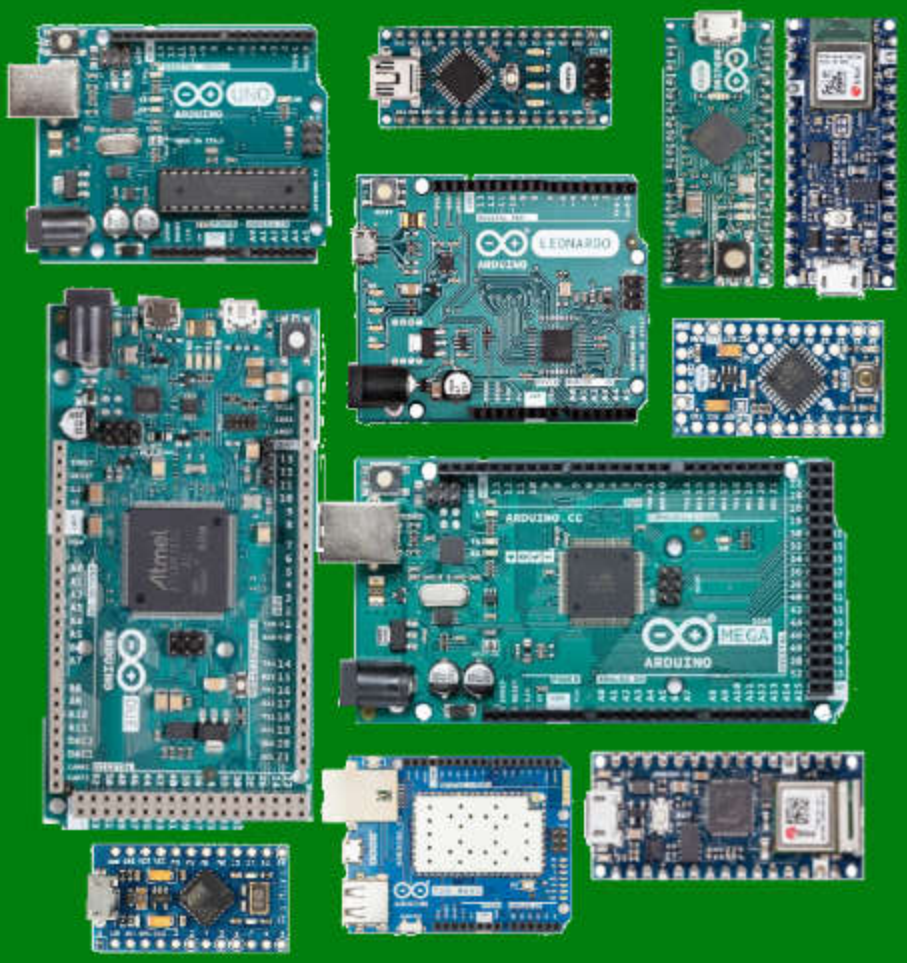
Arduino



For Beginners Step By Step

by Alexander Chukhryaev

1st edition

INDEX

Disclaimer

This eBook has been written for information purposes only. Every effort has been made to make this eBook as complete and accurate as possible.

The purpose of this eBook is to educate. It's a first edition of this book.

The author (Alexander Chukhryaev) does not warrant that the information contained in this eBook is fully complete and shall not be responsible for any errors or omissions.

The author (Alexander Chukhryaev) shall have neither liability nor responsibility to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by this eBook.

This eBook contains code examples which you can use on your own projects, excepted where otherwise noted.

You cannot redistribute this eBook.

This eBook is only available for free download at:
http://acoptex.com/wp/download

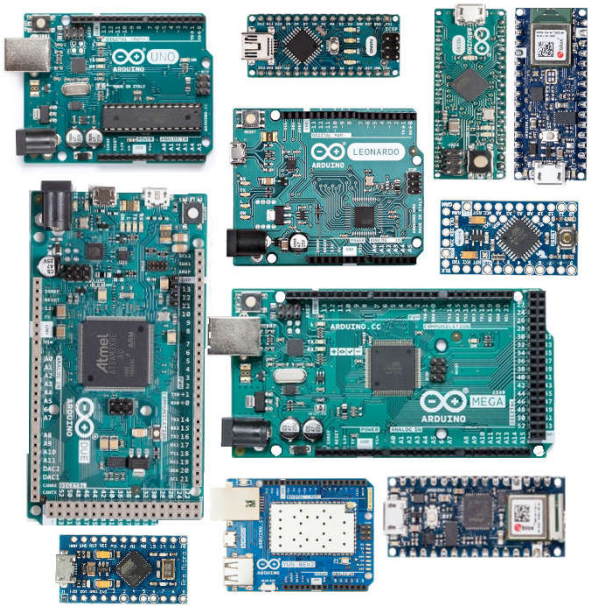
Please send an email to the author (Alexander Chukhryaev - info@acoptex.com), if you find this eBook anywhere else.

Introduction

By starting with some basic projects and learning the principles of how various electronic parts work, you can create your very own inventions and designs. Most modern electronic devices are actually a combination of both hardware and software, and that's where Arduino comes in. Arduino boards are tiny computers that let you build your own hardware projects, and then run small programs called "sketches" to make that hardware do different things. This combination of hardware and software is what makes electronic devices seem so magical, and it's what makes them both flexible and powerful. The hardware provides the inputs and the outputs, while the software controls how the device will behave.

This eBook will help you get started with your first adventures in electronics, but there's a whole world of opportunities to explore if you want to take things further.

Arduino can make your projects responsive, but only you can make them beautiful. I will provide some suggestions along the way as to how you might do that. Arduino was designed to help you get things done. To make that happen, I kept the background material on programming and electronics to a minimum. If you decide you want to know more about these aspects, there are lots of good DIY projects available on ACOPTEX.COM. I will provide a couple of references.



Arduino Boards. Image credit : Acoptex

Whether you are building a robot or working with Arduino, knowing how to use Arduino Boards and do C-based programming will come in handy.

Have fun with your projects. Thank you for reading,http://acoptex.com

Connect with Acoptex.Com

If you have any questions, please don't hesitate to contact us.

Here are some ways to stay in touch.

