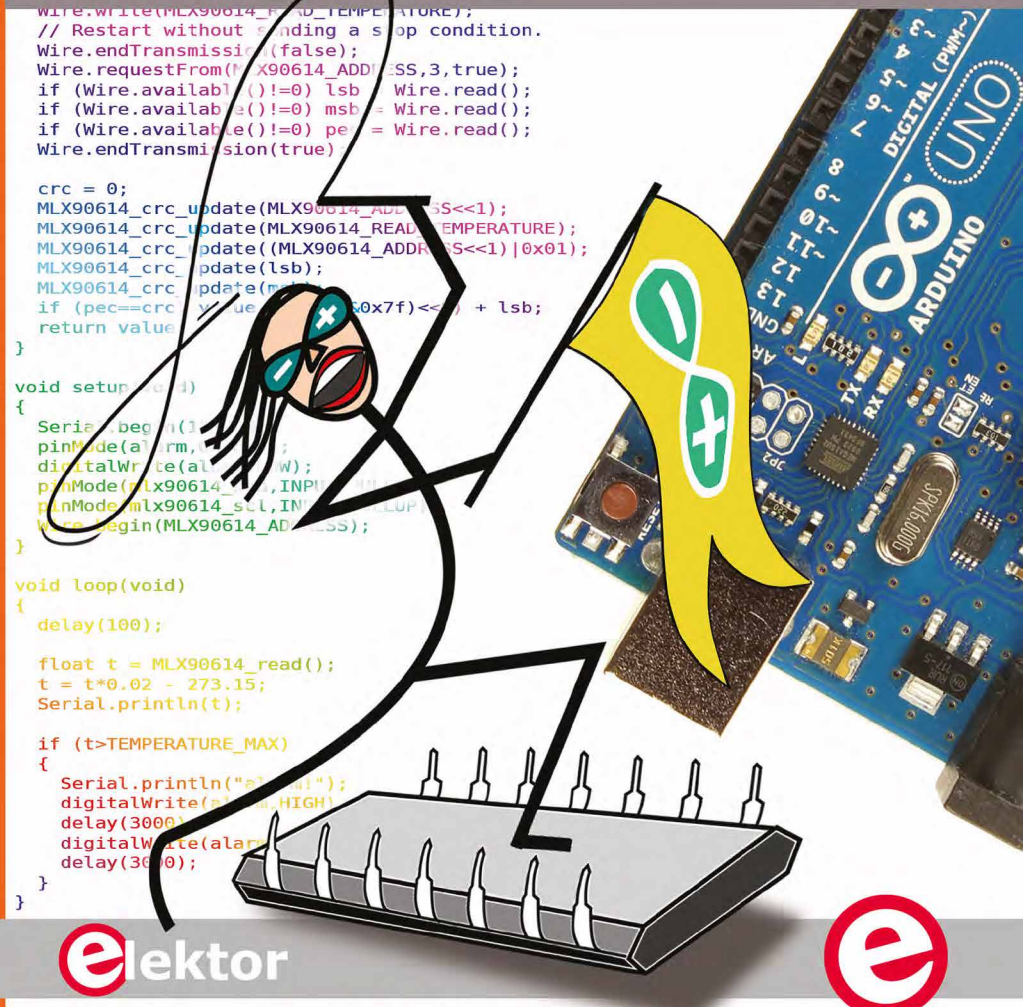
Clemens Valens

Mastering Microcontrollers

Helped By Arduino



elektor

www.elektor.com/arduino

Table of Contents

4. Rapid Prototyping Italian Style

Arduino, from Italy, is a rapid prototyping platform for microcontroller applications. This means that it is a set of tools developed to facilitate the design of microcontroller-based circuits without wasting too much time on learning the ins and outs of the platform. With Arduino, you too can do "microcontrollers" and "embedded electronics". *Now I'm gonna make you an offer you can't refuse*: get yourself one of those cheap Arduino boards, install the free software tools, keep your evenings free and I will teach you how to use it all.

4.1 The Godfather 1, 2 and 3

Arduino is related to the open-source techno-popularizing projects Processing from which it borrowed the programming environment and Wiring that provided the foundations of the code libraries.

