

An Arduino

The Arduino microcontroller platform has been widely embraced by both young and old as an easy way to prototype digital electronics. The Arduino ecosystem has grown to be both extraordinarily popular and incredibly diverse. Tim Blythman explains where Arduino came from and where it is headed.

While strictly the name of a company which owns the 'Arduino' name and trademark, in practice, the term Arduino is used to describe the open-source hardware and software for which Arduino is known. There is a vast community of people spread around the world who have helped make Arduino what it is today.

Believe it or not, the name "Arduino" actually comes from a bar in the Italian town of Ivrea; the bar, in turn, appears to be named after an Italian king, Arduin, from over 1000 years ago!

The official Arduino website is at www.arduino.cc/

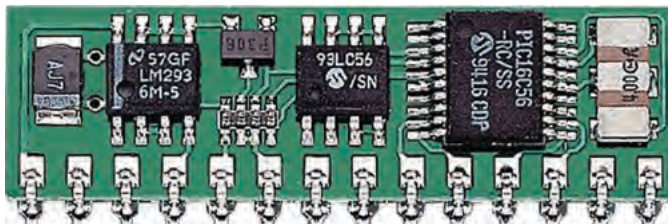
Before reading further, you might like to glance at our Arduino Jargon Guide panel.

History

The first inklings of what we now know as Arduino began around 2003, at the IDII (Interaction Design Institute Ivrea) in Italy, under a project called Wiring. Wiring was intended to allow design students at the Institute to create digital electronics projects, despite not having an engineering background.

Part of the goal of the Wiring project was to find a cheaper alternative to the BASIC Stamp microcontroller.

We published an article on the BASIC Stamp back in January 1999 – see siliconchip.com.au/Article/4630 It is a small PCB fitted with a microcontroller and EEPROM. It has a SIL (single in-line) header and can be plugged into



The BASIC Stamp was remarkable for its time, but it required a separate programming cable. The compiler and bootloader are proprietary, meaning it was difficult for third parties to create and develop tools for it. It has now been genuinely eclipsed by systems like the Arduino and Micromite.



The town of Ivrea is crossed by the Via Arduino. It is not far from Turin, where the early Arduino boards were manufactured. In 2018, Ivrea was declared a UNESCO World Heritage Site and an Industrial City of the 20th Century.

a breadboard. Back then, it cost around \$80 and required an extra \$20 programming cable.

PC software was needed to compile and upload a BASIC program (up to 100 lines long) via the programming cable. A separate editor program was needed to write the code.

The Wiring concept consisted of a microcontroller board and an IDE (integrated development environment) based on the Processing language. The IDE would combine the editor, compiler and uploader into one program.

The Processing language is intended to allow people who are not familiar with programming to create graphi-

Retropective

cal software. We used the Processing language to create patterns for our Stackable Christmas Tree in December 2018 (siliconchip.com.au/Article/11333).

One vital element which set Wiring apart from other platforms was to be open-source from the start. This allowed people to take the original idea, develop it further and put their own twist on it.

The Arduino platform is thus a 'fork' of Wiring. In fact, Wiring still exists and can even be used to program an Arduino Uno. The software and hardware designs can be downloaded from <http://wiring.org.co/download/>

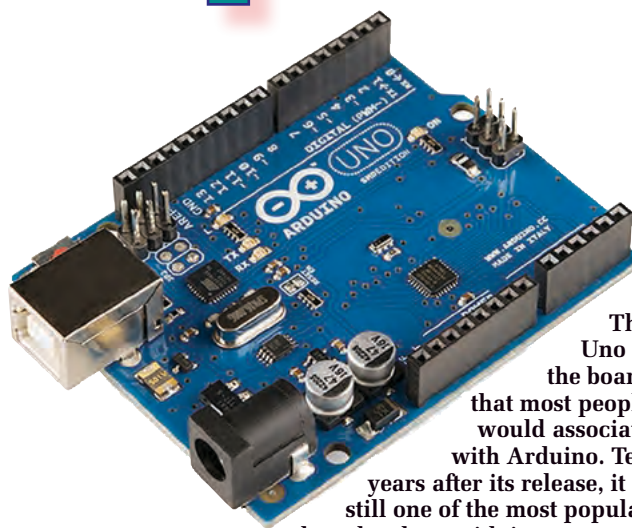
The clear advantages of the Arduino platform over the BASIC Stamp, apart from cost, include:

- not requiring an external programming cable or separate editor software;
- allowing considerably more complex programs to be created;
- better performance; and
- more features.

Hardware evolution

When the name Arduino comes up, most people would immediately think of the Uno board. A board which looked a little like the Uno first appeared around 2005.

A few years after that, a board called the "Arduino Serial" appeared. Practically all of the familiar Arduino pins are present in that layout, although



The Uno is the board that most people would associate with Arduino. Ten years after its release, it is still one of the most popular boards, along with its numerous clones and variants. Its ubiquity is no doubt enhanced by the fact that it is an open-source hardware design.
Source: Sparkfun Electronics.

PC communication is via a DE-9 multi-pin serial connector instead of the later USB port. Interestingly, the order of the ADC-capable pins is reversed, compared to current boards.

The DE-9 serial connector is not capable of supplying any useful amount of power, so an external power supply feeding the DC jack (or other pins on the board) was needed.

These earlier variants used an ATmega8 microcontroller. This is pin-compatible with the ATmega328 used in the Uno, although features such as PWM are missing from some pins. The ATmega8 also has less flash memory, EEPROM and RAM than its

successor.

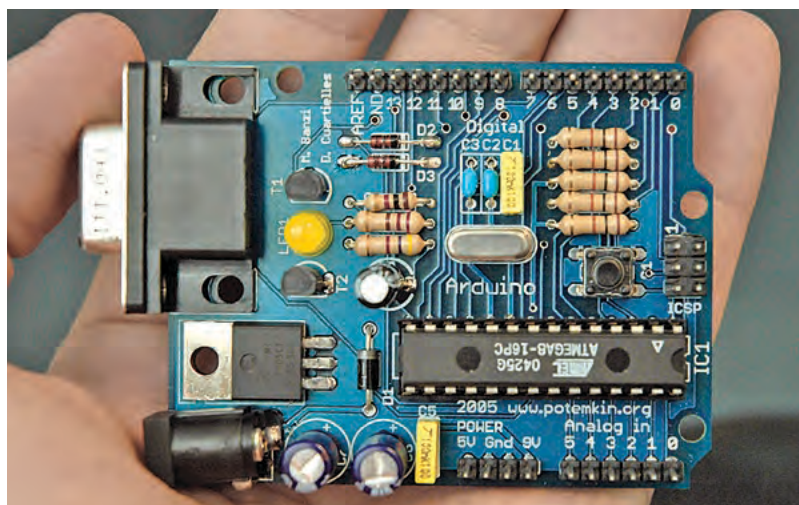
The design files for that version of the Arduino Serial are still available, so it can be made at home, if you have the facilities to make a single-sided PCB. The files are available at: siliconchip.com.au/link/aaxq

