](#). (PWM) signal. A PWM signal is like an analog output; it allows your Arduino to "fake" an analog voltage between zero and the system voltage.

UART: This is the number of separate [serial communication](#) lines your Arduino board can support. On most Arduino boards, digital I/O pins 0&1 double as your serial send and receive pins and are shared with the serial programming port. Some Arduino boards have multiple UARTs and can support multiple serial ports at once. All Arduino boards have at least one UART for programming, but some aren't broken out to pins that are accessible.

Flash Space: This is the amount of program memory that the chip has available for your to store your sketch. Not all of this memory is available as a very small portion is taken up by the bootloader (usually between 0.5 and 2KB).

Programming Interface: This is how you hook up the Arduino board to your computer for programming. Some boards have a USB jack on-board so that all you need to do is plug them into a USB cable. Others have a header available so that you can plug in an [FTDI Basic breakout](#) or [FTDI Cable](#). Other boards, like the Mini, break out the serial pins for programming but aren't pin-compatible with the FTDI header. Any Arduino board that has a USB jack on-board also has some other hardware that enables the serial to USB conversion. Some boards, however, don't need

additional hardware because their microcontrollers have built-in support for USB.

This table serves to overview all of the Arduino boards. You may have an idea of which Arduino board is right for you now. On the next few pages we'll split into different categories of Arduinos and explore their differences more closely.

ATmega328 Boards

The ATmega328 (and the ATmega168 before that, and ATmega8 before that,…) is a staple of the Arduino platform. 32kB of flash (program space), up to 23 I/Os – eight of which can be analog inputs – operating frequencies of up to 20 MHz. None of it's specifications are flashy, but this is still a *solid* 8-bit microcontroller. For many electronics projects, what the 328 provides is still more than enough.

The Arduino boards on this page all feature the ATmega328 as their main MCU brain. The microcontroller alone makes every board on this page nearly identical in terms of I/O count and memory. Their differences stem from things like programming interfaces, form factors, and operating voltages.

The Main Event: Arduino Uno

The Arduino Uno is the "stock" Arduino. It's what we compare every, other, Arduino-compatible board to. If you're just getting into Arduino, **this is the board to start with**
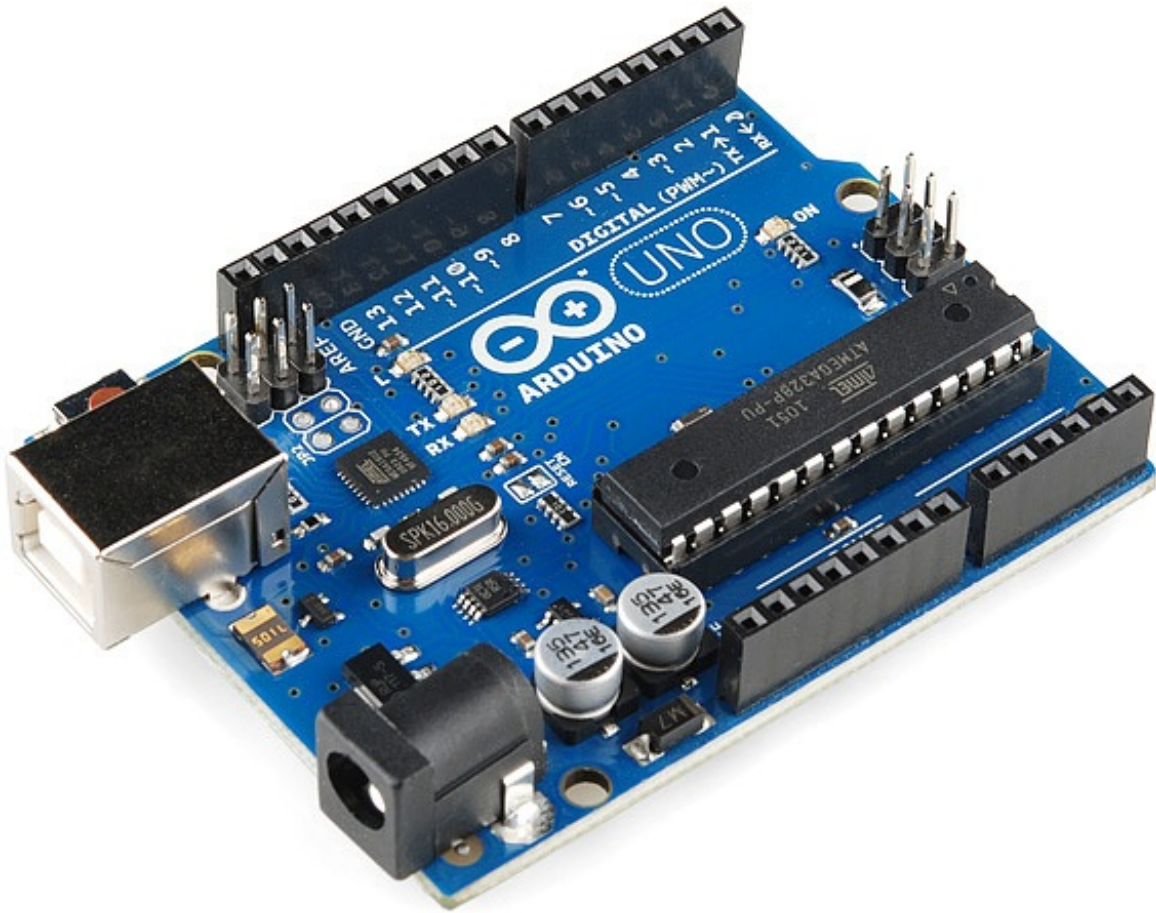
The Uno comes in two flavors, through-hole and SMD, which use either a through-hole or surface-mount ATmega328. The through-hole version (pictured above) is nice because you can take the chip out and swap in a new one (in case the magic, blue smoke is released), but the SMD version has the potential to be more readily available (PTH chips are increasingly being phased out of existence).

The Arduino Uno can be powered through either the USB interface, or an external barrel jack. To connect it to a computer you'll need a type-B-to-A USB cable (like the USB connector on most printers).

A Modification: RedBoard

One of the greatest things about Arduino is the fact that the entire project is open-source. The schematics, hardware design files, and source code are all freely available for viewing and modification. Released under a Creative Commons Share Alike license, anyone is free to riff on the hardware design and produce their own version. That's how a product like the RedBoard comes to be. It still looks and acts just like an Arduino Uno, but is slightly modified to make the board better-suited to certain purposes.
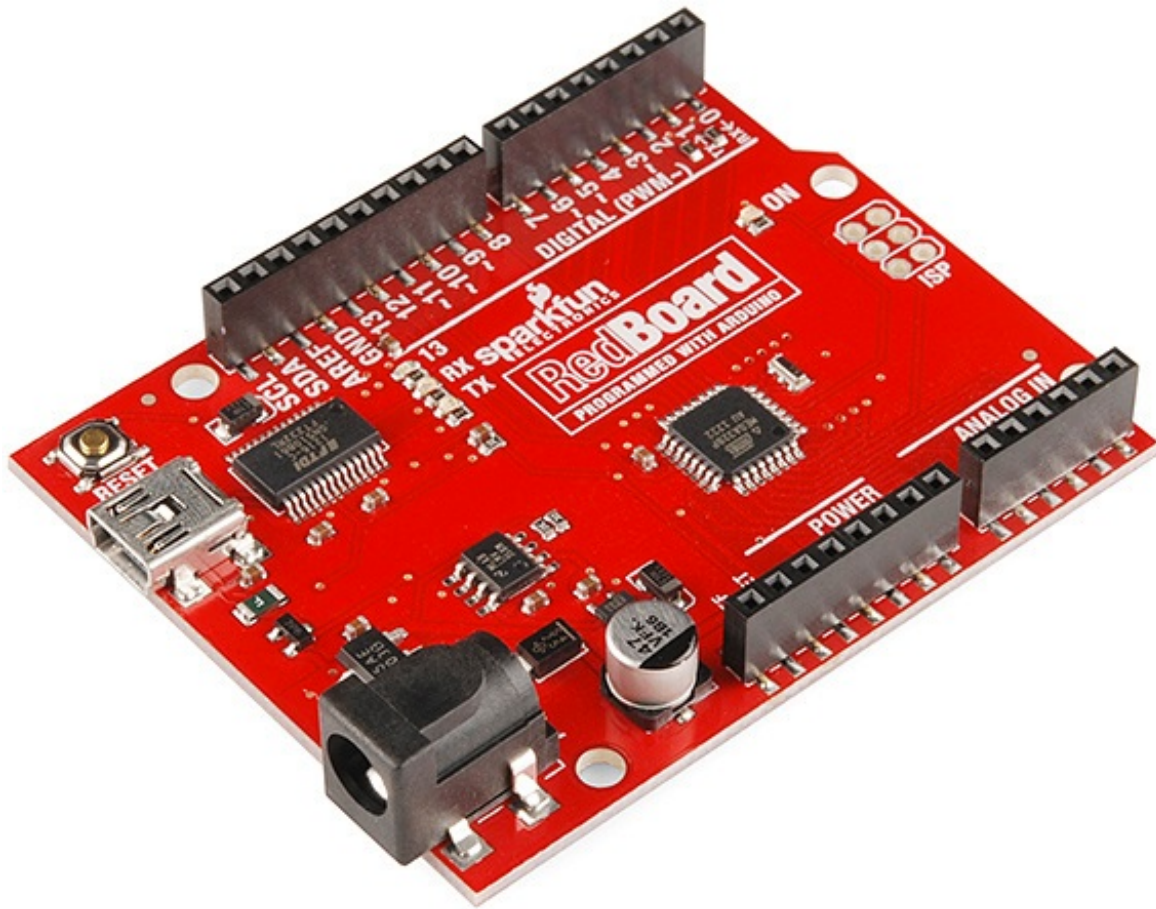
The RedBoard is nearly identical to the Uno, but there are a few key differences:

- **USB connector**: The Redboard uses the smaller mini-B connector, so you'll need a [mini-B-to-A USB cable](#) to connect it to your computer.
- **USB-to-Serial Transciever**: The Arduino Uno uses an ATmega16U4 loaded with custom firmware to convert between USB and serial. The RedBoard uses the FTDI FT232RL. This difference is only really prevalent when **installing drivers** because each requires a different driver file.
- **SMD vs PTH**: The RedBoard is only offered in a SMD version, and it takes SMD a step further by making *every* component surface-mount. No sharp edges on the bottom of the board!
- **Color**: True to it's name, the RedBoard comes in ~~Ferarri~~ SparkFun red. It won't have any real influence on the operation of the Arduino, but it certainly affects the board's swag-factor.
- **Price**: Because we manufacture the board in-house, here in Boulder, CO, we can afford to keep the price-tag a tad lower.

Like the Uno, the RedBoard is great for beginners. On the whole, it should offer the same Arduino experience as an Uno might. For a deeper comparison between the RedBoard and Uno, check out our [RedBoard vs. Uno tutorial](#).

For the Pros

Arduino Pros are a scaled-down version of the Uno. There's still an ATmega328 on there, but removed are the connectors and USB-to-serial-converting circuitry. Basically, this is the bare-

minimum an Arduino needs to still be an Arduino. As the name would imply, these boards are intended for use by more experienced Arduino-ers.
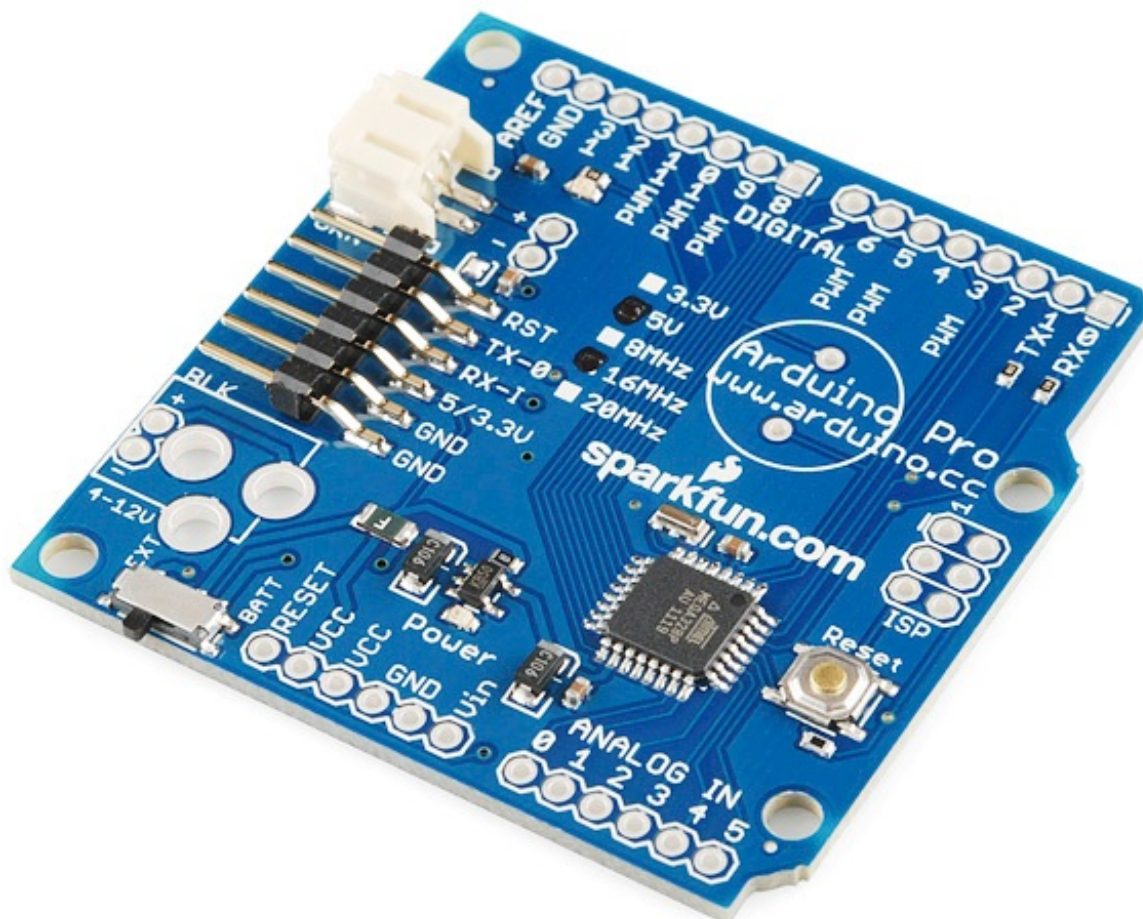
You'll need more than just a USB cable to program an Arduino Pro; an external board is required to convert USB from your computer to serial that the Arduino understands. There are various boards and cables that can accomplish this task, we recommend the FTDI Basic Breakout.

This board mates up to the 6-pin, right-angle connector on the edge of the board. When you're done programming and ready to stick the board into a project, just unplug the FTDI Basic.

The smaller form factor and absence of connectors means this board can be more**custom-tailored** to fit into a project. You can solder wires or connectors directly onto the pins you need. Then again, it has the same pin footprint as the Uno, so it's still shield compatible.

The Pros come in two varieties: 5V/16MHz and 3.3V/8MHz. The 5V/16MHz board runs at the same voltage and speed as the Arduino Uno. The 3.3V/8MHz board is unique, though, because it can operate at a lower voltage. A lower operating voltage makes the board easier to power with batteries (LiPos specifically), but it also means the clock speed has to be turned down. The 3.3V/8MHz board runs at half the speed a regular Arduino Uno…but 8MHz is still pretty darn fast for many applications. You can still turn an LED on and off more than a million times per second!

Of course, if this board is still to big, you can shrink it down even further…

Pro Mini's

The Mini boards pack all of the remaining punch of the Arduino Pro into a much smaller footprint. Every pin is still broken out (actually, *more* pins are broken out), they're just in a very different footprint.
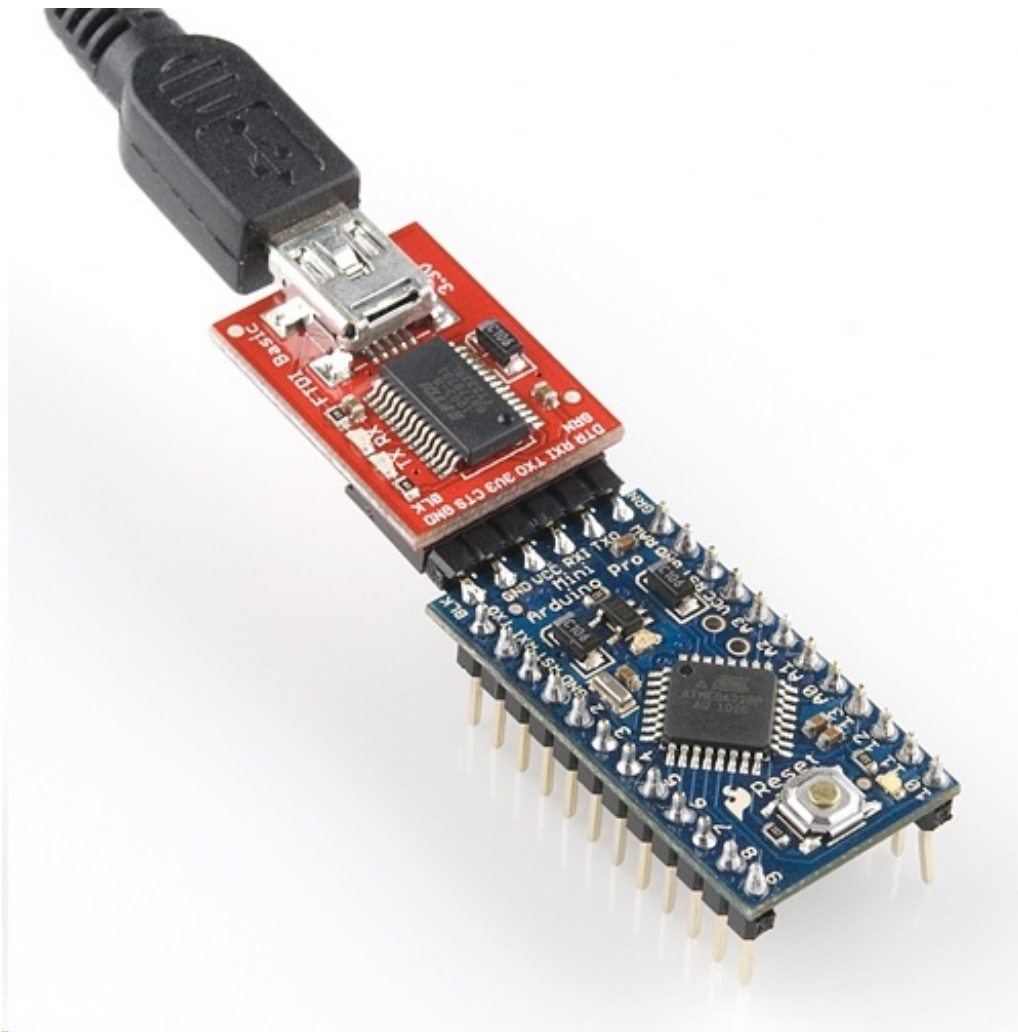