Processing "*is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work.*" (source: `www.processing.org`)

Wiring, Italian like Arduino, is a programming framework for microcontrollers "*created with designers and artists in mind to encourage a community where beginners through experts from around the world share ideas, knowledge and their collective experience.*" (source: `www.wiring.org.co`)

Arduino (`www.arduino.cc`) was conceived at the same school as Wiring and is a simplification of the latter, which in itself was already quite easy to use. Originally Wiring was not open-source, but under the pressure from the Arduino team, Wiring gave in and published its source code. Then the Arduino team could get going.

The goal of Arduino is to make microcontrollers accessible to students, non-specialists, artists, designers, enthusiasts and all those who are interested in the creation of interactive objects and environments, but who do not have (well-developed) electronics and/or programming skills and who do not always have a lot of money. Therefore the Arduino team decided that their board should not cost more than $30 (€25), which was indeed the price of the first board. Since then prices have gone up marginally to reach about $35 (€30) in 2013.

The first Arduino boards were more basic than the Wiring boards, but over time the Arduino hardware became more elaborated, whereas Wiring extended its range by simplifying their boards. Now (in 2013), Wiring S and Arduino Uno boards are very similar.

Despite the equivalence of the two projects, Wiring did not meet with the same success as Arduino. From the early beginning the Arduino team has done everything to spread their word by publishing all source code, circuit diagrams, printed circuit board designs and detailed documentation for free, whereas the Wiring team tried to control everything. In the Fall of 2012 over 600,000 Arduino boards had been sold worldwide.

Arduino libraries are based on those from Wiring and the Integrated Development Environment (IDE), based on the one from Processing, is almost identical to the Wiring IDE. The only striking difference is its color: orange for Wiring and light blue for Arduino. Processing uses shades of grey for its IDE with looks very familiar to the two others. Other projects have started to use the Processing IDE as well and we can almost speak of an emerging standard.

The Wiring IDE supports Arduino hardware, but the Arduino IDE does not know about Wiring hardware. Initially Arduino limited itself to a few microcontrollers from Atmel's 8-bit AVR family while Wiring openly flirted with microcontrollers from for instance Microchip and Texas Instruments and also with MCUs based on an ARM core. Arduino too has been transformed into a multi-MCU platform, but a little "unbeknownst to its own free will". The Arduino philosophy has been embraced by many enthusiasts who have ported it to different MCU platforms from various manufacturers. It took a few years to modify the Arduino IDE to allow it to include compilers for other microcontrollers as well, but with version 1.5 and the introduction of the Arduino Due that sports an ARM Cortex-M3 based MCU, this goal has been reached.

4.2 Pasta, Cheese and Tomato Sauce

The Arduino platform is built on three pillars:

1. The hardware consisting of a collection of microcontroller and expansion boards. Circuit diagrams and printed circuit board designs are available for free;

2. Software comprising of programming tools and an extensive library of high-level functions. Everything is free, open-source and multi-platform;

3. Distribution and communication in the shape of a website through which the Arduino hardware and software are made available to interested parties. The address is `www.arduino.cc` (`arduino.cc` also works). For the curious, `.cc` is the internet country code top-level domain reserved for the National Territory of Cocos (formerly Keeling) Islands, an Australian territory. This top-level domain was probably chosen for its low cost, but I digress.

The website is *the* official reference for everything that concerns Arduino. The website contains software and hardware updates and all the information necessary to use it all. If you are looking for a reseller of authentic Arduino hardware, it is here that you can find one. A forum to ask questions that are not answered in this book is also available. The site is in English, but there are sub-forums accommodating other languages too.