The Duemilanove (Italian for "2009") came not long

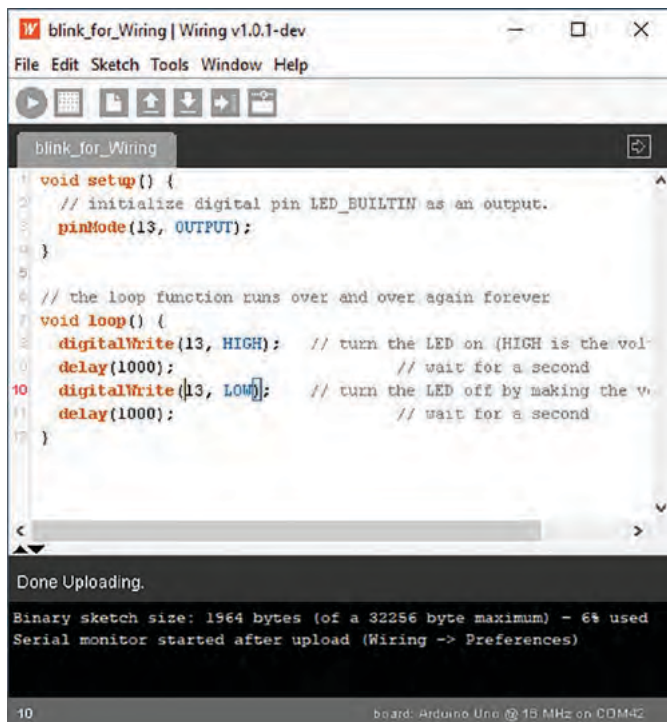
after. It looks very much like the Uno. In fact, it is practically interchangeable with the Uno in most cases, having the familiar USB connector and the now-standard ATmega328 processor, although variants with an ATmega168 also exist.

As this board could be programmed and powered over USB, it was now possible for a beginner to program a microcontroller with no extra parts.

The following year saw the release of the



The Arduino Serial looks a lot like an Uno board but lacks the USB socket. The processor is an 8KB ATmega8 rather than the 32KB ATmega328. Curiously, its analog pins are in a different order from the Uno.



The Wiring IDE is almost but not quite the same as that for the Arduino IDE. Both can program the Uno; the Wiring IDE also supports a variety of other boards.

Uno board and subsequently, in 2011, the first official release (version 1.0) of the Arduino IDE.

Nearly ten years later, the Uno (and its clones) are still among the most popular boards to be used with the Arduino IDE.

Software support

The code used to program an original Wiring board would probably be indistinguishable from that used to program any other Arduino board nowadays. It is the development of the IDE that has spurred the Arduino phenomenon the most.

The IDE hides a lot of the 'difficult' side of microcontroller programming. Features such as port and pin allocations and device-specific quirks are hidden away, so that inexperienced users do not have to worry about them.

This also means that boards with different processors (such as the Uno and Leonardo) can be used almost interchangeably. It is probably the areas in which they differ which are the greatest source of frustration for beginners!

Some people might complain that this abstraction hides a lot of what really goes on behind the scenes, and also

Using Arduino			
+	Installation & Troubleshooting For problems with Arduino itself, NOT your project Last post: Today at 10:47 pm by JustW32 microsystems by Dobry	106,517 Posts	34,418 Topics
+	Introductory Tutorials Tutorials for new people on the forum. Last post: Nov 23, 2019, 09:10 pm by GrittingBoardsWeb by hathcockd	697 Posts	186 Topics
+	Avrduino, stk500, Bootloader issues Problems related to uploading your compiled sketches Last post: Today at 10:24 pm by Piering troubles by mroldred	6,482 Posts	1,228 Topics
+	Project Guidance Advice on general approaches or feasibility Last post: Today at 11:27 pm by Antonio Uno is Arduino by Bub_Sager	567,903 Posts	75,986 Topics
+	Programming Questions Understanding the language, error messages, etc. Last post: Today at 11:47 pm by Vitaly long resolves by Vitaly8982	782,753 Posts	96,894 Topics
+	General Electronics Resistors, capacitors, breadboards, soldering, etc. Last post: Today at 11:52 pm Learning an exciting lot by mrmiscar	251,177 Posts	24,757 Topics
+	Microcontrollers Sensors or alternative microcontrollers, in-system programming, bootloaders, etc. Last post: Today at 11:11 pm by Discontinued programming by Klaus_K	52,407 Posts	11,563 Topics
+	LEDs and Multiplexing Controlling lots of inputs and outputs Last post: Today at 10:19 pm by digital480 clock by Paul_B	72,116 Posts	9,074 Topics
+	Displays LCDs, OLEDs, PAL, NTSC, etc. Last post: Today at 11:09 pm by Led matrix spinning by FelixG	85,680 Posts	11,591 Topics

The Arduino forum has had over one million posts on over 200,000 topics dating back nearly ten years. Many of the people who have contributed to the Arduino software and IDE are on the forum answering questions. See: <https://forum.arduino.cc/>

reduces performance. But these are probably not the people that would get the most from using the Arduino IDE.

In any case, the features which are available through the IDE are fairly broad and suit a great many applications.

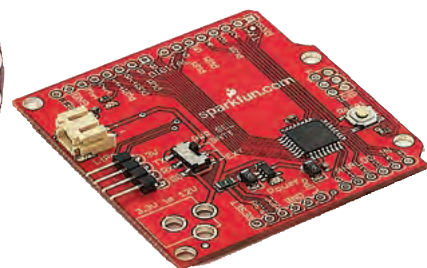
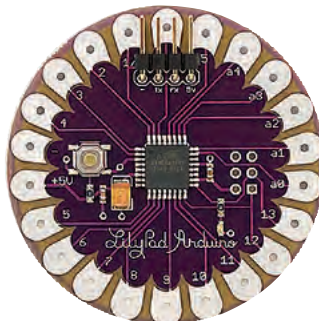
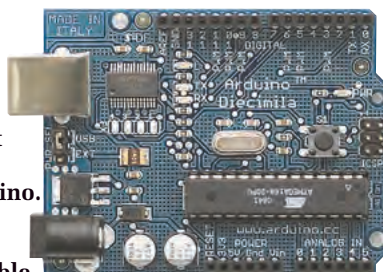
We'll get into more detail about this shortly, but Arduino has grown well beyond its original hardware and is no longer restricted to Atmel AVR-based microcontrollers. Hence, newer versions of the IDE have a Boards Manager and a Library Manager, making it much easier to target diverse hardware and accessories.