An Arduino Pro Mini attached to an FTDI Basic, which provides power and uploads code.

Obviously, these boards aren't shield-compatible, but they are breadboard-compatible. You can solder male headers into the Pros, and straddle it across the breadboard's middle strip. The small form-factor also makes them very conducive to embedding into projects (like in the H2O pH Probe).

Like the regular Pro boards, these are offered in 5V/16MHz and 3.3V/8MHz varieties. And you still have to program them with an FTDI Basic.

The Arduino Mini 05 could also be lumped into this category. It shares a lot in common with the *Pro Minis*, except it comes with male headers pre-soldered in.
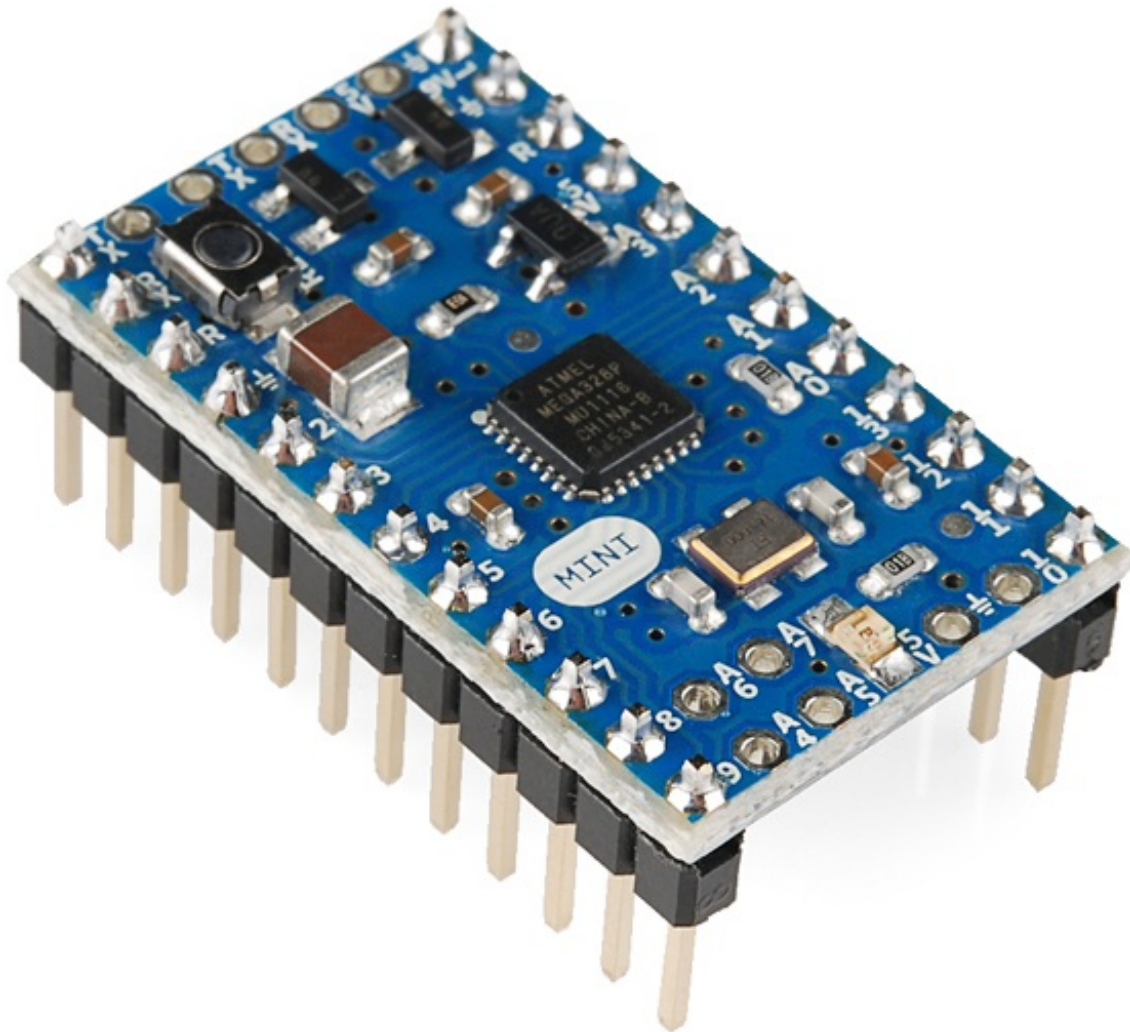
Etcetera

There are innumerable Arduino-compatible boards out there which make use of the ATmega328. Many, like the Arduino Pros, require an FTDI Basic to receive code, but they add on extra hardware to make them unique. The Arduino Ethernet, where the Arduino Uno and an Ethernet Shield are smashed onto a single board, is a good example of this.

And, the Arduino Fio too. This board wires up the ATmega328 to an XBee (or XBee-compatible) wireless transceiver, so your Arduino can communicate wirelessly with other devices.
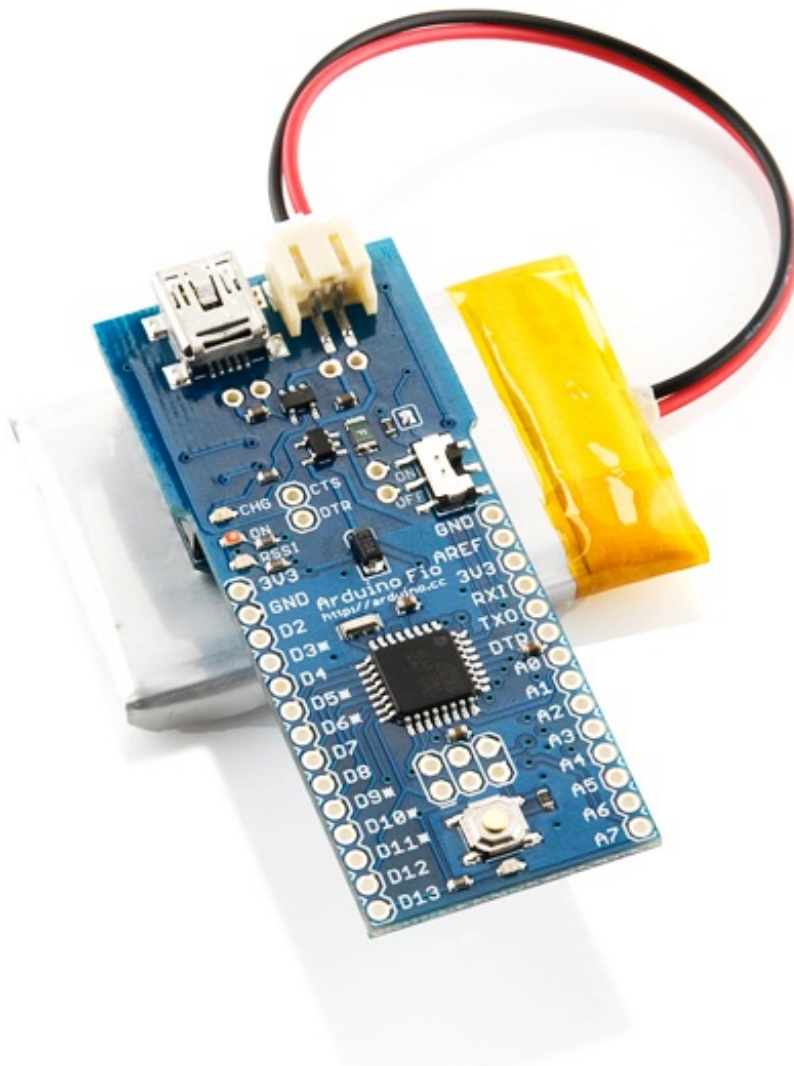
The list could go on and on. If you see a board with that recurring, six-pin, serial header, and an ATmega328 doing all of the processing, its specifications probably aren't all that different from an Arduino Pro.