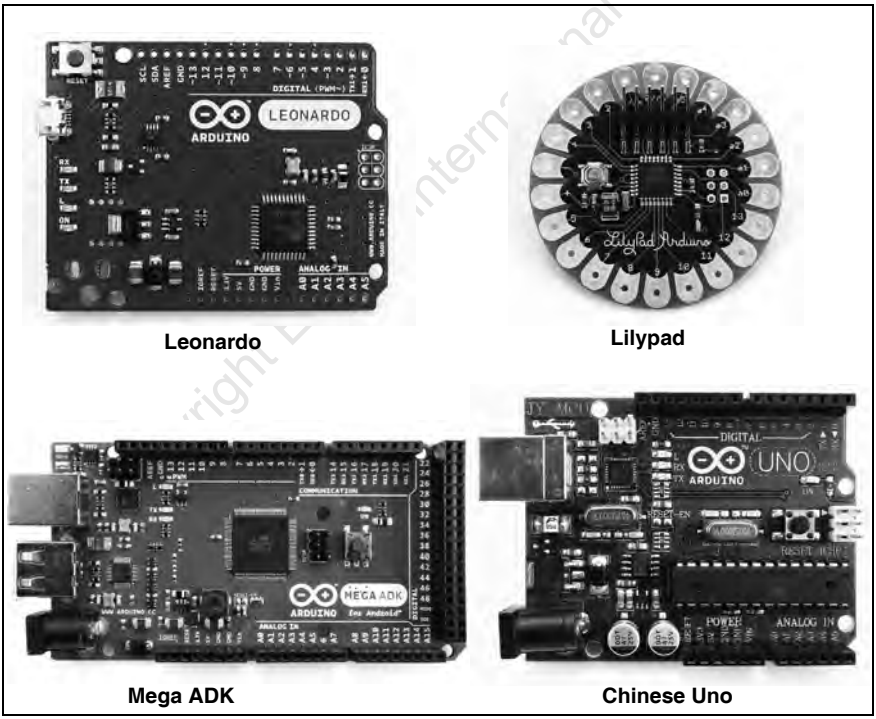


Figure 4-1 - Some less common Arduino boards: the Leonardo (without connectors) is equipped with an ATmega32U4 and targets USB devices; the daisy-shaped Lilypad aimed at wearable applications; the Mega ADK for communication with Android devices and a Chinese Uno clone recognizable from the missing “Made in Italy” statement.

On the hardware side of things, Arduino is a microcontroller board with a USB port through which it can be programmed. Several models exist, but the most common are the small Uno-shaped boards (Diecimila, Duemilanove, etc.) and the larger Mega 2560-style boards (Mega, Mega 1280, Mega ADK, etc.). There are others like the Mini, the Micro, the Nano or the round, daisy-shaped Lilypad intended for wearable applications, supposed to be used in clothing. The Arduino Due is different from all others because it is equipped with a 32-bit microcontroller instead of an 8-bit one and it is therefore much more powerful than the others[1]. The Yún, a combination of an Arduino Leonardo extended with a Wi-Fi system-on-a-chip running Linino (a MIPS GNU/Linux based on OpenWRT) has just been released.

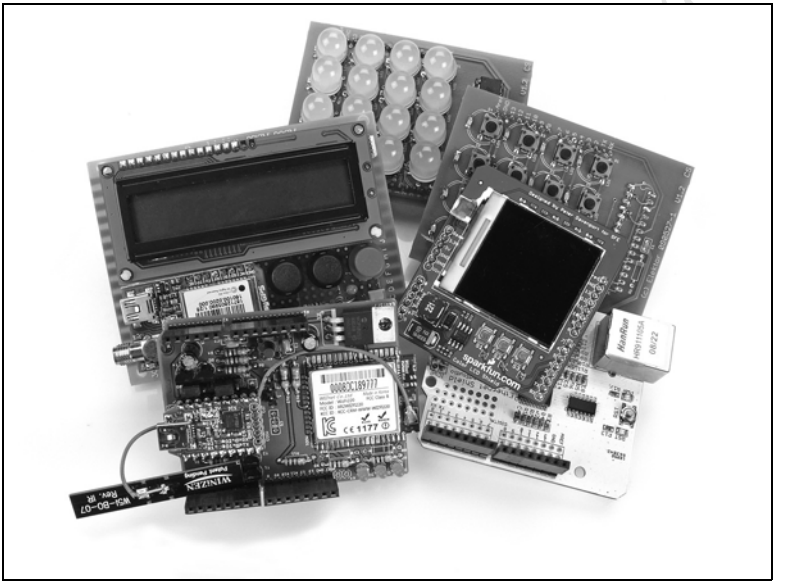


Figure 4-2 - A collection of Arduino compatible add-on boards, or Shields. Shown are a Wi-Fi/Bluetooth shield, an Ethernet shield (white), a graphical display shield with a mobile phone display on it, a shield with a 4 x 4 matrix keypad, a monome *10h* with 16 large white LEDs (see `www.monome.org`) and left from the center an experimental shield that we will build in Chapter 8.