The open-source community

While the software and hardware behind Arduino are very tangible, there are some intangibles which have boosted its success. A large part of this is the massive community which does all sorts of things, from developing libraries to answering questions on the forums and more.

The Arduino forum is a great example of this. A web search relating to a specific Arduino problem will most often locate a forum post about someone else having the same problem many years ago (and hopefully, a solution!).

In 2008, Ocean Controls introduced the Diecimila USB, Skinny Board and Lilypad. This was probably the first time that many SILICON CHIP readers became aware of the Arduino. Ocean Controls continues to stock many SparkFun-branded Arduino-compatible products, plus some genuine Arduino boards.



Arduino jargon dictionary

Much of the language used when describing Arduino designs comes from the electronics community and C/C++ programming. Still, there are some other terms which are unique to the Arduino world. Here is an explanation of some of those.

analogWrite(): A function which configures a pin to deliver a PWM waveform. While not a true analog output on most boards, it has a similar outcome when applied to a motor or LED. PWM signals can be fed through a low-pass filter to more closely approximate an analog output. Some later boards have true analog (DAC) outputs.

Board: Usually used to describe a PCB fitted with a specific microcontroller and support components, plus a well-defined I/O pinout matching some reference design. Sometimes may refer to a bare microcontroller on a breadboard.

Core: A core is the set of files needed to support a family of Boards (see above), and can be added to the Arduino IDE via the Boards Manager. It usually includes a compiler, I/O pin profiles and a tool to upload to those Boards.

digitalRead(): This function returns the logic level on a digital input pin.

digitalWrite(): A function which sets a given digital pin to a logic value (0=LOW, 1=HIGH).

Library: A collection of files which provide extra functions to a sketch, by defining functions and other features. Often they are written to work with a Board's specific features and may form part of the Core.

MKR: The MKR series of Arduino boards have a standard form factor that is well-suited to breadboards. Most of these use 3.3V I/O signalling levels and have an ARM processor, which helps to differentiate them from the older 5V R3 AVR-based boards. Several MKR form-factor shields also exist.

Nano: A smaller, more breadboard-friendly form factor than the Uno. This includes the original Nano, the Nano Every and the Nano 33 series.

R3: The R3 form factor describes the most common Uno boards. This includes 32 pin sockets, of which 20 provide I/O functions. The most significant change since the R2 is that the hardware I²C pins are duplicated at a fixed location, independent of the I/O pins to which they map. Many clone shields do not follow the R3 form factor and may not work as intended with boards other than the Uno. Mega R3 and Leonardo R3 boards are generally compatible.

Serial Monitor: A basic serial terminal utility built into the Arduino IDE, which can be used for debugging, displaying program output or sending data to the attached device.

Serial Plotter: The Serial Plotter in the IDE receives serial data from the Arduino as numeric values separated by commas (as though it were CSV data), and displays it as a plotted graph with coloured traces.

Sketch: The Arduino name for what many people would term a program. It derives from the Processing language's graphic design background and its use among artists and designers, similar to the notion of a drawn sketch.

Uno: The Italian word for "One" and the name of one of the most popular Arduino based boards; you may even hear the word "Arduino" used to refer to the Uno. It has an ATmega328 IC with 20 I/O pins.

Upload: This describes the process of transferring a program from a computer to the target board after compiling. This is typically done via a USB-serial device, although some boards support Bluetooth or WiFi.

Verify: Unlike other microcontroller platforms, where this term usually means to check that the program uploaded to the target board matches that stored on the host computer, the verification process under Arduino simply checks that the sketch compiles correctly.

The forum can be found at: <https://forum.arduino.cc/>

There are any number of tutorials and how-to guides on it. There's a good chance that, if you have an idea, somebody has already attempted it and posted about it on the forums.

The Arduino hardware and software is not so different from that of the BASIC Stamp, PICAXE or even the Micro-mite. But one major distinction is the open-source nature of Arduino. See our panel for more about how open-source works and why it has had such a large effect.

Arduino and SILICON CHIP

We first made mention of an Arduino-compatible board in 2008, when a Diecimila ("10,000" in Italian) board appeared in Product Showcase, courtesy of Ocean Controls. Not long after this, readers started reporting their experiences with Arduino via letters in the Mailbag section.

The January 2012 issue saw a detailed review of the Arduino platform and its associated hardware (siliconchip.com.au/Article/806).

We now have over 600 articles that mention Arduino in some form or another, including dozens of projects that either use an Arduino board, or are designed to work with one.

There is still a continuous stream of new Arduino hardware rolling out. Last year, we reviewed three new Arduino main boards. We covered the MKR Vidor 4000 in March (siliconchip.com.au/Article/11448) and two Nano variants in October (siliconchip.com.au/Article/12015). The IDE software is also continually being updated.

Those boards are 'official' Arduino releases, but many other shields and modules are also being released, and a great many third party boards are being created, too. There is even work going on to allow other microcontroller boards to be used with the Arduino IDE, as well as several different IDE variants to cater for different users.

The Boards Manager

As mentioned above, the original Arduino boards were based on Atmel AVR microcontrollers. The Due changed this, bringing a 32-bit ARM processor to the Arduino world. This required changes to the IDE, to support different compilers and uploaders.

Incidentally, Due means "two" in Italian; a clear indication that this was something different to the Uno ("one").

While the open-source nature of the IDE allows people



The Altronics K9660 TFT Touchscreen Maker Plate uses the same 32-bit SAM3X8E ARM CPU as used in the Arduino Due to drive a TFT display. It fits in a standard electrical wallplate, changing it from a development platform into practically a finished product.

to modify the Arduino IDE to suit other boards, this is not a straightforward process, and it was evident that another solution was needed.

This was solved when the IDE version 1.5 was introduced, which added the Boards Manager. Now, different architectures could be easily supported, and the multitude of board and processor types became possible.

Version 1.5 also brought streamlined installing of libraries. Finding a library to work with a new module is now simple, as you just need to run a search in the Library Manager.

This has access to a well-maintained and comprehensive list of libraries; the necessary files are downloaded and installed with a click, often including example sketches.