ATmega32U4 Boards

The next step in the Arduino evolutionary chain was merging the USB-to-Serial programming part of the board onto the main MCU. That meant we had to leave the ATmega328 behind – becuase it doesn't natively support USB – in favor of the ATmega32U4. Aside from the additional USB support, the 32U4 is largely similar to the 328. Both are 8-bit AVRs with 32kB of flash memory, 22-ish I/O lines, ADCs, UARTs, timers, etc.

These ATmega32U4 boards often have the benefit of being**cheaper** than the ATmega328-based boards – there's one less costly IC to put on there. They can also do things regular Arduino boards can't, like emulate a USB keyboard/mouse. On the downside, they can be less reliable, and more difficult to use.

Arduino Leonardo

The Leonardo is the patriarch of all ATmega32U4 Arduino boards. It shares the same form factor and I/O placement (analog, PWM, I^{2}C pins in the same place) as the Arduino Uno, so it remains shield compatible.

Differences between the Leonardo and the Uno? Aside from the new microcontroller, and lack of a second USB-to-Serial-converting IC, there's not many. The USB connector is different, the Leonardo connects to a computer via a micro-B USB cable. The driver installation process is also a bit more involved – sometimes it can take a little extra fidgeting to get the board installed on your computer.

Pro Micro

Just as the Pro Mini took the guts of the Arduino Uno and shrunk them down, the Pro Micro works as a miniature version of the Leonardo. Unlike the Pro Mini, the Pro Micro doesn't require an external board to upload a sketch – the 32U4 takes care of everything!
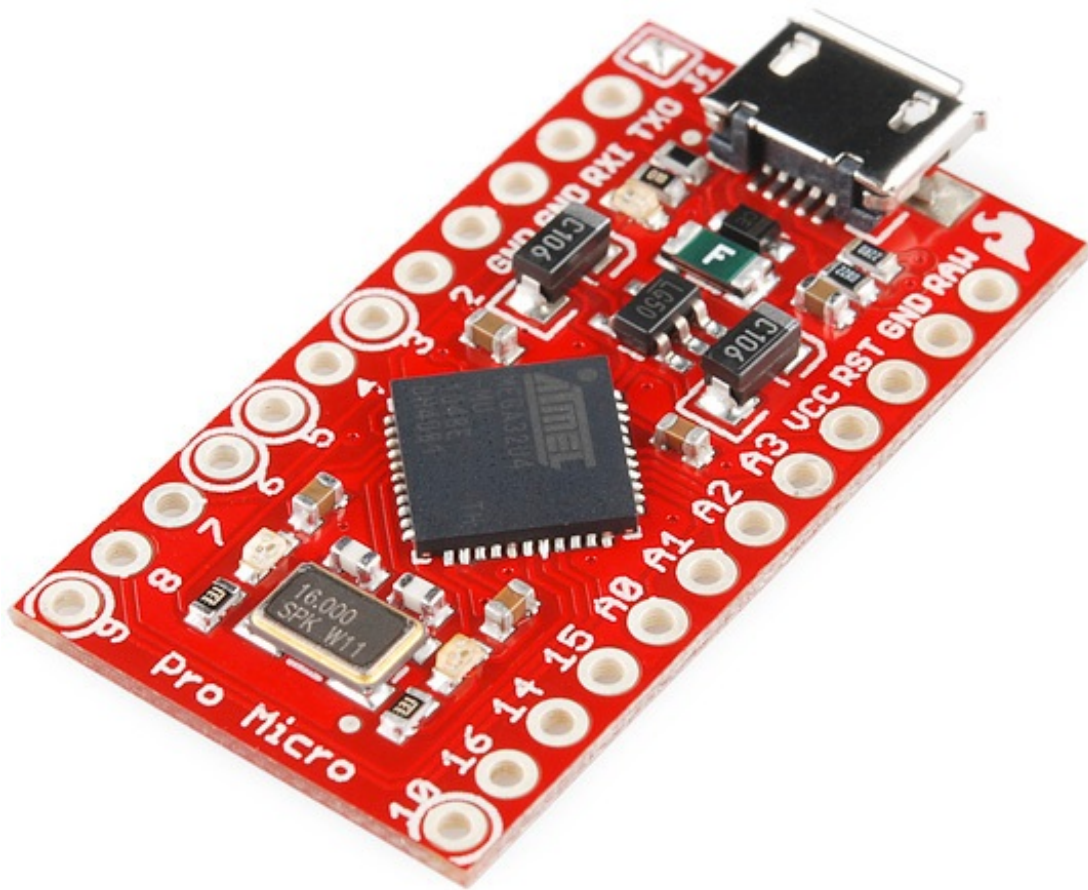
The Pro Micro comes in the standard 5V/16MHz operating range or a more unique 3.3V/8MHz variant.

Pro Micros are among the more complicated Arduino boards to get up and running. There are extra steps required to enable them in your Arduino environment, and a misstep can (at least temporarily) "brick" the Pro Micro.

These boards are a good choice if you're an advanced Arduino-er and have a small USB-oriented project in mind (a mini USB keyboard/mouse?).

More Variants!

There are plenty of other riffs on the Leonardo design as well. There's the Fio v3, for any Arduino Leonardo project you might want to add an XBee to.
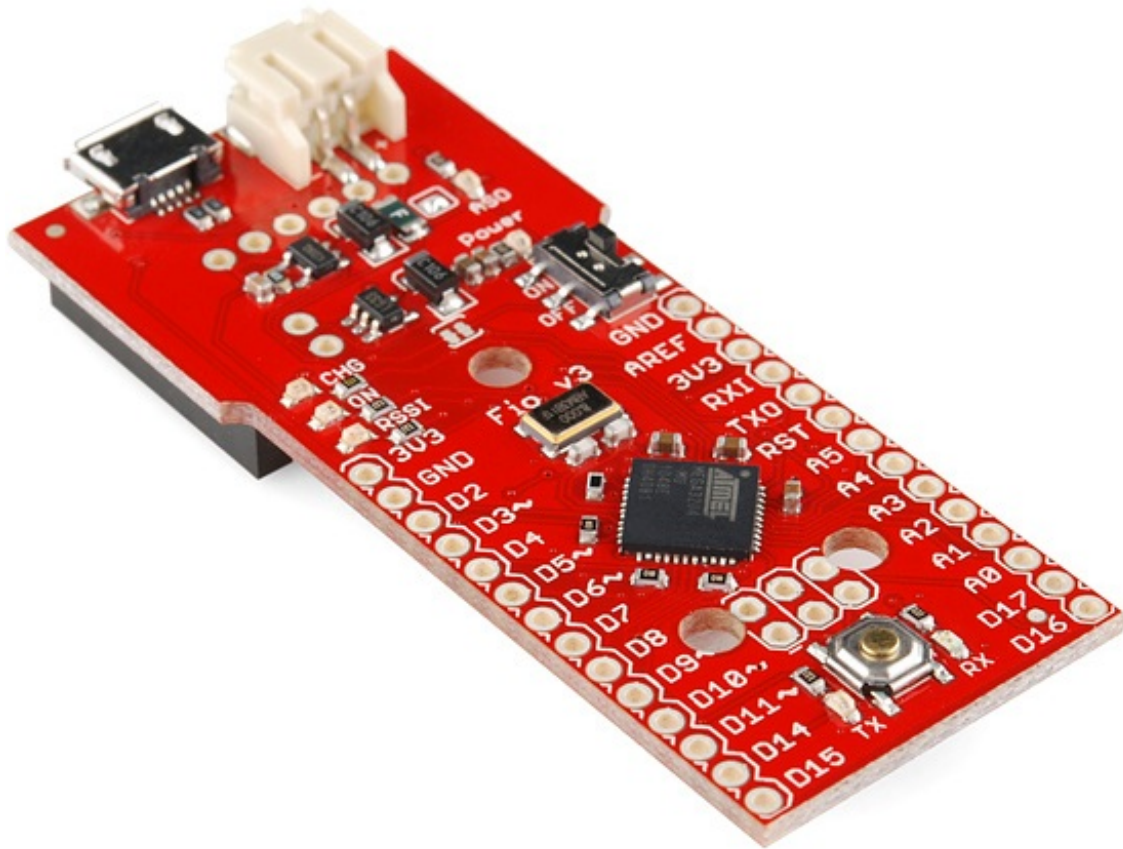
And there are more unique Leonardo-compatible boards, like the MaKey MaKey, which can turn almost anything (from tin foil to Play-Doh to bananas) into a keyboard key.
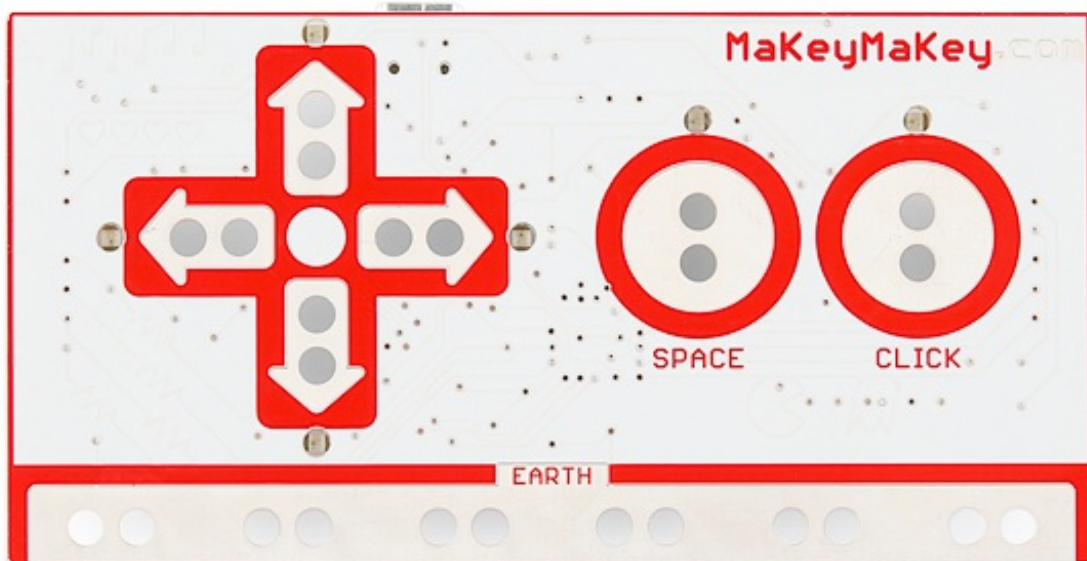
The ATmega32U4 can make some very cool **USB-integrated** Arduino projects.