1. The Arduino Due is not treated in this book. Not only did it arrive when I had almost finished this book, but it also makes things more complicated. A bit of Arduino's simplicity has been traded in for a more powerful microcontroller.

The Arduino board can be programmed in C, C++ or assembler using open-source tools available for Windows, Mac OS X and Linux. The hardware is also open and anyone can make their own Arduino board. Schematics and printed circuit board (PCB) designs can be downloaded for free and instructions on how to use it all are published on the website. If you buy a cheap Arduino Uno from an unofficial dealer, it is possible or maybe even likely, that you will receive a clone produced somewhere in China. A genuine Arduino board has "Made in Italy" printed on it, clones do not. Even though all the information to make an Arduino board is available for free on the internet, you are not authorized to make thousands of them. Marketing a compatible board is of course allowed as long as it is not identical.

The Arduino board is equipped with expansion connectors that can accommodate compatible add-on boards. Such an extension board is called a Shield and there are hundreds of them out there, designed by Arduino users all over the world for all kinds of applications. Several official Arduino-branded shields are also available, such as the Ethernet Shield, the Wireless SD Shield, the Motor Shield or the perforated Proto Shield for prototyping.

4.3 Base Ingredients

An Arduino board is fairly simple – until the appearance of the Due – since it is basically an 8-bit microcontroller with expansion slots to which the in- and outputs of the microcontroller are connected. Such a board is also known as a break-out board and they exist for many components, especially those that are too small to use on a prototyping board or a breadboard. ATmega*X* MCUs (where *X* is a number) belong to the megaAVR family from Atmel. The first Arduino boards were equipped with an ATmega8, the pre-Uno boards sported an ATmega168 whereas the Uno has an ATmega328. The Mega 2560 is based on an ATmega2560, and the Mega 1280 on an ATmega1280. All these MCUs are more or less compatible; the only things that differ are the pin count, the memory sizes and the on-chip peripherals. The Arduino Due is different from all other boards because it is equipped with an Atmel SAM3X8E 32-bit microcontroller based on an ARM Cortex-M3 core.

The microcontroller on an Arduino board contains a small boot loader, which allows loading a new application into the MCU through a serial port without special tools and without overwriting the boot loader. With this little piece of software, reprogramming the board is easy. Unfortunately, modern computers no longer have serial ports, only USB ports (besides of course other unusable ports

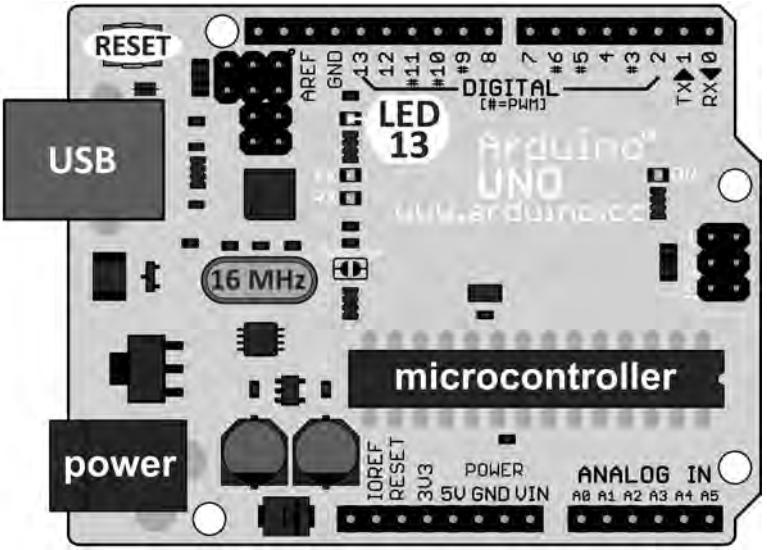


Figure 4-3 - Drawing of the Arduino Uno board showing the most important components. The board was designed for educational use, which is why it has this atypical outline that allows to explain to students how to position the board without requiring any knowledge about what exactly is on the board.

like HDMI, VGA, etc.), and an adapter that converts a USB port into a serial port is required. To avoid having to buy a converter on top of the board, the Arduino designers decided to include one on the board.

During development the USB port of the PC is also used to power the board, but when the application is finished and the board has to survive on its own, it must be powered in another way. An on-board voltage regulator offers some freedom in the choice of the external power source.