Visit our website
(http://acoptex.com)

Subscribe on YouTube
(https://www.youtube.com/channel/UCPQwh8eONWeNUeZ24CqDR1g/)

Like on facebook
(https://www.facebook.com/acoptexco/)

Follow on Twitter
(https://twitter.com/Acoptexcom/)

Check us on GitHub
(https://github.com/AcoptexCom/)

Follow us on Instagram
(https://www.instagram.com/acoptexcom/)

What is Arduino?

Arduino is an open source programmable circuit board that can be integrated into a wide variety of Do-It-Yourself (DIY) projects both simple and complex. This board contains a microcontroller which is able to be programmed to sense and control objects in the physical world. By responding to sensors and inputs, the Arduino is able to interact with a large array of outputs such as LEDs, motors and displays. Because of it's flexibility and low cost, Arduino has become a very popular choice for makers and Acoptex.com is looking to create interactive hardware projects.

Arduino was introduced back in 2005 in Italy by Massimo Banzi as a way for non-engineers to have access to a low cost, simple tool for creating hardware projects. Since the board is open-source, it is released under a Creative Commons license which allows anyone to produce their own board. If you search the web, you will find there are hundreds of Arduino compatible clones and variations available but the only official boards have Arduino in it's name.

Genuino is Arduino.cc's sister-brand created by Arduino co-founders, the Arduino.cc team and community. This brand is used for boards and products sold outside the US. So, Arduino and Genuino boards are exactly identical. From now on we will use the word "Arduino" to refer to the boards.

Similarly to a conventional computer (PC), an Arduino can perform a multitude of functions, but it's not much use on its own. It requires other inputs (sensors) or outputs (actuators) to make it useful.

Sensors listen to the physical world. They convert energy that you give off into electrical signals. Buttons and knobs are sensors that you touch with your fingers, but there are many other kinds of sensors.

Actuators take action in the physical world. They convert electrical energy back into physical energy, like light and heat and movement.

Arduino Boards

As Arduino is an open source platform anyone can make Arduino compatible boards. Many such boards are available in the market like the FreeDuino and NetDuino.

One way to differentiate between original boards and the compatible models is to look for the trade name on the board. The name Arduino is trademarked and reserved for boards made from the original patent company in Italy.

All compatible Arduino boards use a different but look alike trade name like Freeduino, Netduino, etc. If you want a cheaper alternative of the Arduino, you can always buy the clone boards. Adafruit and Sparkfun for example, sell variations of the Arduino boards which cost less but still have the same quality of the originals.

Another factor to consider when choosing a board is the type of project you are looking to do. For example, if you want to create a wearable electronic project, you might want to consider the LilyPad board from Sparkfun. The LilyPad is designed to be easily sewn into e-textiles and wearable projects. If your project has a small form factor, you might want to use the Arduino Pro Mini which has a very small footprint compared to other boards.

Many types of Arduino boards are available, each with its own design to suit various applications (Table 1).

Name	Processor	Operating Voltage	Analog In/ Out	Digital IO/ PWM	EEP ROM	SRAM (KB)	Flash (KB)	UA RT
Uno	ATmega328	5 V/7-12V	6/0	14/6	1	2	32	1
Due	AT91SAM3X8E	3.3 V/7-12V	12/2	54/12	-	96	512	4
Leonardo	ATmega32u4	5 V/7-12 V	12/0	20/7	1	2.5	32	1
Nano	ATmega168	5 V/7-9 V	8/0	14/6	0.512 1	1	16	1
Nano	ATmega328	5 V/7-9 V	8/0	14/6	1	2	32	1
Mega 2560	ATmega2560	5 V/7-12 V	16/0	54/15	4	8	256	4
Micro	ATmega32u4	5 V/7-12 V	12/0	20/7	1	2.5	32	1
Mini	ATmega328	5 V/7-9 V	8/0	14/6	1	2	32	-
Yun	ATmega32u4	5 V	12/0	20/7	1	2.5	32	1

Name	Processor	Operating Voltage	Analog In/ Out	Digital IO/ PWM	EEP ROM	SRAM (KB)	Flash (KB)	UA RT
Mega ADK	ATmega2560	5 V/7-12 V	16/0	54/15	4	8	256	4
Ethernet	ATmega328	5 V/7-12 V	6/0	14/4	1	2	32	-
Esplora	ATmega32u4	5 V/7-12 V	-	-	1	2.5	32	-
ArduinoBT	ATmega328	5 V/2.5-12 V	6/0	14/6	1	2	32	1
Fio	ATmega328P	3.3 V/3.7-7 V	8/0	14/6	1	2	32	1
Pro	ATmega168	3.3 V/3.35-12 V	6/0	14/6	0.512	1	16	1
Pro	ATmega328	5 V/5-12 V	6/0	14/6	1	2	32	1
Pro Mini	ATmega168	3.3 V/3.35-12 V 5 V/5-12 V	6/0	14/6	0.512	1	16	1
LilyPad	ATmega168V	2.7-5.5 V	6/0	14/6	0.512	1	16	-
LilyPad	ATmega328V	2.7-5.5 V	6/0	14/6	0.512	1	16	-
LilyPad USB	ATmega32u4	3.3 V/3.8-5V	4/0	9/4	1	2.5	32	-
LilyPad Simple	ATmega328	2.7-5.5 V	4/0	9/4	1	2	32	-
LilyPad SimpleSnap	ATmega328	2.7-5.5 V	4/0	9/4	1	2	32	-

Table 1. Boards

Deciding what board to use can be a daunting prospect because the number of boards is increasing, each with new and exciting prospects. I would suggest to use the **Arduino Nano** for beginners as the **Arduino Uno** and the **Arduino Mega** are more suitable for advanced users. I will be using the **Arduino Nano** in this EBook.