Hardware evolution

The software evolution of the Arduino IDE has been predictable. Steady improvements to the IDE have continued to make programming easier for a widening audience, while maintaining continuity and uniformity for existing users and software.

However, the hardware evolution has been rapid. There is now a vast array of hardware that can be programmed using the IDE. Even experienced microcontroller aficionados such as ourselves are amazed by the convenience and features that are on offer.

This makes it easy to see what can be done with new hardware without having to re-learn anything on the software or coding side. We think that this is a great feature that suits even very advanced users.

The ESP8266

One processor family that has seen a lot of use by being accessible under the Arduino IDE is the Espressif ESP8266, and subsequently its successor, the ESP32. Their ‘killer



The Altronics Z6360 is practically identical to the first ESP8266-based boards that began to appear around five years ago, labelled “ESP-01”. We reviewed these and used one for the “Clayton’s GPS” project in April 2018. Many variants have appeared, and most of them can now be programmed using the Arduino IDE.

feature’ is integrated WiFi and, for the ESP32, Bluetooth, at a very low cost.

We saw the first examples of this hardware just over five years ago. Now they are used even in many consumer goods (which keen Arduino users are pulling apart and reprogramming).

The first of these ESP8266 modules went on sale with little to no documentation, except as WiFi modules controlled by AT commands over a serial port. At about USD \$5 each, many people snapped them up just to try them out. At the time, even an Ethernet shield cost many times more than that.

Soon enough, projects involving LEDs being controlled through basic web-pages abounded. DIY home automation using WiFi seemed achievable.

While the AT interface worked, the serial port limited the speed. It wasn’t long before a small community popped up with the intention of getting the open-source GCC compiler to program the microcontroller on this module.

This microcontroller is a 32-bit Tensilica Xtensa LX3 running at 80MHz; the program is typically stored on an external flash chip of at least 512kB. This had the potential to be much faster than the Uno at a lower cost; never mind that it had onboard WiFi!

Espressif noticed the popularity of their modules and released some tools to allow code to be compiled for the ESP8266; no doubt, this helped the Arduino community.

So less than the year after the ESP8266 appeared, it became possible to program it with the Arduino IDE, although a lot of manual file manipulation was necessary back then to enable it.

This was resolved with the Arduino IDE v1.6.2 with the simplified installation of new boards and libraries. This support was tweaked in v1.6.4, which we now consider the oldest version suitable for new projects.

In response to the above, a flurry of new Arduino-compatible boards appeared using the ESP8266 chip. These borrowed many of the features that made the original Uno so popular. Probably the best-known example is the WeMos D1 R2, which was subsequently cloned due to its popularity.

This is sold in our Online Shop (Cat SC4414), and we’ve used it in a few of our projects.

Typical features include a USB connector that allows the unit to be powered easily, while a USB-serial IC provides a channel for communicating, debugging and uploading code. Some clever circuitry on the board automatically detects when code needs to be uploaded, so the entire process is as seamless as it would be for an Uno.

Nowadays, the ESP8266 is one of the most popular Arduino IDE addons. We even use it when we want to quickly test out a 3.3V part or module, since we’re so familiar with it.

What is open sourcing?

While open-source software and free software are not quite the same, they often coincide. One could argue that the Arduino IDE is popular because it is free, but it has continued to develop because it is open-source. In fact, the first mention of open source in SILICON CHIP was when Ocean Controls introduced the Decimila.

The notion of open sourcing is fairly new (around twenty years old). Although it may appear at first to be a strange business model, it has been successful for several companies and individuals.

The simplest definition of open-source software is that it is software where third parties can legally download the source code. Usually, the tools to turn the source code into a working program are also freely obtainable.

In a sense, the movement was a reaction to the very 'closed' models of early software companies. This often led to computer users being saddled with glitchy software, where the originating company wasn't interested in fixing it, and the users couldn't.

Another important element in the development of the open-source model is the observation that digital objects such as code can be copied without requiring material resources; they simply exist as bits and bytes on a storage medium such as a hard drive.

So open-source software was devised as a way to release software so that others (including users) could assist in its development, and bug-fixing, but (in some cases) still allowing the authors to make money or otherwise benefit from their hard work.

Open-source hardware exists, but is nowhere near as common as open-source software. Maybe this is because it's unusual to get hardware for free! In any case, open-source hardware usually just refers to the design; in other words, 'some assembly required'.

Because of the nature of copyright, you are not automatically permitted to make copies of software. Thus, numerous open-source licenses exist. The simplest of these is to simply declare the code to be 'public domain', which means that there are basically no restrictions placed on its use.

But it's more common to see source code released under either the GPL (GNU Public License) or with a BSD-style license. The BSD-style license is only slightly more restrictive than public domain, while GPL places more strict restrictions on how the software may be redistributed. You might even see the term 'copyleft' applied to some of these licenses, to highlight the contrast with copyright.

The smart part of many open-source licenses is that there is a condition that any derivative works must be released under a similar license to the original software. This keeps the open-source software open.

How does Arduino fit in?

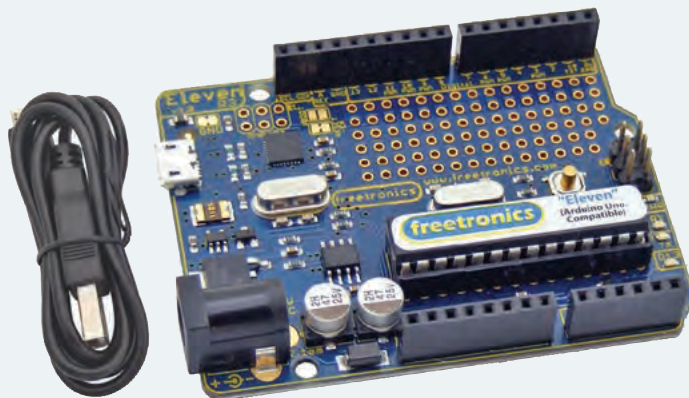
The Arduino IDE code is available under a GPL open-source license. Also, it depends on several open-source tools to work.

This includes the "avr-gcc" compiler (AVR GNU Compiler Collection), which takes the C/C++ code in the Arduino sketch and turns it into machine code to run on the microcontroller. Then there's "avrdude" (short for AVR uploader/downloader), which loads the compiled machine code into the target processor.

Arduino open hardware

Many official Arduino boards are also fully open-sourced hardware. For example, the Uno circuit and PCB layout are available under a Creative Commons Attribution Share-Alike license.