Besides a pushbutton to restart the MCU (reset) and some configuration jumpers, the board does not contain much else. An LED is often present – it is used in several Arduino programming examples – and it can be helpful to determine if the Arduino environment (board + programming software) is functioning as it should.

The MCU's clock is based on a 16 MHz crystal oscillator. This is an important detail, because the boot loader relies on this value to adjust the baud rate of the serial port. If you overclock the board (the MCU is specified up to 20 MHz) the boot loader will have communication problems.

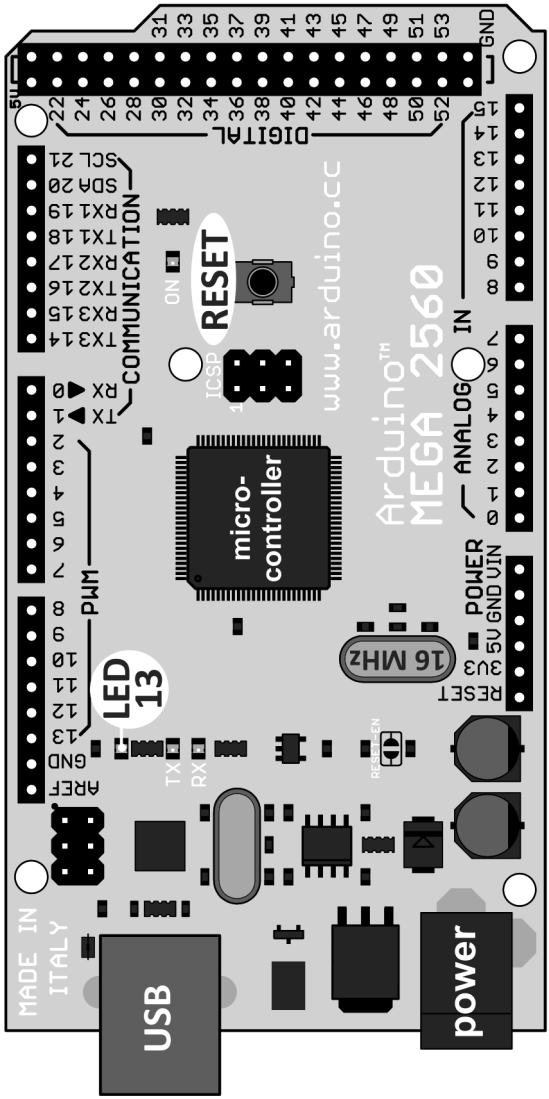


Figure 4-4 - The Arduino Mega (2560) is longer than the Uno because of the extra connectors needed to access the additional I/O that the ATmega2560 offers compared to the ATmega328 of the Uno. The Mega is compatible with the expansion boards for the Uno.

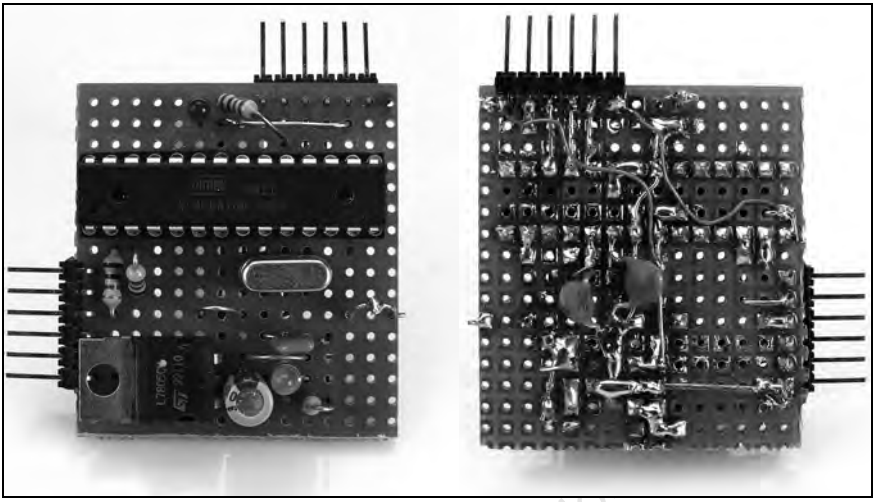


Figure 4-5 - A homemade Arduino compatible board, component side (left) and solder side (right).