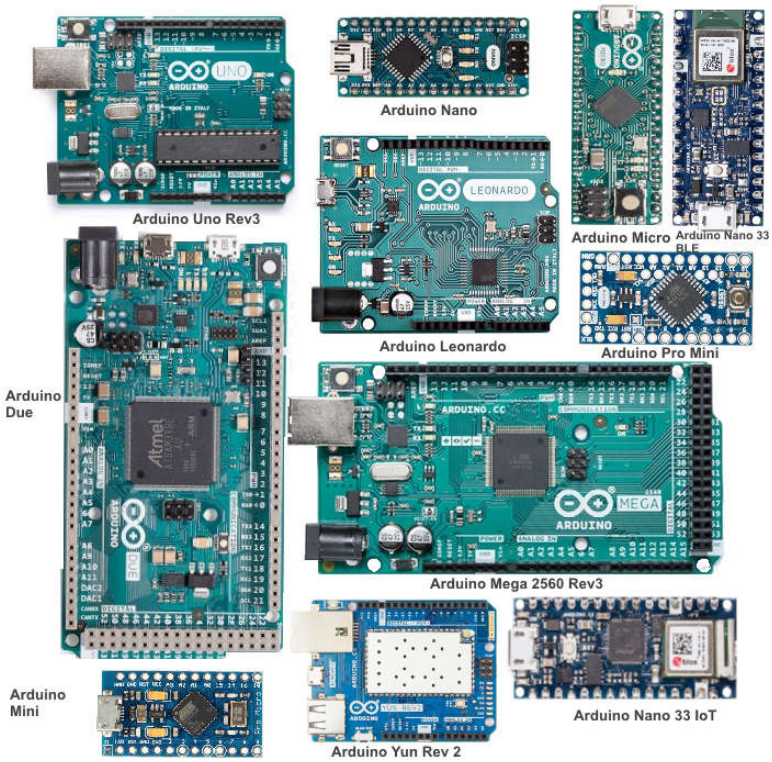


Image credit: Acoptex

Arduino Uno

The Arduino Uno is one of the best boards to get started with electronics and coding. The Arduino Uno is the most used and documented board of the whole Arduino family.

The most recent main board to date is the Arduino Uno R3 (released in 2011). "Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. R3 relates to the revision of the features on the board, which includes updates, refinements, and fixes. In this case, it is the third revision.

Arduino Uno is a microcontroller board based on the ATmega328P chip. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.

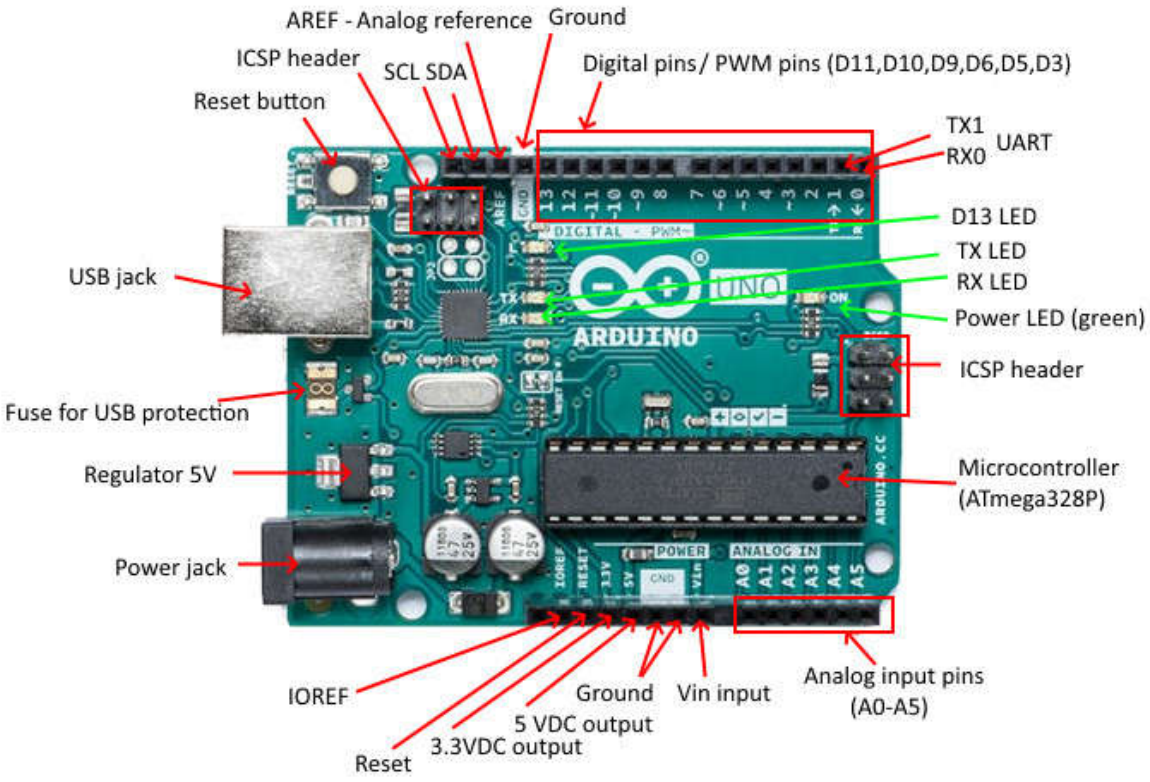


Image credit: Acoptex

Programming

The Arduino Uno can be programmed with the (Arduino IDE). The ATmega328 on the Arduino Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar.

Warnings

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Differences with other boards

The Arduino Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Power

The Arduino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector. The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are:

Vin. The input voltage to the Arduino Uno board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
5V. This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. I don't advise it.
3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
GND. Ground pins.

IOREF. This pin on the Arduino Uno board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

Memory

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Arduino Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 - A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. There are a couple of other pins on the board:

AREF. Reference voltage for the analog inputs. Used with analogReference().
Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

Arduino Uno has a number of facilities for communicating with a computer, another Arduino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino IDE includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino IDE includes a Wire library to simplify use of the I2C bus. For SPI communication, use the SPI library.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino IDE uses this capability to allow you to upload code by simply pressing the upload button in the interface toolbar. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Arduino Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line.

REV 3 of the Arduino Uno board has the following new features:
1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V

and with the Arduino Due that operates with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
Stronger RESET circuit.
Atmega 16U2 replace the Atmega 8U2.