Wearable Arduinos

The e-textiles segment of the Arduino market is ruled by LilyPads. These are identifiable as unique purple, flowery-looking, circular boards. The pins on LilyPads are called "petals", they have bigger

holes and copper filled to the edge of the board. These are designed so [conductive thread](#) can be sewn through the holes, and make electrical contact with the exposed copper on the petal.
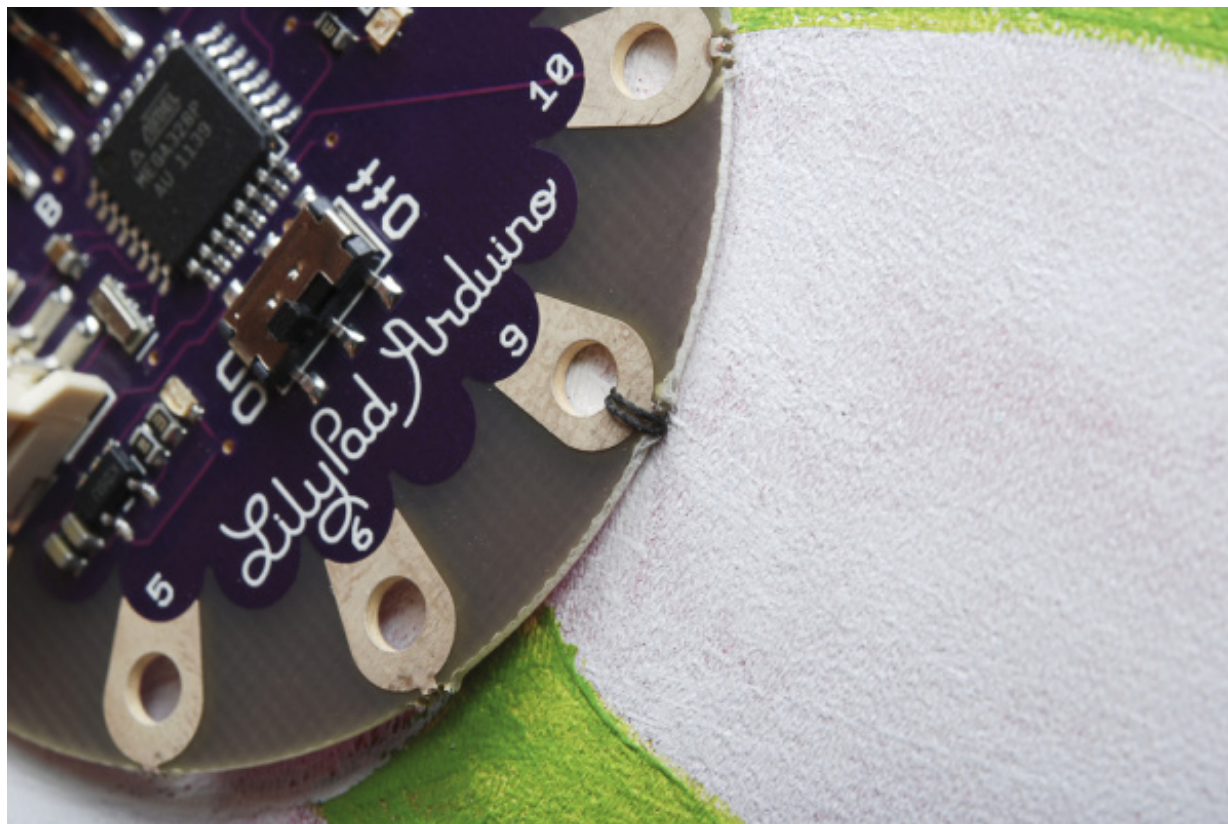
LilyPads are great for e-texitles – projects which combine electronics and fabric wizardy. They can be used to make nifty [talking aprons](#), [dice gauntlets](#), or more simple trinkets like [light-up firefly jars](#).

ATmega328 LilyPads

The O.G. LilyPad is the [LilyPad Arduino](#). This is basically a purple Arduino Pro 3.3V/8MHz. *All* of the pins – 14 digital and six analog – are arranged around the outer edge of the board. There are also inputs for power at the '+' and '-' pins.
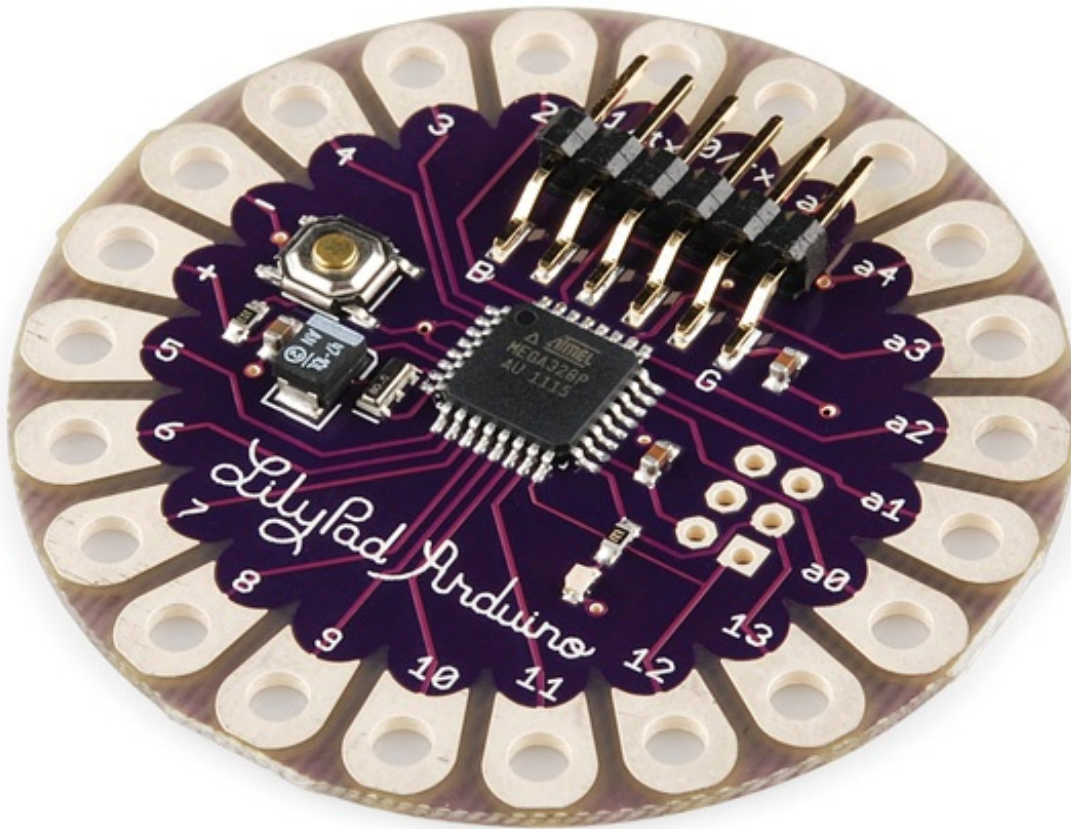
Like the Arduino Pro, the ATmega328 LilyPads are all programmed using an external FTDI Basic. There's even a [purple one](#) especially for LilyPads!