That's all there is to know in case you would like to make your own Arduino compatible board. Basically, all you have to do is to program the boot loader into the ATmega device of your choice, clocked at 16 MHz. If you have a USB-to-serial-port converter or a computer with a real serial port, the USB portion of the board can be omitted. An RS-232 to TTL level adapter is essential in the last case (and maybe also in the first case, depending on the adapter).

Building an Arduino compatible board seems easy, but there are some complications. The most important one is the fact that an AVR programmer is required to load the boot loader into the microcontroller. Such a tool is inexpensive, but it means an extra piece of hardware. Another obstacle may very well be the boot loader that is not necessarily available as an executable for the AVR of your choice. Therefore use the ATmega328 (or ATmega168) or prepare yourself to adapt and recompile the boot loader. This is not an impossible task, but being a novice, you are probably not yet up-to-speed to do this successfully.

Finally, the MCU has some configuration fuses that must be programmed before you can use it. For this you also need an AVR programmer. The fuse values depend on the AVR type but, fortunately, the Arduino IDE contains all the necessary data. The IDE is capable of programming the boot loader into the microcontroller and configure its fuses for you, but your AVR programmer must be supported by the IDE.

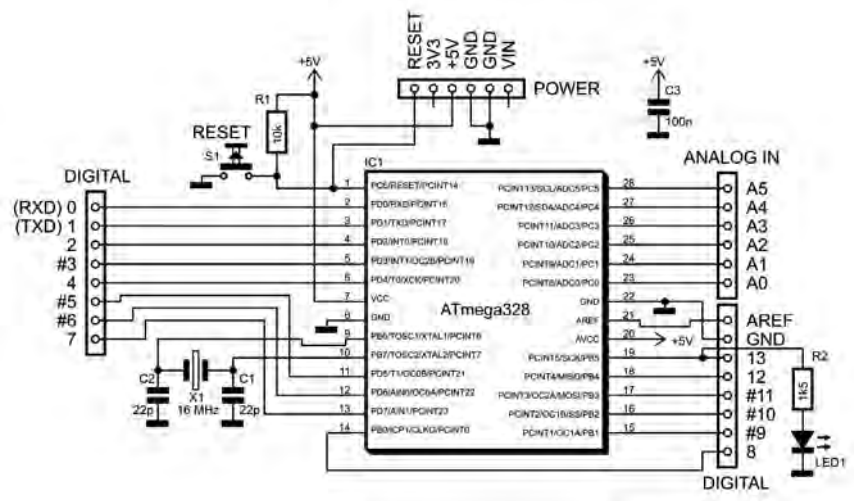


Figure 4-6 - This is all you need for an Arduino compatible board.

When you have completed your Arduino compatible board, you can give it a name. You are not allowed to call it "Arduino Something" since only boards approved by the Arduino team are entitled to the appellation. A solution is provided by the Freeduino community (`www.freeduino.org`) that allows the use of the Freeduino name for homemade Arduino boards. On their website there even is a tool to generate names that rhyme with Arduino.

4.4 The Kitchen

The details on how to install the Arduino IDE can be found in Chapter 1.

The Arduino IDE is rather simple. It is written in Java and includes:

- A rudimentary text editor to write the program;
- Some shortcuts to configure the IDE and to quickly find examples and help;
- Functions to compile the program and load it into the MCU;
- A simple terminal to communicate with the board over the serial port.

The six most frequently used functions have buttons below the main menu. From left to right we have *Verify*, *Upload*, *New*, *Open*, *Save* and *Serial Monitor*.

Figure 4-7 - The Arduino IDE and its three main areas.

A program is called a Sketch and it is written in C or C++ (or assembler if necessary).

With version 1.0.1 the IDE was extended with some major new features including menu translation into some thirty languages. By default, the IDE is in English, but it can be translated to, for instance, French, Danish or Lithuanian.

The introduction of version 1.5 of the IDE did not add many visible changes. The real novelty of this version is the addition of support for the Arduino Due board which, because it is based on a 32-bit microcontroller, needs a special programming tool chain.