Arduino Uno Pinout

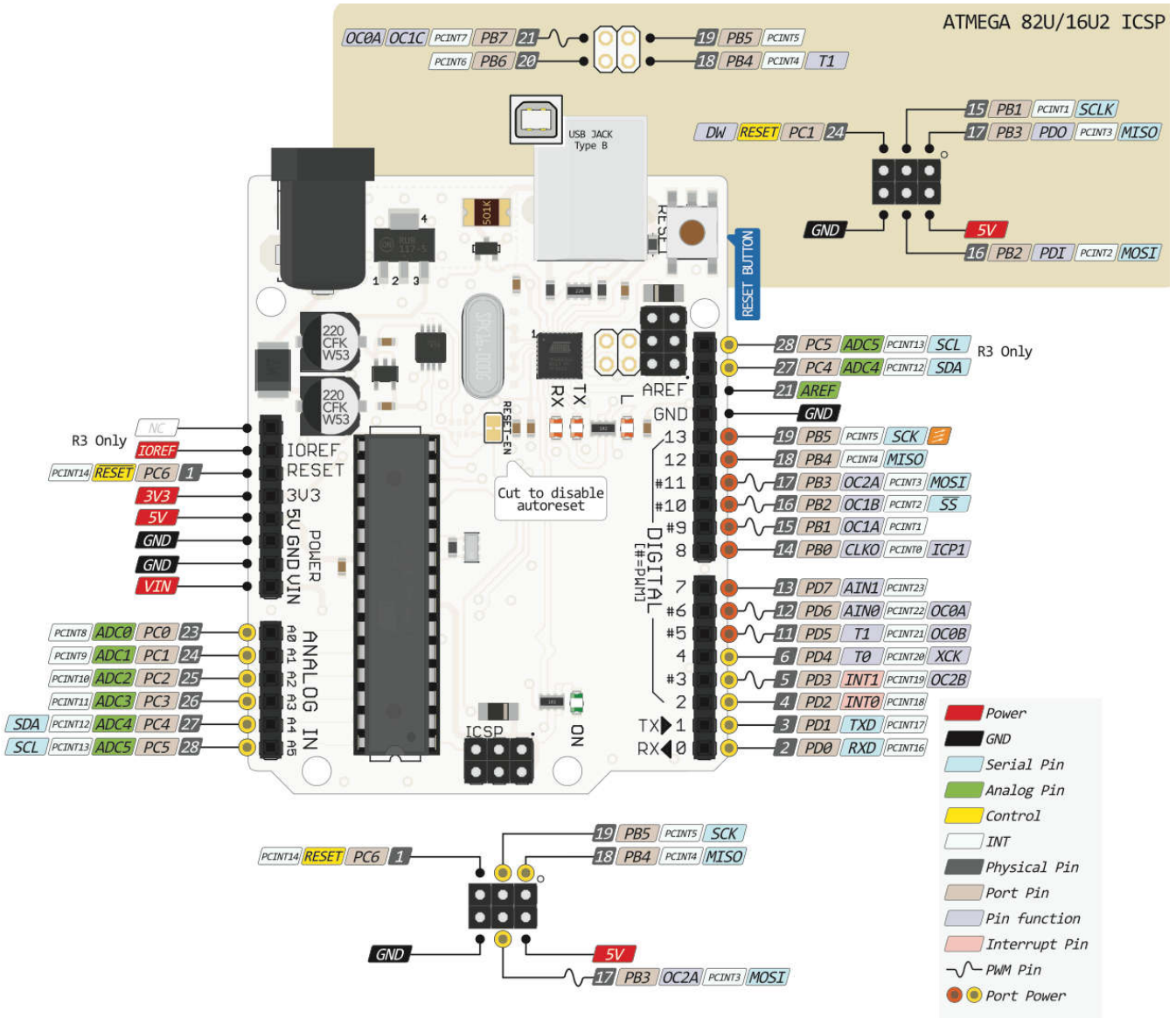


Image credit: bq.com

Arduino Nano

The Arduino Nano is a compact board similar to the UNO. It is a breadboard-friendly board based on the ATmega328 (Arduino Nano 3.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one.

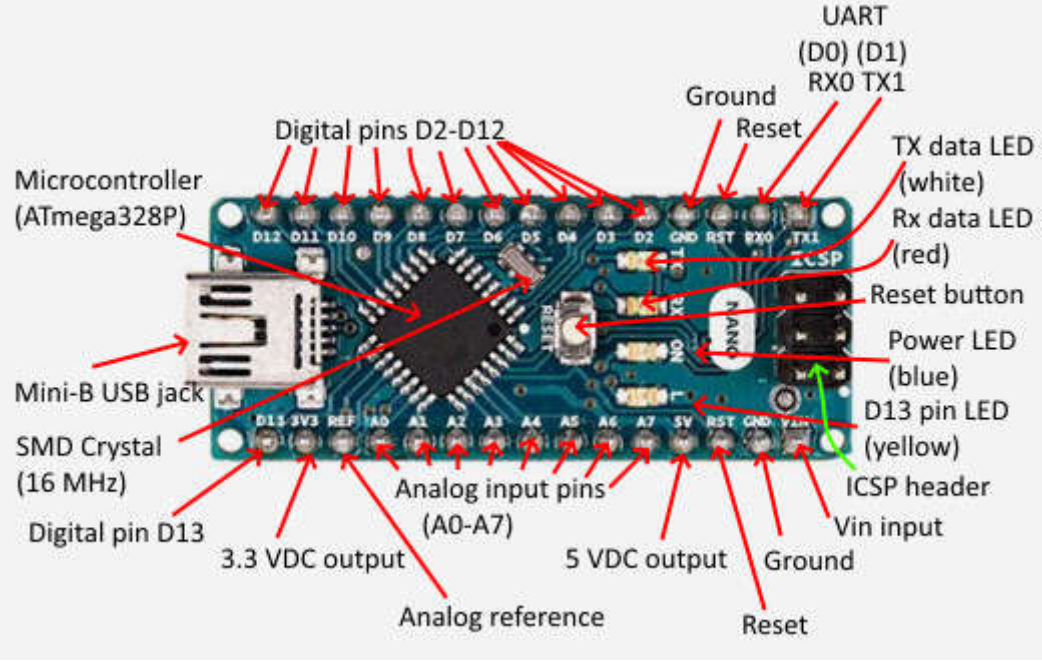


Image credit: Acoptex

Programming

The Arduino Nano can be programmed with the Arduino IDE. The ATmega168/328 on the Arduino Nano comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar.

Power

The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

Memory

The ATmega168 has 16 KB. The ATmega168 has 1 KB of SRAM and 0.5121 KB of EEPROM. The ATmega328 has 32 KB, (also with 2 KB used for the bootloader). The ATmega328 has 2 KB of SRAM and 1 KB of EEPROM.