Keep it Simple

True to its name, the [LilyPad Arduino Simple](#) is an easier-to-use version of the standard LilyPad Arduino. Fewer pins are broken out – nine digital and four analog – but the LilyPad Simple has some added functionality too. This board is designed specifically to be powered by a [Lithium Polymer battery](#). There's an **on-board JST connector** for the battery to plug into, and there's even a special circuit dedicated to **charging the battery**.
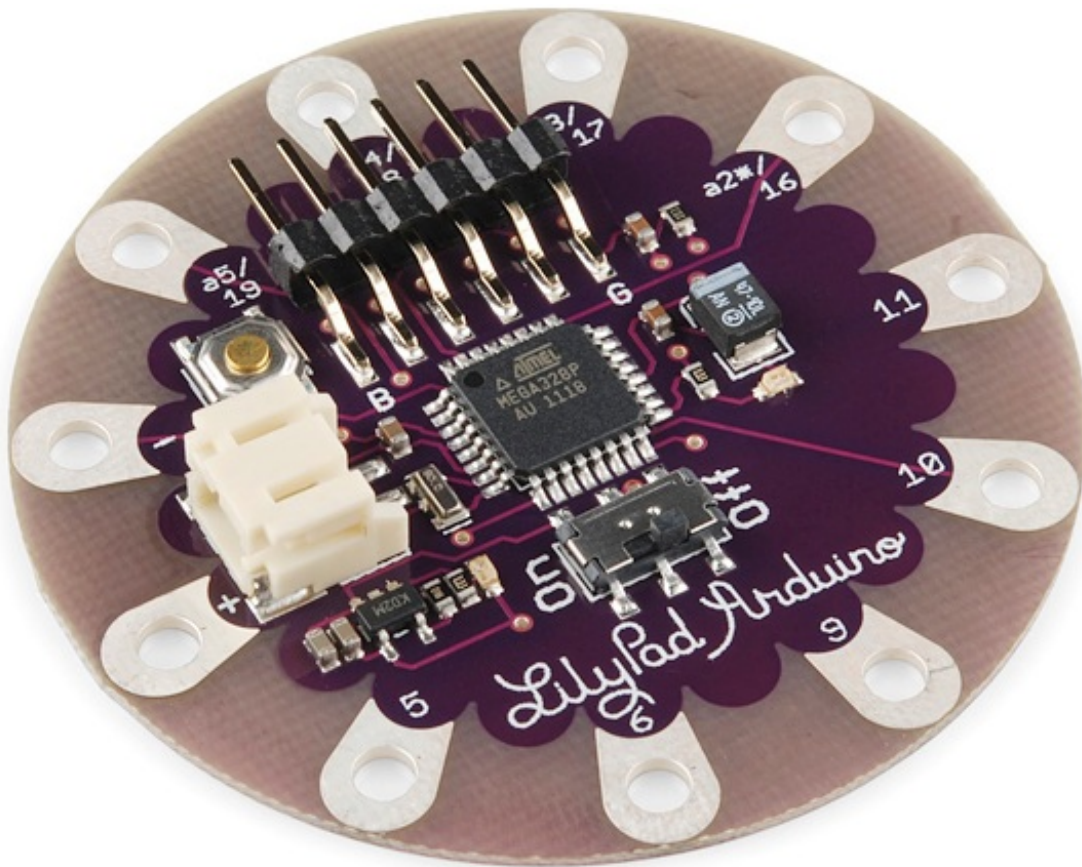
This Board is recommended for beginners and experts alike. As long as you're OK with having less I/O pins, the extra functionality of the Simple makes it well worth it.

Make it Snappy

The LilyPad Arduino SimpleSnap is an offshoot of the Simple. The layout is the same, the battery-charging functionality is still there, but instead of an array of petals around the board there are snaps.
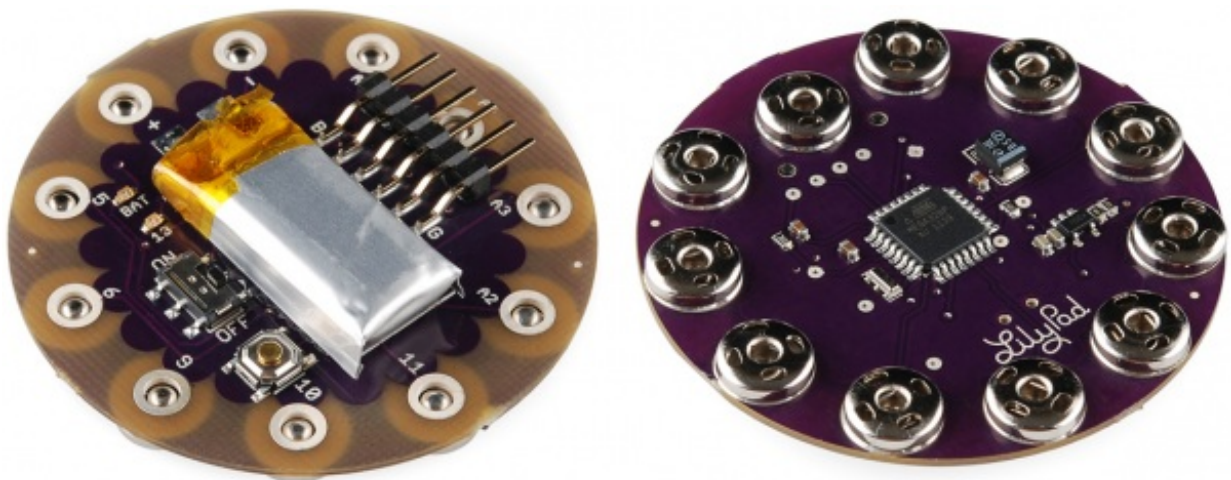

These snaps can be plugged into the mating snaps on a SimpleSnap ProtoBoard, or any size 1/0 snap already sewn into a fabric.

A **permanently attached battery** on the SimpleSnap also sets it far apart from other LilyPad boards. Just flick the switch to "ON" and the board should start running its sketch. To charge the battery, plug an FTDI Basic into the SimpleSnap.

This board is an awesome choice if you want the added ability of easily removing your LilyPad from an e-textiles project. You can unsnap the board and wash your textile, or even share it with another project.

LilyPads with USB Built-In

The boards above – all with an ATmega328 as the brain – all require an external FTDI Basic to have code uploaded to them. The LilyPad Arduino USB – which uses an ATmega32U4 – has all of the USB functionality on-board. It has a micro-B USB connector, so you just plug the cable in, attach it to your computer, and program away!
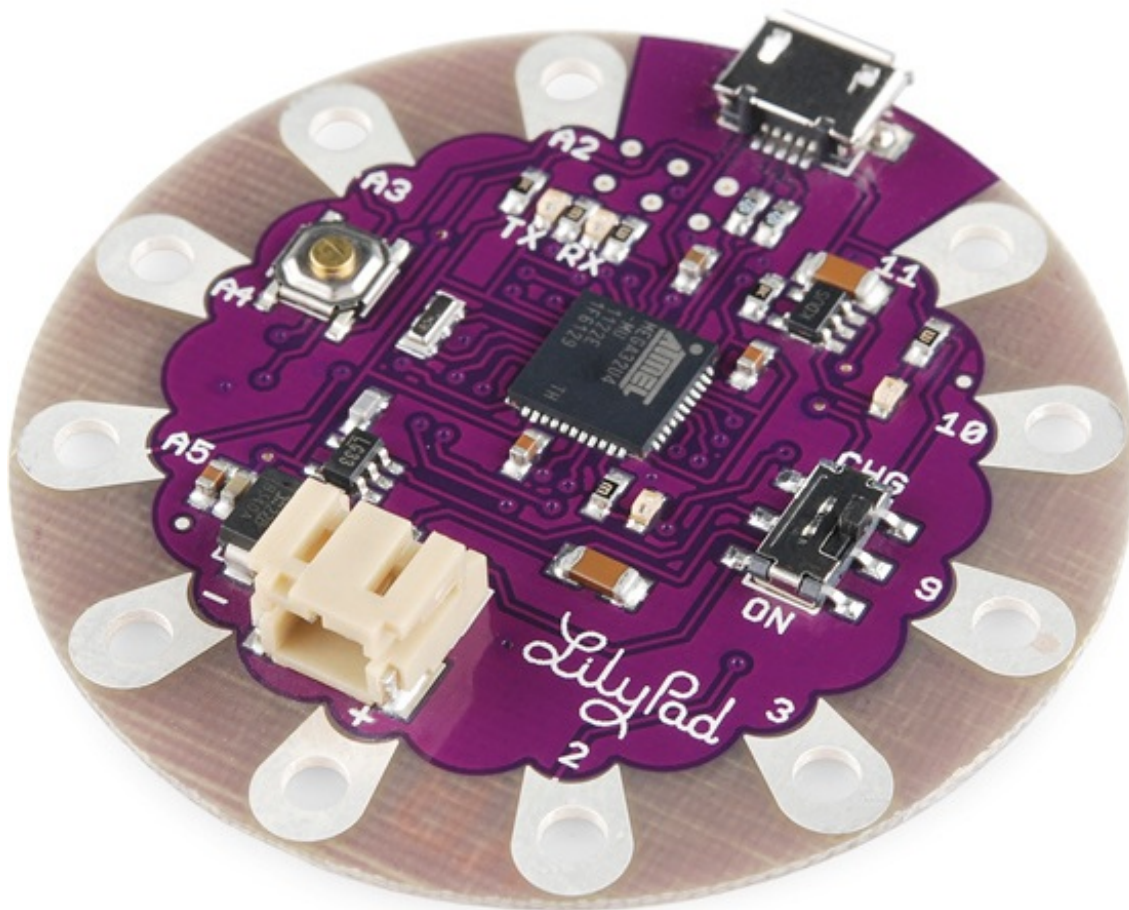
The LilyPad Arduino USB is comparable to other ATmega32U4 boards, like the Leonardo. It's got nine digital I/O's and 12 analogs. Like the LilyPad Simple, this board has an on-board JST connector and **LiPo battery charge circuit**. So it's very easy to pair with a LiPo battery and embed into your project.