4.4.1 File Menu

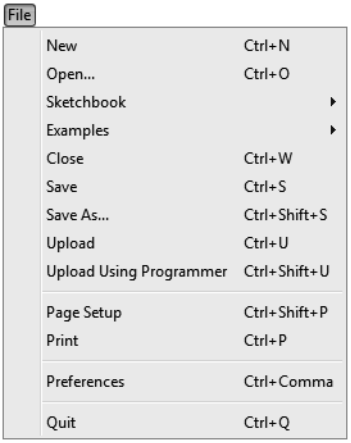


Figure 4-8 - The File menu.

New (Ctrl + N) – open a new window for a new sketch.

Open… (Ctrl + O) – open an existing sketch.

Sketchbook – open the Arduino IDE's working folder, the default location for storing sketches. Consult *Preferences* to find out where this folder is.

Examples – open the Arduino IDE's example folder.

Close (Ctrl + W) – close the active sketch.

Save (Ctrl + S) – save the active sketch.

Save as... (Ctrl + Shift + S) – saves the active sketch under a new name.

Upload (Ctrl + U) – compile and load the active sketch into the microcontroller using the serial port previously selected in the menu *Tools -> Serial Port*.

Upload Using Programmer (Ctrl + Shift + U) – compile and load the sketch into the microcontroller using the programmer previously selected in the menu *Tools -> Programmer.*

Page Setup (Ctrl + Shift + P) – prepare the active file for printing.

Print (Ctrl + P) – print the active file.

Preferences (Ctrl + comma) – display the IDE's preferences window. The window that opens shows only the most useful parameters, many others are available in the *preferences.txt* file whose location is indicated at the bottom of the window and that can be edited outside the IDE. Unfortunately, the documentation of these options leaves much to be desired.

- *Sketchbook location* specifies the default location where your sketches are saved. These are the sketches that are accessible when you click on *Sketchbook* in the *File* menu. *Editor language* lets you choose the language of the IDE. *Editor font size* sets the font used to write the sketch. By checking *Show verbose output during* the display of messages issued by the compiler and/or programmer is activated. The option *Verify code after upload* verifies that the program has been loaded correctly into the microcontroller, but it slows down the programming process. I recommend activating it only when you suspect a communication problem between the computer and the Arduino board.
- The option *Use external Editor* is also interesting. If you check this option, the IDE's editor is disabled. The IDE will compile the latest version of the sketch that you may have modified (and saved!) in an editor external to the IDE. It allows you to write the program in the editor of your choice, which is probably more powerful than the one that comes with the Arduino IDE. A disadvantage is that the sketch must be opened in two tools (the IDE and the external editor) and you have to go back and forth between the two to edit and compile the sketch.
- *Check for updates at start-up*: the Arduino IDE will contact the Arduino website to see if a new version of the IDE is available;
- *Update sketch files to new extension on save*: IDE versions before 1.0 used the file extension .PDE for sketches. With the introduction of version 1.0 of the IDE the sketch file extension was changed to .INO. Check this option to automatically change the extension of old sketches to .INO (with the result that older versions of the IDE will no longer recognize the renamed sketches).
- *Automatically associate .ino files with Arduino*: this option allows you to launch the IDE by clicking on a sketch with the extension .INO.

Quit (Ctrl + Q) – close all open windows of the IDE. Clicking on the small white cross in the red square in the upper right of a window only closes that one.

4.4.2 Edit Menu

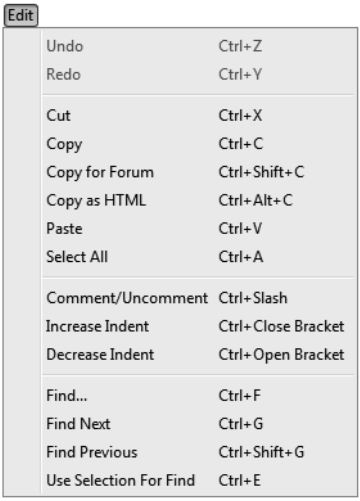


Figure 4-9 - The Edit menu.

Undo (Ctrl + Z) – cancel the last editor action.

Redo (Ctrl + Y) – redo the last editor action.

Cut (Ctrl + X) – cut the selected text.

Copy (Ctrl + C) – copy the selected text.

Copy for Forum (Ctrl + Shift + C) – copy the selected text including its colors and layout for posting it on the Arduino forum.

Copy as HTML (Ctrl + Alt + C) – copy the selected text including its colors and layout for inclusion in an HTML page.

Paste (Ctrl + V) – paste the selection that was previously cut or copied.