This has had a two-fold effect. The first is that it allows (generally lower-cost) clones to be sold. For the most part, these are practi-



cally identical and thus utterly interchangeable with the 'genuine' Arduino boards. This makes for a very low barrier to entry to the Arduino system.

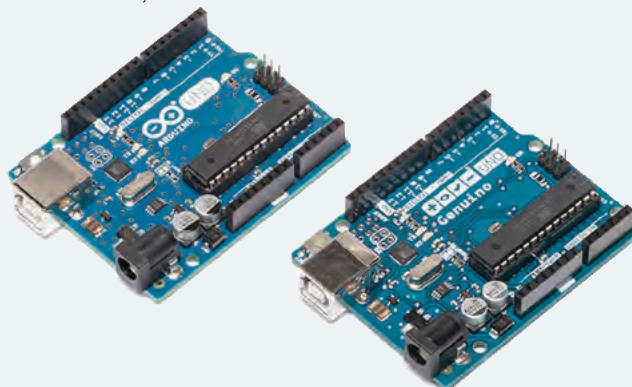
Secondly, improved versions of the original hardware have appeared too. Firms such as FreeTronics have created boards like the Eleven, which is an improved but still compatible version of the Uno.

Having downloaded the design files for the Uno or Eleven, you could build your own copies, or even improved versions.

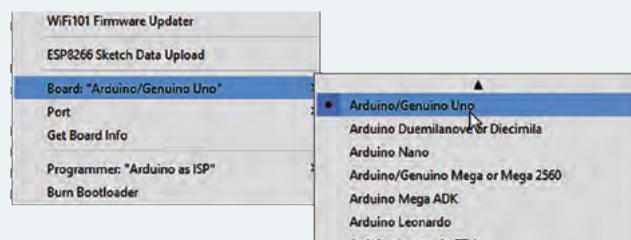
Trademarks and disputes

While Arduino software and hardware are open-sourced, the Arduino name itself is a trademark, and this has been the focus of at least one controversy.

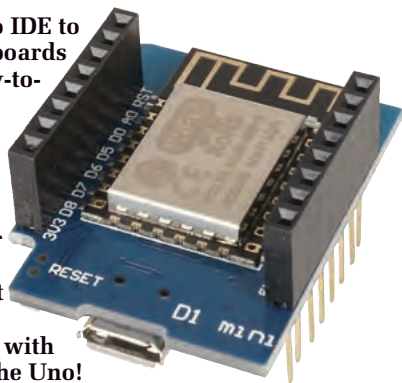
The short version is that different people registered the Arduino trademark in different parts of the world. So some people trying to sell genuine Arduino boards could not use the Arduino name in some parts of the world. Therefore, they had to come up with another name, "Genuino".



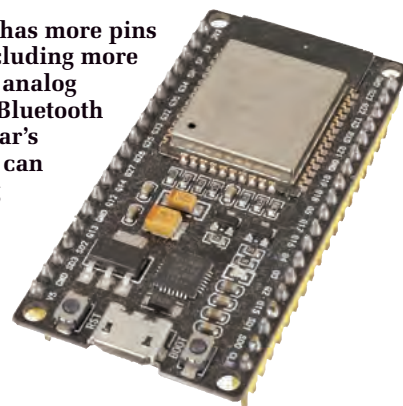
Eventually, the two groups merged, and for the most part, the Genuino is now part of history. In any case, the Arduino trademark and name appear to be valuable. This is why so many of the clones have other different names ending in -duino; simply because, legally, the Arduino name is otherwise off-limits. Instead, we have the term 'Arduino-compatible' to describe anything else.



The ability of the Arduino IDE to program ESP8266-based boards spurred the design of easy-to-use hardware. Jaycar's XC3802 WiFi Mini combines an ESP8266 module with a USB-serial converter, a voltage regulator and some clever hardware to allow firmware to be loaded without user intervention. Thus, a very capable 32-bit micro with WiFi is as easy to use as the Uno!



The ESP32 processor has more pins than the ESP8266, including more which can be used as analog inputs, and supports Bluetooth along with WiFi. Jaycar's XC3800 (shown here) can be programmed using the Arduino ESP32 Boards Manager add-on.



The ESP32

Noticing the popularity of the ESP8266, Espressif addressed several shortcomings that had been noted by the Arduino community when they created the dual-core ESP32.

One core can be dedicated to radio functions while the other core is free for functions needing a real-time response. The 2.4GHz radio of the ESP32 can even be used to implement Bluetooth, another compelling feature. Arduino support for the ESP32 came quickly.

The ESP8266/ESP32 Arduino support is a fantastic example of what an open-source community can achieve when allowed to put WiFi support into the hands of the masses.

And the rest

The support for other micros doesn't end there. Our ChipKIT Lenny review in last month's issue describes how a Microchip PIC32-based board can be programmed under the Arduino IDE. It compares favourably with other official Arduino branded 32-bit boards in performance, although some functions and features don't quite work the same.

There is a good list of third-party processor boards which can be added to the Arduino IDE at: <https://github.com/arduino/Arduino/wiki/Unofficial-list-of-3rd-party-boards-support-urls>

Some of the so-called 'boards' added by the Boards Manager are sometimes no more than a bare microcontroller, relaxing the requirement to use Arduino-compatible hardware. It isn't even that hard to add support for custom

hardware.

Note that not all of the 'core' libraries may be fully implemented for each board, as it is up to the board designer to write those libraries. This includes support for serial protocols like I²C and SPI.

From small...

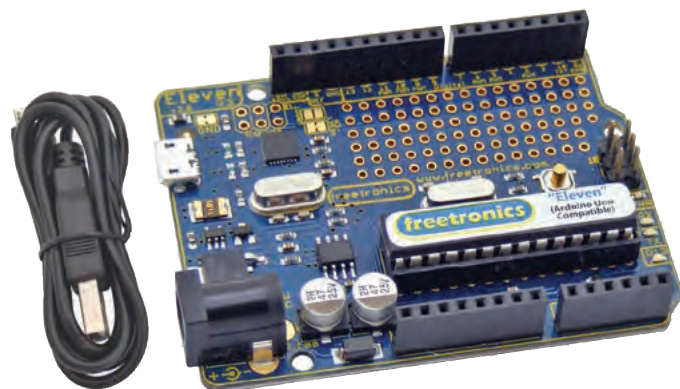
Newer boards such as the Nano Every (reviewed in October 2019; see: siliconchip.com.au/Article/12015) use the recent megaAVR series ATmega4809, which incorporates features from some Microchip microcontrollers since their takeover of Atmel.

There are a good number of other Atmel microcontrollers which have been made to work with the Arduino IDE, and they have been fitted to a surprising number of development boards.