Input and Output

Each of the 14 digital pins on the Nano can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.

External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Nano has 8 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the analogReference() function. Analog pins 6 and 7 cannot be used as digital pins. Additionally, some pins have specialized functionality:

I2C: A4 (SDA) and A5 (SCL). Support I2C (TWI) communication using the Wire library (documentation on the Wiring website).

There are a couple of other pins on the board:

REF. Reference voltage for the analog inputs. Used with analogReference().

Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Nano has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provide UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An FTDI FT232RL on the board channels this serial communication over USB and the FTDI drivers (included with the Arduino IDE) provide a virtual com port to software on the computer. The Arduino IDE includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Arduino Nano's digital pins. The ATmega328 also support I2C (TWI) and SPI communication. The Arduino IDE includes a Wire library to simplify use of the I2C bus.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Nano is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the FT232RL is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino IDE uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Arduino Nano is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Arduino Nano. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

Arduino Nano Pinout

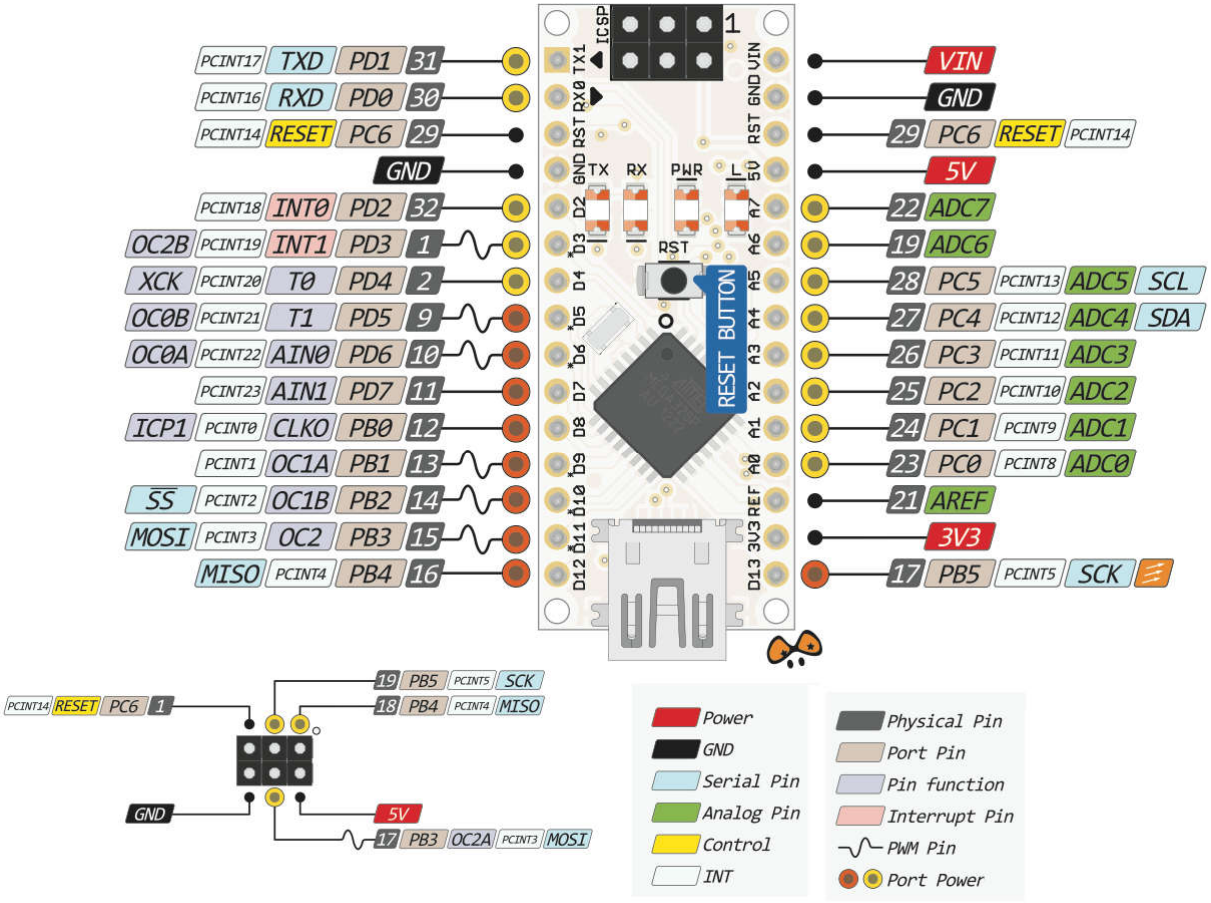


Image credit: bq.com

Arduino IDE

The open-source Arduino IDE makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board, ESP8266 WiFi module, ESP32 WiFi module. The Arduino IDE is a multiplatform software, which means that it runs on Windows, Mac OS X or Linux (it was created in JAVA).

Preparations:

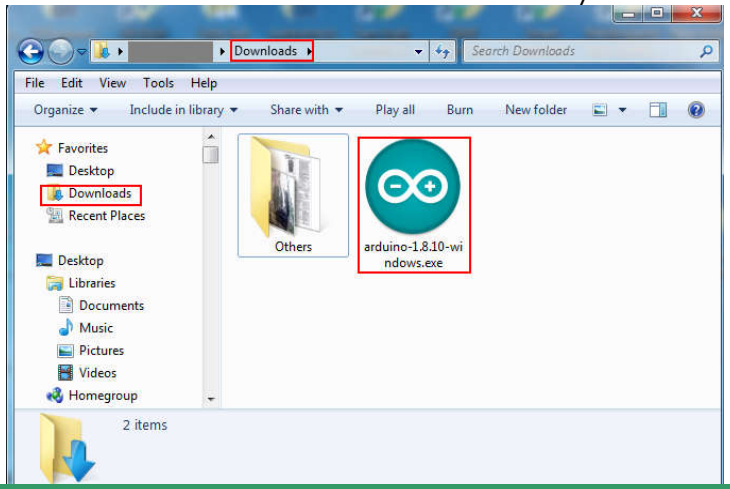
You need to have the JAVA installed in your computer (PC). If you do not have it, go to this website: https://www.java.com/en/download/, download and install the latest version.