Select All (Ctrl + A) – select all the text in the editor.

Comment/Uncomment (Ctrl + */)* – add "//" to the beginning of each line of the selected text to comment it out or remove "//" from the beginning of each line of the selection to uncomment it. Note that the shortcut Ctrl + / is for a QWERTY keyboard and may not work on a keyboard with another layout. Tip for Windows: to change your keyboard in a QWERTY keyboard and back, press Shift + Alt. Now you can find the key that is at the position of '/'.

Increase Indent (Ctrl +]) – shift the selection to the right. Note that the shortcut Ctrl +] is for a QWERTY keyboard and may not work on a keyboard with another layout. Tip for Windows: to change your keyboard in a QWERTY keyboard and back, press Shift + Alt. Now you can find the key that is at the position of ']'.

Decrease Indent (Ctrl + [) – shift the selection to the left. Note that the shortcut Ctrl + [is for a QWERTY keyboard and may not work on a keyboard with another layout. Tip for Windows: to change your keyboard in a QWERTY keyboard and back, press Shift + Alt. Now you can find the key that is at the position of '['.

Find… (Ctrl + F) – find the next occurrence of a string. You can also replace the string with another.

Find Next (Ctrl + G) – find the next occurrence of the string selected by *Find* or *Use Selection For Find.*

Find Previous (Ctrl + Shift + G, since 1.0.1) – find the previous occurrence of the string selected by *Find* or *Use Selection For Find.*

Use Selection For Find (Ctrl + E, since 1.0.1) – select a string to use with the search functions *Find, Find Next* and *Find Previous* This function has no visible effect. Select some text, for example the name of a variable, press Ctrl + E and then start the search with Ctrl + F, Ctrl + G or Ctrl + Shift + G.

4.4.3 Sketch Menu

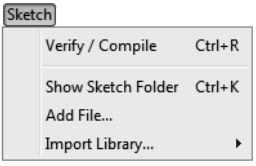


Figure 4-10 - The Sketch menu.

Verify / Compile (Ctrl + R) – compile the sketch without loading it into the microcontroller.

Import library... (version 1.5 or later) – add a line *#include <xxx.h>* to the top of the sketches main file, the one that bears the name of the sketch. "*xxx*" is the name of the library to import. This option provides a sub-option Add Library... for installing libraries prepared by other users.

Show Sketch Folder (Ctrl + K) – open the folder of the active sketch in a separate window.

Figure 4-11 - The Arduino IDE does not feature debugging. For example, setting breakpoints to stop the sketch is not possible.

Add File... – add a file to the sketch. Files that can be compiled (.ino, .pde, .h, etc.) are copied into the folder of the sketch and they are opened in new tabs in the IDE. For other files a *Data* subfolder is created (if it didn't exist yet) in the folder of the sketch and the file is copied to this subfolder. Such files will not be compiled.

Import Library... (version 1.0.x) – add a line *#include <xxx.h>* to the top of the sketches main file, the one that bears the name of the sketch. "*xxx*" is the name of the library to import.

4.4.4 Tools Menu

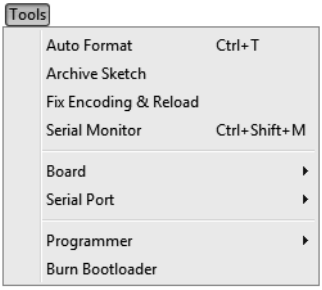


Figure 4-12 - The Tools menu.

This menu may take some time to open, especially with Windows XP. So be patient when you click on it.

Join the elektor Community

Take out a GOLD Membership now!

Your GOLD Membership contains:

- 8 Regular editions of Elektor magazine in print and digital
- 2 Jumbo editions of Elektor magazine in print and digital (January/February and July/August double issues)
- Elektor annual DVD-ROM
- A minimum of 10% DISCOUNT on all products in Elektor.STORE
- Direct access to Elektor.LABS
- Direct access to Elektor.MAGAZINE; our online archive for members
- Elektor.POST sent to your email account (incl. 25 extra projects per year)
- An Elektor Binder to store these 25 extra projects
- Exclusive GOLD Membership card

Also Available:

The all-paperless GREEN Membership, whick delivers all products and services, including Elektor magazine, online only.

Take out your Membership now at www.elektor.com/member