For example, there is the ATtiny series; the ATtiny85 is an eight-pin device that can be had in a DIP, SOIC or even QFN package. Because of their similarity to the ATmega processors, adding full support for these micros to the Arduino IDE is not all that difficult.

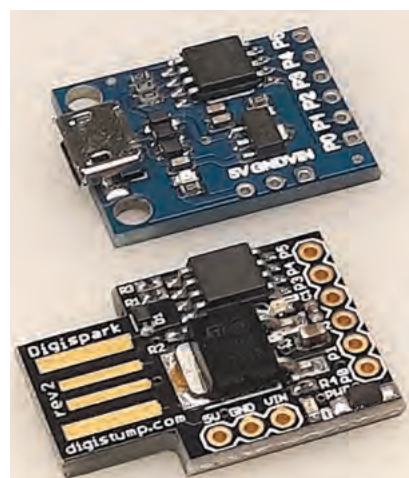
One of the more interesting ATtiny boards is the Digispark (<http://digistump.com/products/1>), which looks like a small USB drive. It fits an SOIC-package ATtiny85 alongside a 5V regulator and breaks out six I/O pins.

The ATtiny85 does not have a native serial UART, let alone USB support, but the Digispark is loaded with a cleverly-written USB bootloader which operates entirely in software. The bootloader uses the HID protocol normally used by keyboards and mice. This is enough to allow sketches to be uploaded with minimal effort via a conveni-



The Freetronics Eleven is a great example of the benefits of open-source hardware. It is an improved version of the Uno, adding a prototyping area and replacing the full-size USB socket with a smaller micro-USB socket. Thus, there is less chance of the socket shorting out against a shield. Its design files are available on similar open-source terms to the Uno's.

The Digispark is one of the smallest Arduino-compatible boards. With an ATtiny85 microcontroller, it's the size of a small thumb drive and has six I/O pins, enough for many small projects. Its design files are open source, and clones have naturally appeared. The black board at right is the original Digispark; the blue board above it is a clone.



What's a bootloader?

One of the reasons Arduinos are so easy to program is the bootloader. This is a small program on a microcontroller which allows larger programs to be uploaded. Note that this is not unique to the Arduino world.

The “boot” part comes from the term ‘bootstrapping’, which refers to the notion of lifting oneself up by one’s own bootlaces or, less figuratively, without outside help. Nowadays, the term ‘booting’ is used to describe a piece of electronic equipment starting up.

Practically all computers go through a similar process. A PC has a small program in a ROM chip on its motherboard, which in turn loads another program from its hard drive into RAM (which may, in turn, load another program). Without this small program in ROM, a computer would not be able to start up.

But since microcontrollers typically have non-volatile memory onboard, unlike a PC, this process does not need to occur every time a microcontroller starts up.

Every time an Uno is powered up or reset, the bootloader runs for the first second. It monitors the serial port, and unless it sees the correct sequence of data, it runs the program already stored in its flash memory.

If it detects a sequence which indicates that programming needs to occur, the bootloader continues to run, accepting data from the host computer and writing it to the internal flash memory. When programming is complete, the bootloader runs the freshly uploaded program code.

To program an early Arduino board, it had to be manually reset with a pushbutton. But now there is another microcontroller which detects when the host computer initiates a serial connection, and this triggers a reset automatically. This means that no action is required to load the sketch, apart from running the upload program on the PC. That is one factor which makes Arduino so easy to use.

Another clever point is that the bootloader resides in a protected part of flash memory, so it cannot overwrite itself or the configuration fuses. So it is tough to ‘brick’ an Uno through the normal upload process; another upload is usually sufficient to correct a faulty upload attempt, as the bootloader survives and runs at reset.

The bootloader used on the Uno is called “Optiboot”, which is a development of other open-source projects which sprung up independently of Arduino.

For more information, see: siliconchip.com.au/link/aaxo

ent USB interface.

The designer of the Digispark has also released his design files as open source. Unsurprisingly, it is now possible to buy clones of these handy little boards.

... to large and varied

In terms of the 32-bit boards which can be programmed by the Arduino IDE, we’ve mentioned the Microchip-based ChipKITs, the ESP8266 and ESP32 boards and Arduino’s own Atmel SAMD21-based MKR series boards. But there are others, many of which use the ARM (Advanced RISC Machine) Cortex-M0 architecture.

Some also use processors from Gecko and Infineon, or other ARM architectures. This includes the STM32-based development boards with ARM Cortex-M3 cores, and Cortex-M4 based boards with chips from Maxim Integrated and Nordic Semiconductor.

Nordic is known for their radio ICs and modules, such as the 2.4GHz nRF24L01 modules we covered in January 2018 (siliconchip.com.au/Article/10940). Some Nordic ICs have BLE (Bluetooth Low Energy) radios.

There are even several Arduino-compatible boards with Intel processors. An example is the Galileo, which has a 400MHz processor, 512kB of SRAM, 8MB of flash and a standard R3 pinout. It actually runs Linux to handle communications with a custom version of the IDE for managing sketches.


With the rise of open-source tools for FPGA development, people are even creating boards based on ‘softcores’.

A softcore is an implementation of a processor via programmable logic, most commonly an FPGA. While this may seem wasteful, it does give the ability to easily reprogram the device to add new features, or even to emulate a different processor.

If you are not familiar with FPGAs, refer to our introduction to the iCEstick FPGA development board in April 2019 (siliconchip.com.au/Article/11521).

Chips implemented as soft cores include the ATmega328, for example, as used in the Alorium XLR8 (www.aloriumtech.com/xlr8/) and the Lattuino (http://fpgalibre.sourceforge.net/Lattuino_en/index.html).

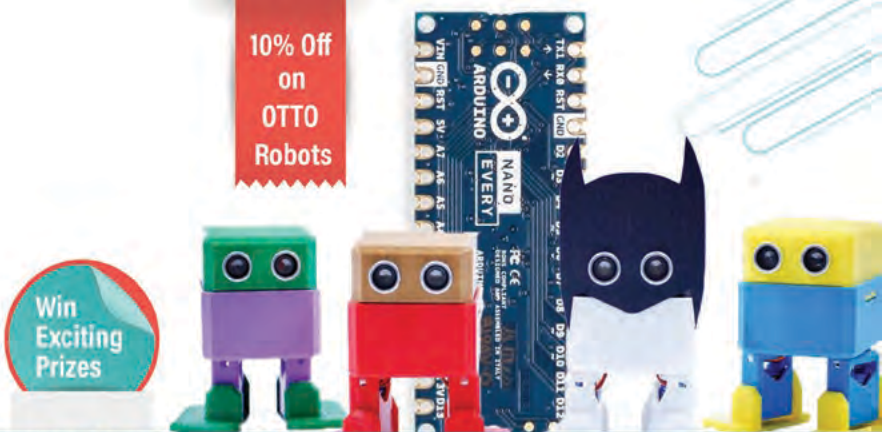
Many of these boards are used for education. So it’s



Pakronics® is Celebrating Arduino's® 15th Birthday!