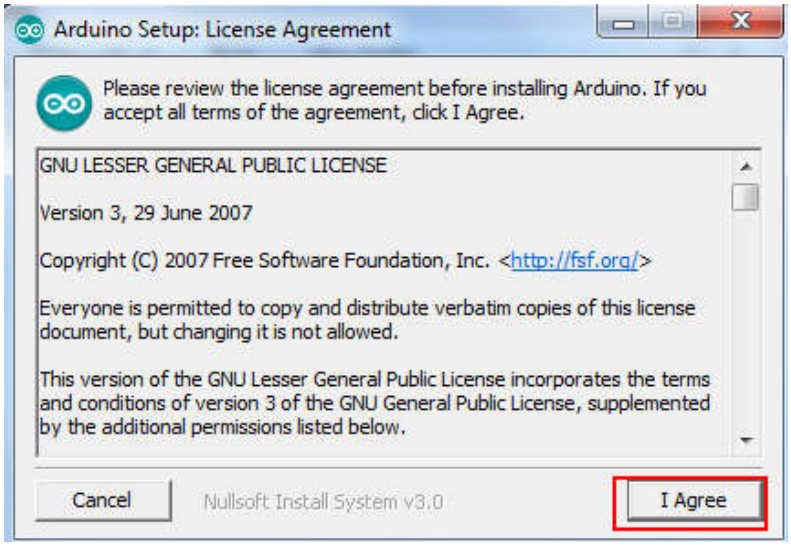
Downloading Arduino IDE:

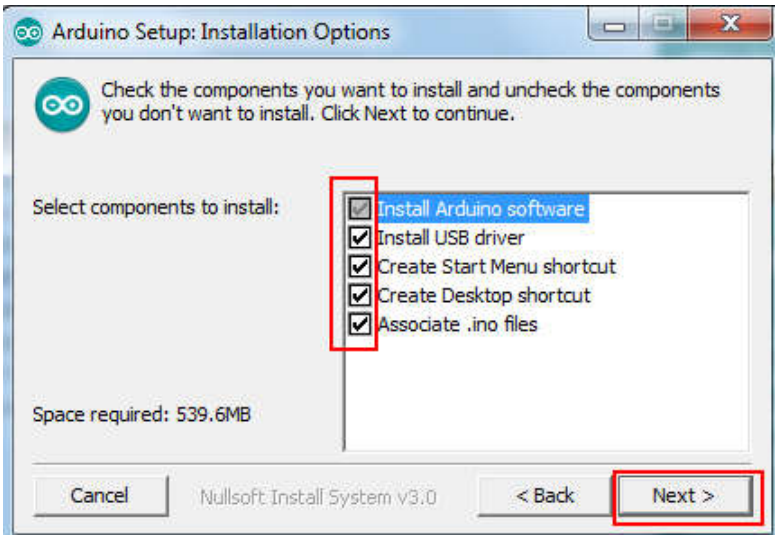
Go to arduino.cc webpage, select your operating system (OS) and download the Arduino IDE

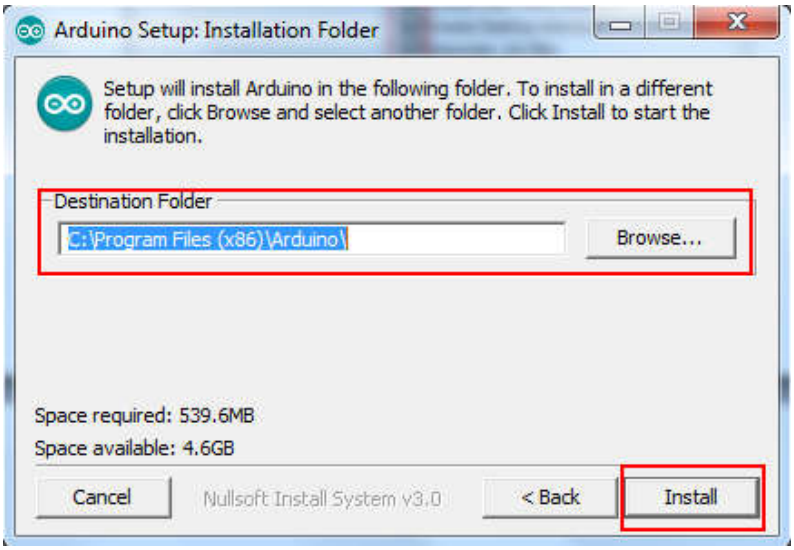
Installing Arduino IDE on Windows OS

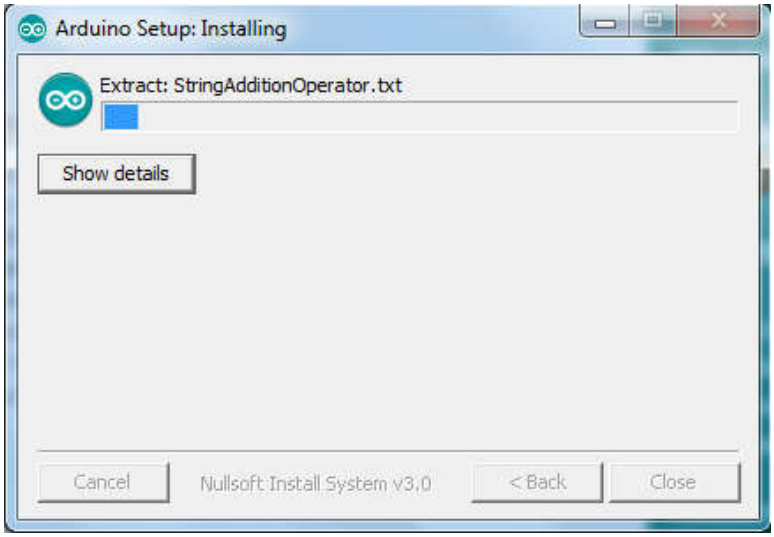
Go to your PC **Download** folder and double-click on file named "**arduino-(...).exe**". Follow the installation wizard that shows on your PC screen.