Megas, ARMs, Yúns...Oh My!

Need some extra "beef" in your Arduino? Need more I/O pins, or a faster processor? That's where

Arduino's like the Mega or the Due come into the picture.

Arduino Mega: The Souped Up Uno

The [Arduino Mega](#) is what you might get if you packed four Arduino Uno's into one board. There are **54 I/O pins**, instead of the 14 an Uno gives you. That's a whole lot of extra LEDs! Instead of one hardware serial port, there are four. And the Mega sports a whopping **256 kB of flash** program space. Not to mention 16 analog inputs, and 14 PWM outputs. The Mega just has more of everything.
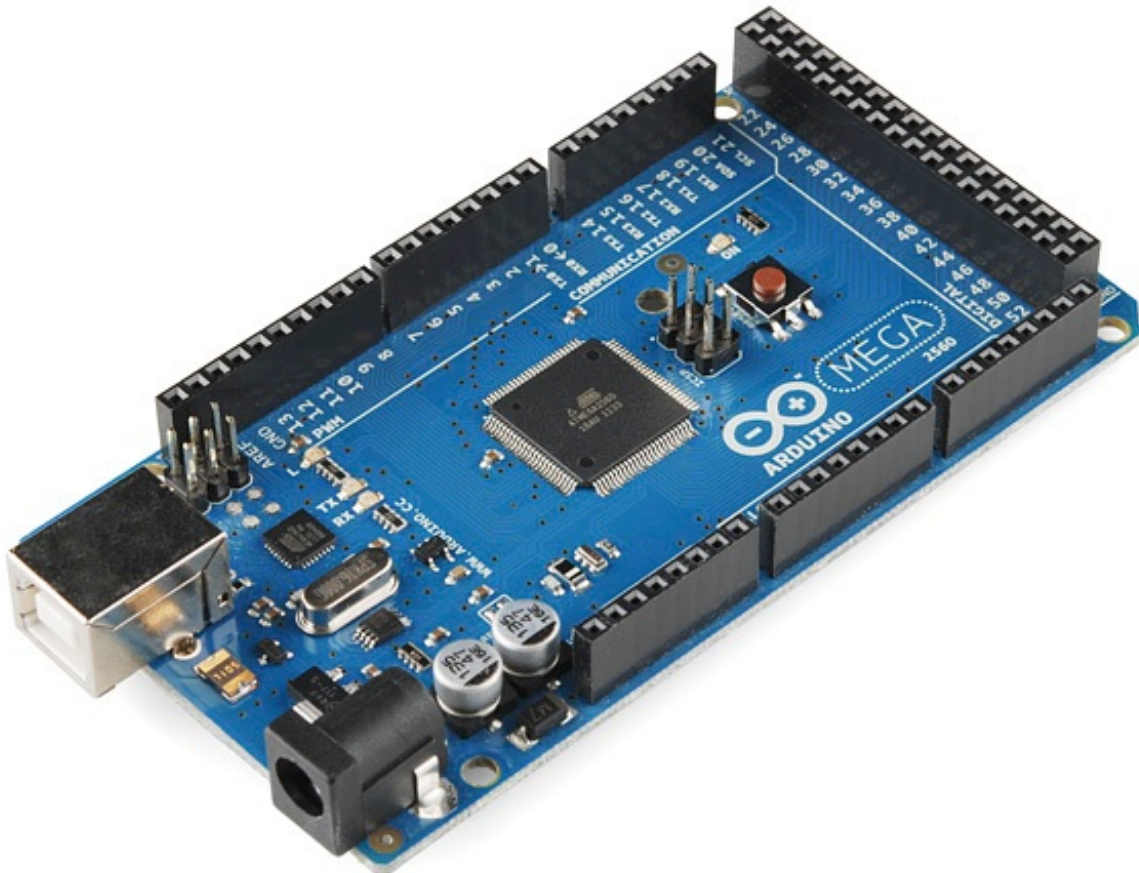

The brain of the Mega is an ATmega2560, a fully souped up ATmega328. Aside from the massive processor overhaul, the Mega still shares a lot in common with the Arduino Uno. There's a secondary IC on-board (an ATmega16U2) to convert USB-to-serial to allow USB programming. It runs at the same speed – 16 MHz. All of the pins are broken out in a way that keeps the board shield-compatible. Because of these similarities, the Mega is a good option for Arduino beginners and experts alike.

If your Arduino project is hitting a wall because you don't have enough I/O, or if you're running out of program space, consider stepping up to the Mega.