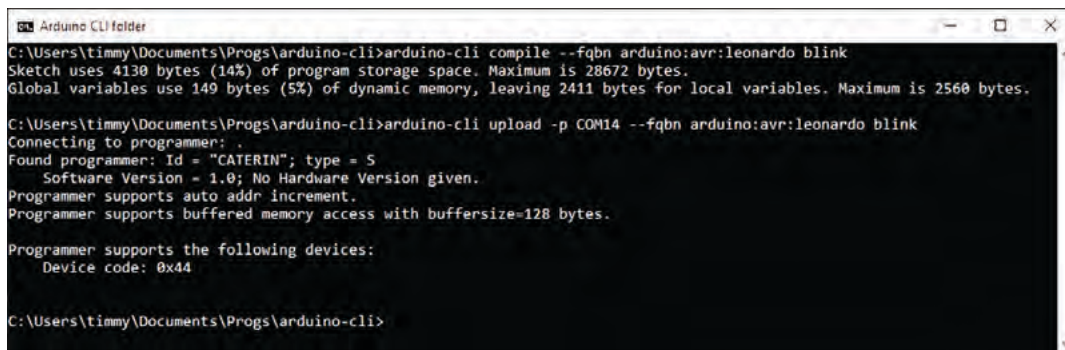
Let's Camera, Roll & Action!

To Participate, Register
@ <https://pakr.xyz/arduino2020>



<https://www.pakronics.com.au> inquiry@pakronics.com.au 1300 952 526 [f: @pakronicsaustralia](https://www.facebook.com/pakronicsaustralia) [t: @pakronics](https://www.instagram.com/pakronics)

Copyright © 2020 Pakronics® Arduino® Trademarks & Brand Logos are the Property of their Respective Owners



```
C:\Users\timmy\Documents\Progs\arduino-cli>arduino-cli compile --fqbn arduino:avr:leonardo blink
Sketch uses 4130 bytes (14%) of program storage space. Maximum is 28672 bytes.
Global variables use 149 bytes (5%) of dynamic memory, leaving 2411 bytes for local variables. Maximum is 2560 bytes.

C:\Users\timmy\Documents\Progs\arduino-cli>arduino-cli upload -p COM14 --fqbn arduino:avr:leonardo blink
Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
Software Version = 1.0; No Hardware Version given.
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
Device code: 0x44

C:\Users\timmy\Documents\Progs\arduino-cli>
```

The Arduino CLI is a command-line version of the Arduino IDE. Compiling and uploading a sketch is as simple as running the two commands shown here. It seems slightly quicker than the IDE, but we think its big advantage is its ability to use scripts to automate processes.

useful that one board can be used to teach both microcontroller and FPGA development, including concepts such as processor and ISA (instruction set architecture) design.

... and the new

We haven't seen many new main boards coming out with the classic Uno footprint. Apart from the Uno WiFi Rev2, most new Arduino boards use the MKR (pronounced 'maker') designation and footprint.

You can still buy the Uno from the Arduino online store (store.arduino.cc), but only third-party manufacturers are really developing the 'classic' footprint. Still, we don't expect it will go away any time soon.

Last year, four new Nano boards and several MKR-format shields were announced by the Arduino company on Arduino Day (March 16th). We expect to see more major hardware announcements from them on Arduino Day this year.

In contrast to the classic Uno and derivatives, the MKR boards are all 32-bit SAMD boards with 3.3V I/Os. There are several shields available with the MKR footprint, which is more breadboard-friendly than the Uno footprint.

Many of the new MKR boards have some form of wireless communication, including WiFi (Arduino MKR WiFi 1010), GSM (Arduino MKR GSM 1400), LoRa (Arduino MKR WAN 1300) and NB-IoT (Arduino MKR NB 1500). Interestingly, WiFi support on the MKR WiFi 1010 is an ESP32-based module, which has its firmware compiled under the Arduino IDE.

As we mentioned in our review of the MKR 4000 Vidor,

a cryptographic chip is also fitted to these boards, allowing for secure communications over these wireless networks. Many of these boards also have a header for a rechargeable lithium battery, and support charging the battery from USB power. Thus, they are well suited to remote or untethered deployment.

These boards have been around for a few years now, but we are not yet aware of any clones of them, although we imagine they would be popular if they were available.

Software for advanced users

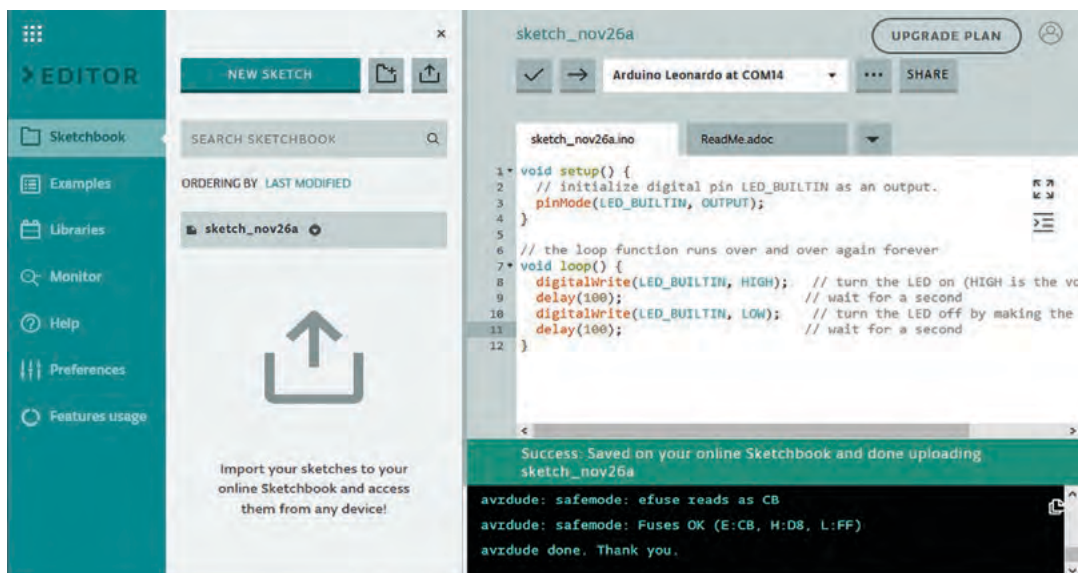
We noted above that some people who are experienced with microcontrollers might complain that the Arduino IDE hides too much. It's no doubt that the resulting simplicity is helpful for beginners, or even experts who are trying a new type of micro. But there are times when you need to know what is going on 'behind the curtain'.