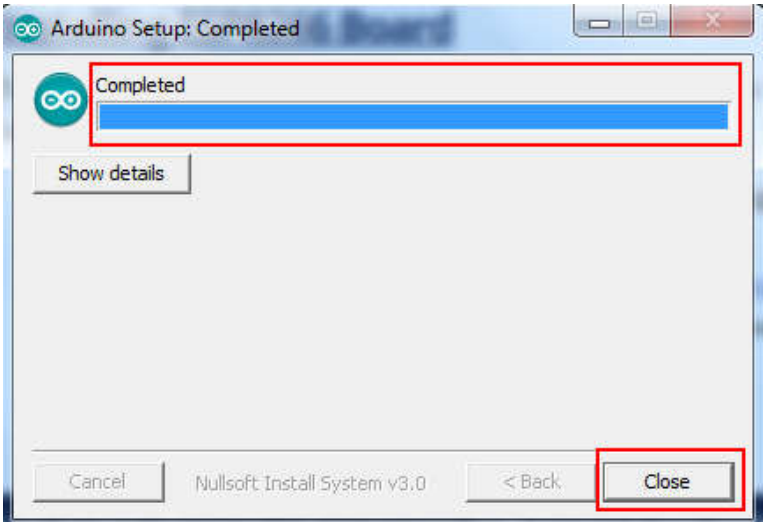








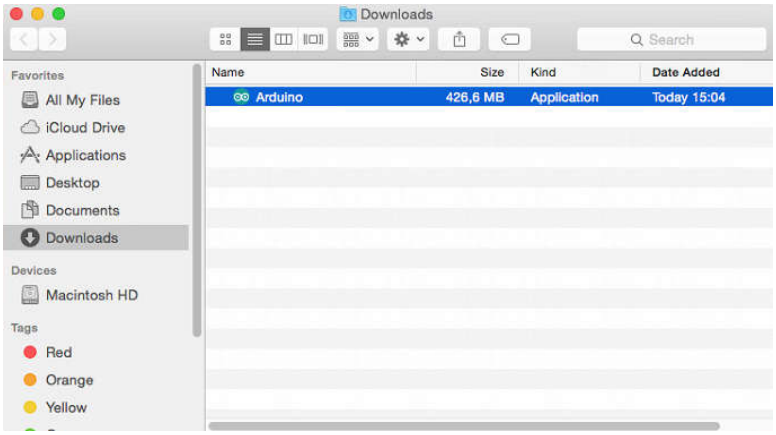




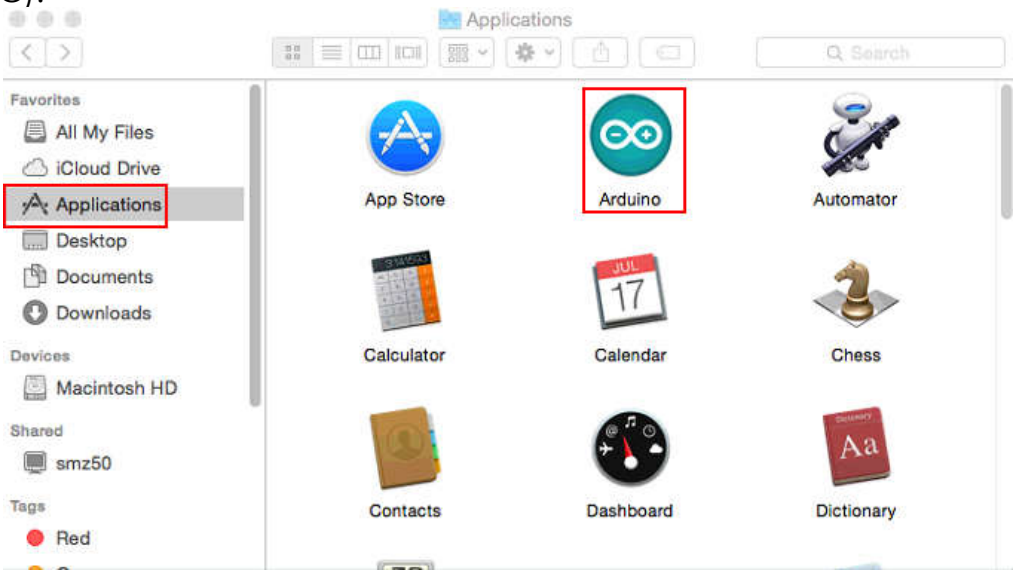
Congrats! You have installed Arduino IDE to your PC now.

Installing Arduino IDE on OS X (MAC)

1. The file downloaded from Arduino website is in Zip format, located in **Downloads**. If you use Safari it will be automatically expanded. If you use a different browser you may need to extract it manually - double-click on it.



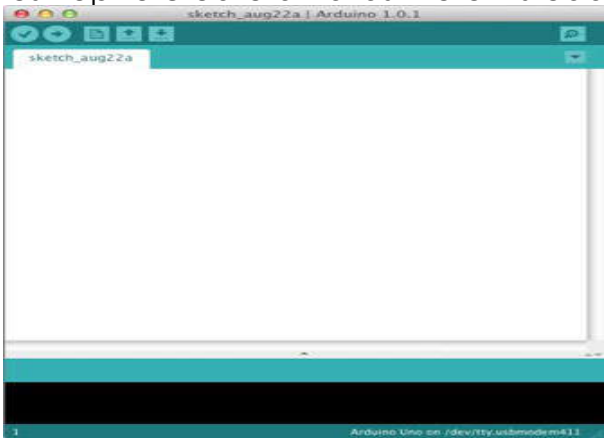
2. Copy the Arduino application into the Applications folder (or elsewhere on your PC).



3. Power up your Arduino by connecting your Arduino board to your PC with a USB cable. You should see the an LED labed 'ON' or 'L' light up. As soon as the board is connected, a dialog box appears, showing the message **A new network interface has been detected**.

4. Click on **Network Preferences..** button, and in the window that appears, click on **Apply** button. Close the Network Preferences window. Please note that your Arduino is displayed in the list on the left side of this window as Not Configured, but don't worry, the software is installed and your Arduino board will work.