Mega Pros

This seems to be a trend, doesn't it? If you're looking for a stripped-down version of the Mega – no USB-to-serial converter, no connectors – there's the [Mega Pro 5V](#) and [3.3V/8MHz](#) variants.
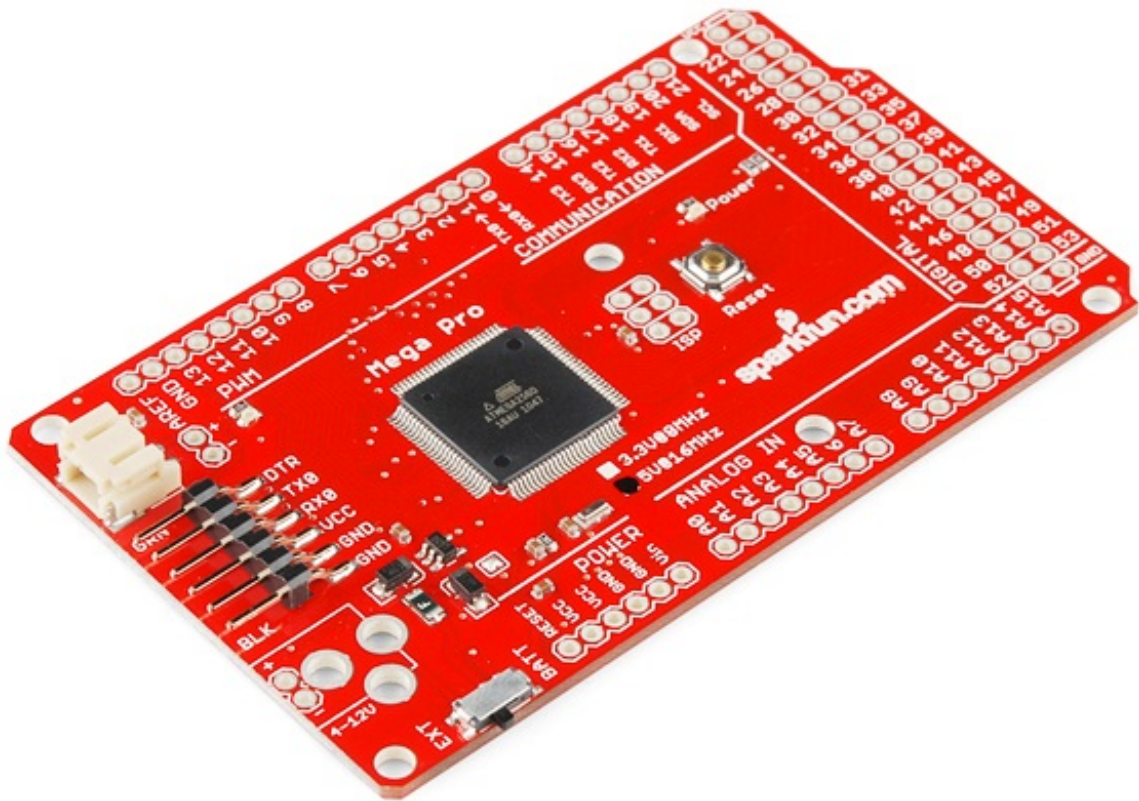
With all of those I/O's it's hard to make a "mini" version, but the Mega Pro Mini 3.3V gets close. This board breaks out each of the pins to sets of small 8-pin connectors. A special cable is required to interface with them.
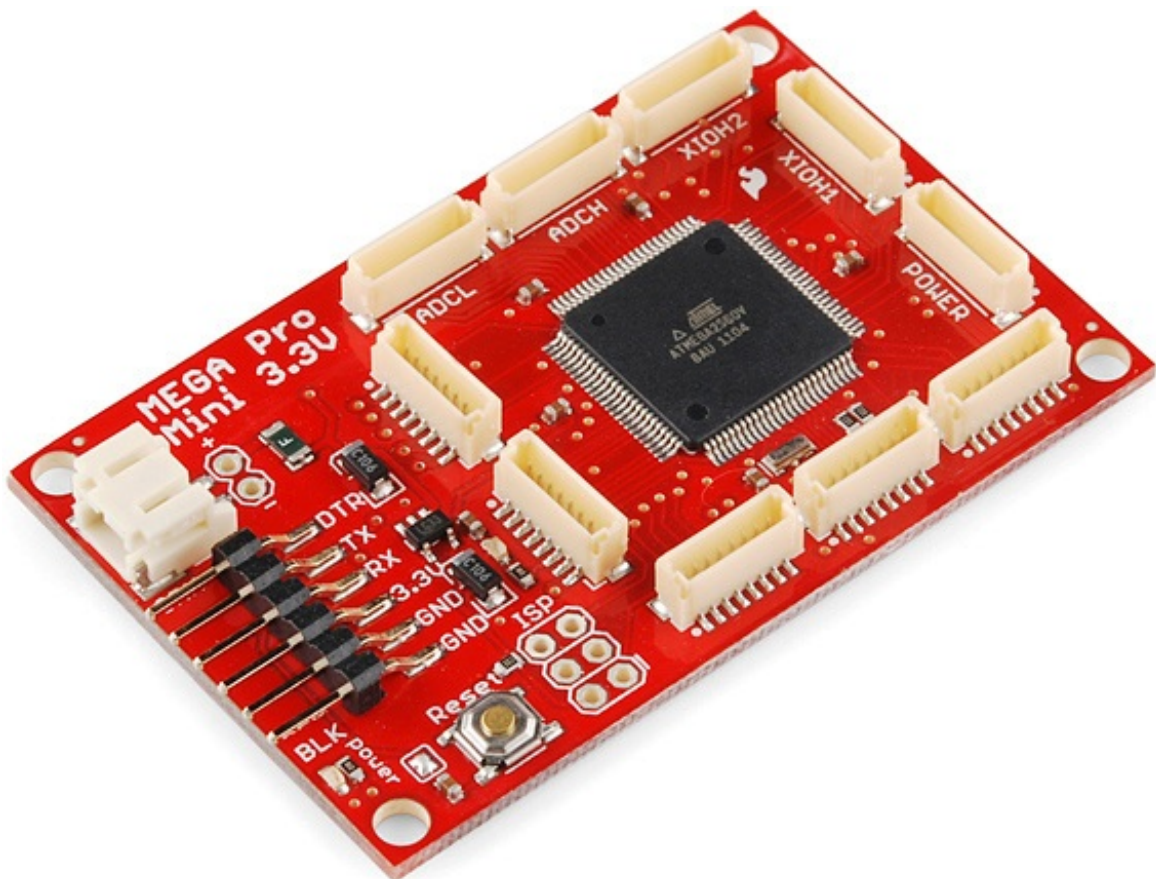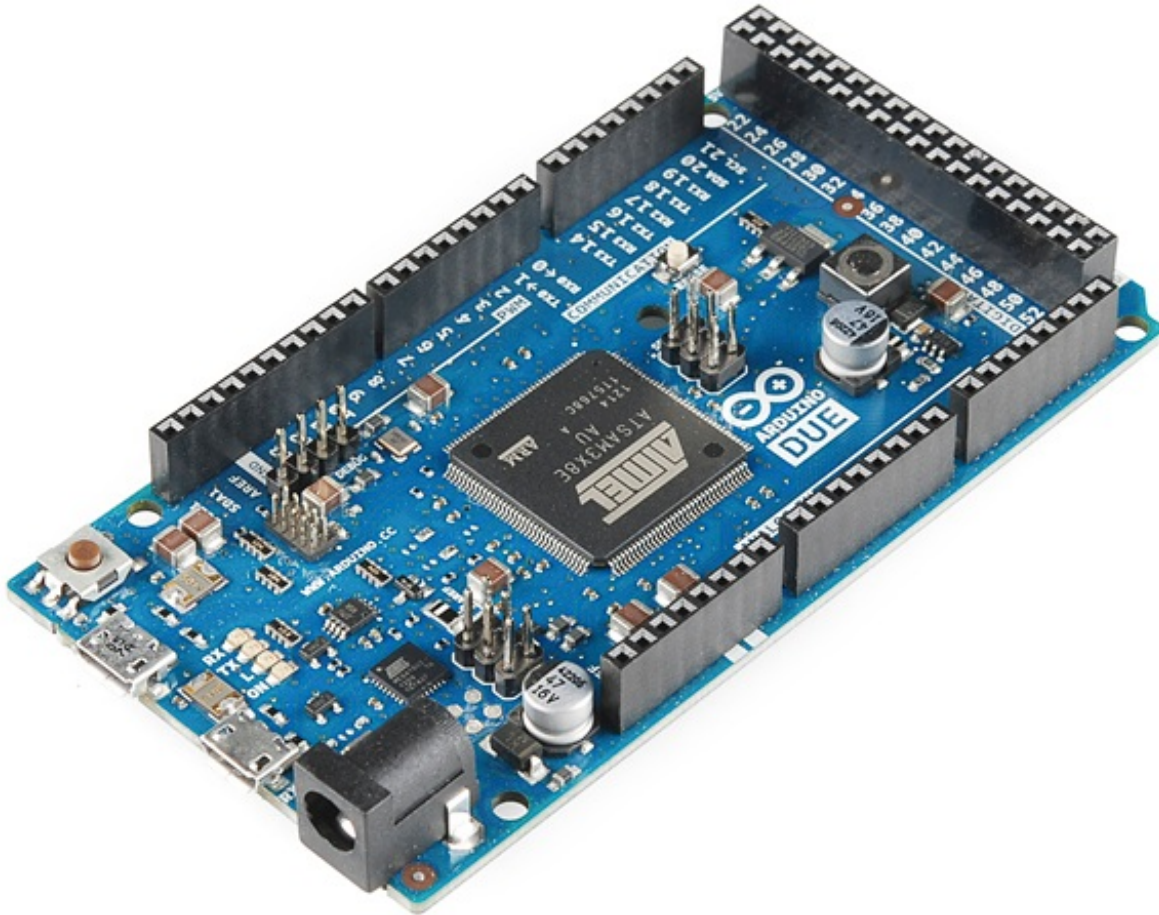
Like other Pro Arduinos before them, the Mega Pros all require an FTDI Basic for programming.

Arduino Due: Arduino Harder

You thought the Mega was powerful? TheArduino Due is a revolutionary take on the Arduino platform. It sports an entirely different processor architecture – ARM instead of AVR. It's a 32-bit processor, clocks in at 84 MHz, and has native USB support.

This thing sports many unique features that other boards don't have. Stuff like:

- Two **digital-to-analog converters** (DACs), which allow the board to output true analog values (instead of PWM). This means you can play audio out it!
- **USB on-the-go (OTG)** capability allows the Due to act as both a USB device and a host. So you can hook up other USB devices – like flash drives, WiFi modules, or phones – *to* the Due.
- **Direct Memory Access** (DMA) allows the microcontroller to offload memory-access tasks, so it can perform other operations at the same time.

There are also some new things to watch out for. The Due's processor – an ATSAM3X8E – can't work at 5V, so the board only runs at **3.3V** This means it may not be compatible with all shields.

The Due has some amazing functionality, but it's also a more**advanced** board. It's not recommended for beginners, but if you have a project that might take advantage of the Due's unique characteristics, check it out!

Arduino Yún

The Arduino Yún has true micro*processor* power. It actually runs Linux! It works over WiFi! It's awesome.

The Yún combines an ATmega32U4 (the same MCU on Leonardos) and an embedded Linux machine running the OpenWRT Linux OS. The two processors are "bridged" together. You program the ATmega32U4, just like you program an Arduino Leonardo, over USB. The 32U4 can send commands to the Linux processor, just as you might send commands to your Linux computer in a terminal.

This board is packed full of all sorts of neat features. It can be programmed over-the-air, as long as your computer is on the same WiFi network. It's designed to easily interact with the Internet, so if your intend on making an **"Internet-of-Things"**-type project, the Yún may be the perfect Arduino.

Resources and Going Further

Have you picked the perfect Arduino for your project? Looking for more Arduino tutorials? Check these out:

- Installing Arduino – Gotta get the software and drivers installed before you can start programming!
- Data Types in Arduino – Learn the difference between int, char, float, and long.

- **Arduino Shields** – Give your Arduino a little extra something with a shield. You could give the Arduino GPS, or allow it to communicate wirelessly with other devices.
- **E-Textile Basics** – If you're intrigued by the possibilities of the wearable LilyPad Arduinos, check out this tutorial to learn the basics of e-textiles.