There are two Arduino software tools which give users more control and power, and they are as follows.

Arduino CLI

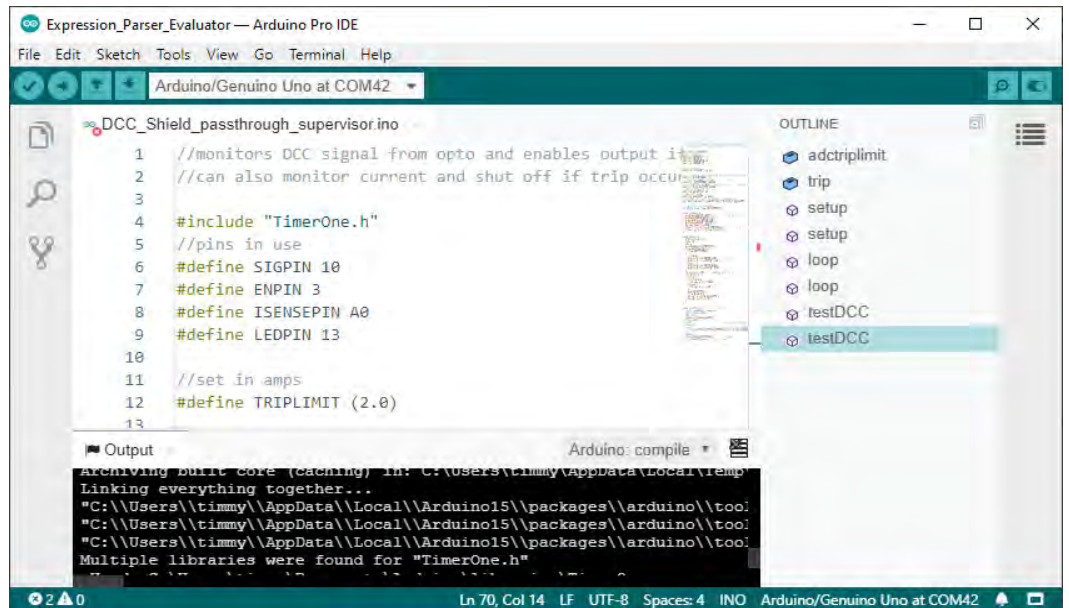
The Arduino CLI, released in 2018, is the first of these. CLI stands for Command Line Interface. As the name suggests, it allows the compilation and uploading processes to be controlled using a command line. You can write a sketch in a simple text editor, then use a CLI command to compile and upload it.

Many of these commands perform the same function as IDE menu options, but the lack of a GUI (graphical user



The Arduino Web Editor looks quite different from the desktop IDE, but many of the same features are present, and the process is much the same. Being accessible from a browser means that sketches can be viewed and edited from mobile devices, although uploading is not yet possible.

While only in the early stages of development, the Arduino Pro IDE has some promising features. The overview and outline at right allow easier navigation of large projects, and there are many more settings available in the Preferences menu.



interface) makes them run faster on slower computers. On our Windows 10 PC, the CLI is a 5MB executable file. It naturally needs the board cores and other files to work, but is certainly smaller than the full IDE.

Because it can be controlled from a command line, it lends itself well to scripting and automation. It could also be the basis of a custom IDE. See <https://github.com/arduino/arduino-cli> for more information.

Arduino Pro IDE

At the other end of the scale of complexity is the Arduino Pro IDE. This is currently only at the 'alpha' (pre-beta-testing) stage, but it appears to offer a lot more features than the standard IDE. In fact, it is a fully-featured development environment.

The Pro IDE gets new, experimental features which would only add confusion to beginners if they were added to the regular IDE. Some of the proposed features include live debugging, and the ability to use third-party plugins and different languages for programming. Eventually, some of these features may migrate to the basic IDE.

The Pro IDE relies on the Arduino CLI for core functionality. It can be downloaded from: <https://github.com/arduino/arduino-pro-ide/releases>

Note that it is still at a very early stage of development, so it is likely to have bugs and undergo significant changes as it evolves.

The 'cloud'

You probably won't be surprised to hear of a cloud-based version of the Arduino IDE. This lets you program an Arduino board without having the IDE installed.

A small program called the "Create Agent" needs to be installed, to communicate with the boards (since a web browser does not have access to serial ports). All sketches are saved online.

The Arduino Create website is at: <https://create.arduino.cc/> You need to set up an Arduino account to access it. This gives you 100MB of space to store up to 100 projects and allows 200 sketch compilations per day. There is also a paid plan which offers more features.

We tried it out and found the online sketch editor easy to work with, and were able to upload a simple sketch within minutes. Apart from the online editor, there are tools for getting started and a project hub where other Arduino projects can be viewed and shared.

There is also an IoT Cloud, which allows devices to be connected to the Internet; this is limited to a small number of Arduino boards from the MKR series, plus the Nano 33 IoT.

Perhaps this is a gentle nudge away from the older, cloned boards towards the newer devices.

One advantage of the online version is that less capable devices such as Chromebooks can be used to work with Arduino. For schools and other institutions that use Chromebooks, this means that they can teach Arduino without worrying about software downloads and installations.

Of course, we all know 'the cloud' is just another term for 'someone else's computer', and some people might object to having their programs stored there. But it could certainly be handy for working on your sketches while you aren't at your desk.

You can even access Arduino Create from a mobile device like a smartphone, although it doesn't yet appear to allow sketch uploads from these yet.

What next?

There is no doubt that Arduino has come a long way in the last ten years. And we don't expect it to disappear any time soon. The new developments in the Arduino CLI and Pro IDE show that the Arduino folks are willing to broaden their audience.

New Arduino-compatible hardware is announced regularly. The Arduino community around the worldwide will no doubt ensure that the Arduino phenomenon will continue, regardless of what happens with the Arduino company.

Arduino Day is coming soon, on March 21st (<http://siliconchip.com.au/link/aaaxt>), so look out for new announcements. If it is anything like the last few years, you can expect to see some new hardware if nothing else. We will find out soon what the future holds.

SC