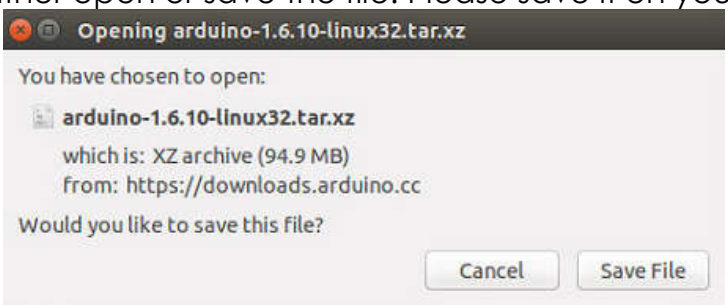
5. To launch the Arduino application, go to your Applications folder, locate the Arduino application, drag it to the Dock, and then click on **Arduino** icon to open the Arduino application. If you prefer, you can also drag the application to the desktop to create an alias there instead.



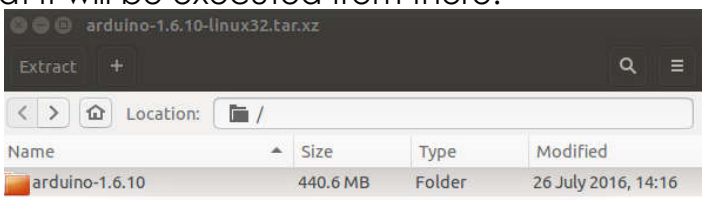
Installing Arduino IDE on Linux

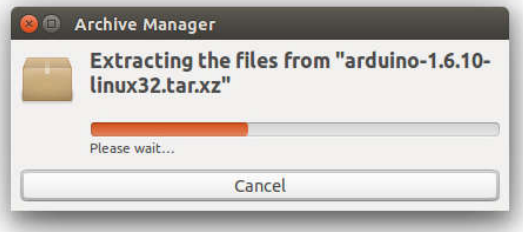
The Linux build of the Arduino IDE is now a package that doesn't require any specific procedure for the various distributions available of Linux. The only relevant information is the 32 or 64 bit version of the OS.

1. You can choose between the 32, 64 and ARM versions on Arduino webpage. It is very important that you choose the right version for your Linux distro. Clicking on the chosen version brings you to the donation page and then you can either open or save the file. Please save it on your PC.

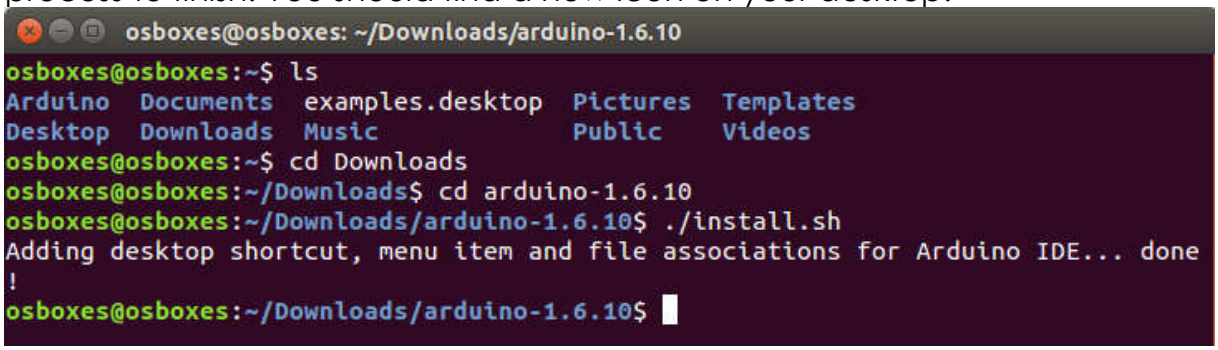


2. The file is compressed and you have to extract it in a suitable folder, remembering that it will be executed from there.





3. Open the **arduino-1.6.x** folder just created by the extraction process and spot the **install.sh** file. Right click on it and choose **Run** in **Terminal** from the contextual menu. The installation process will quickly end and you should find a new icon on your desktop. If you don't find the option to run the script from the contextual menu, you have to open a **Terminal** window and move into the **arduino-1.6.x** folder. Type the command **./install.sh** and wait for the process to finish. You should find a new icon on your desktop.



Do Wiring

Arduino Nano

You just need to connect Arduino Nano with Mini-B USB cable to your PC USB port. The power LED (labelled POW) will be ON.

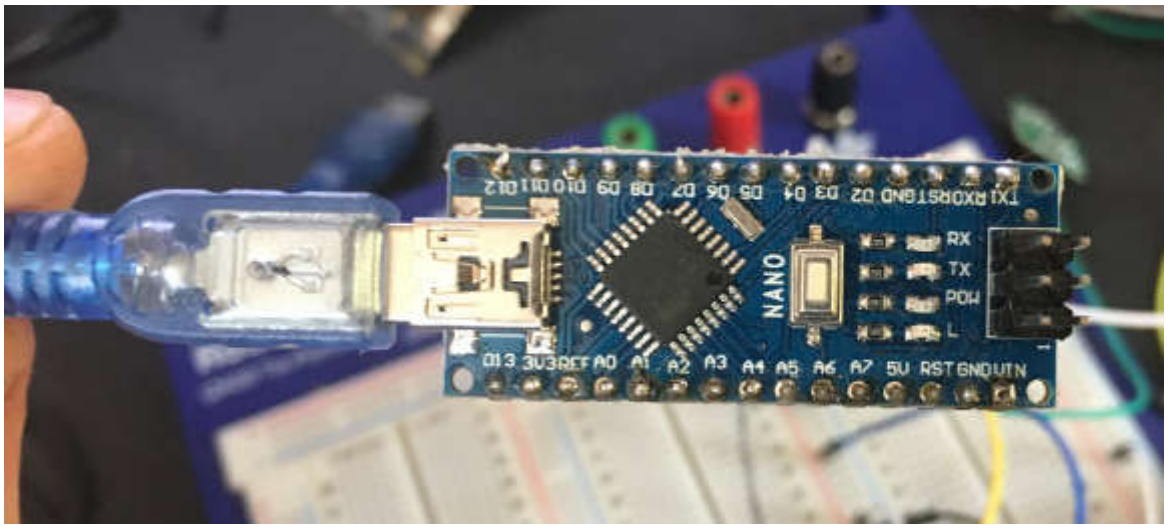


Image credit: Acoptex

Arduino Uno

You just need to connect Arduino Uno with USB 2.0 Cable - A-Male to B-Male (USB printer cable) to your PC USB port. The green power LED (labelled ON or PWR) will be ON.

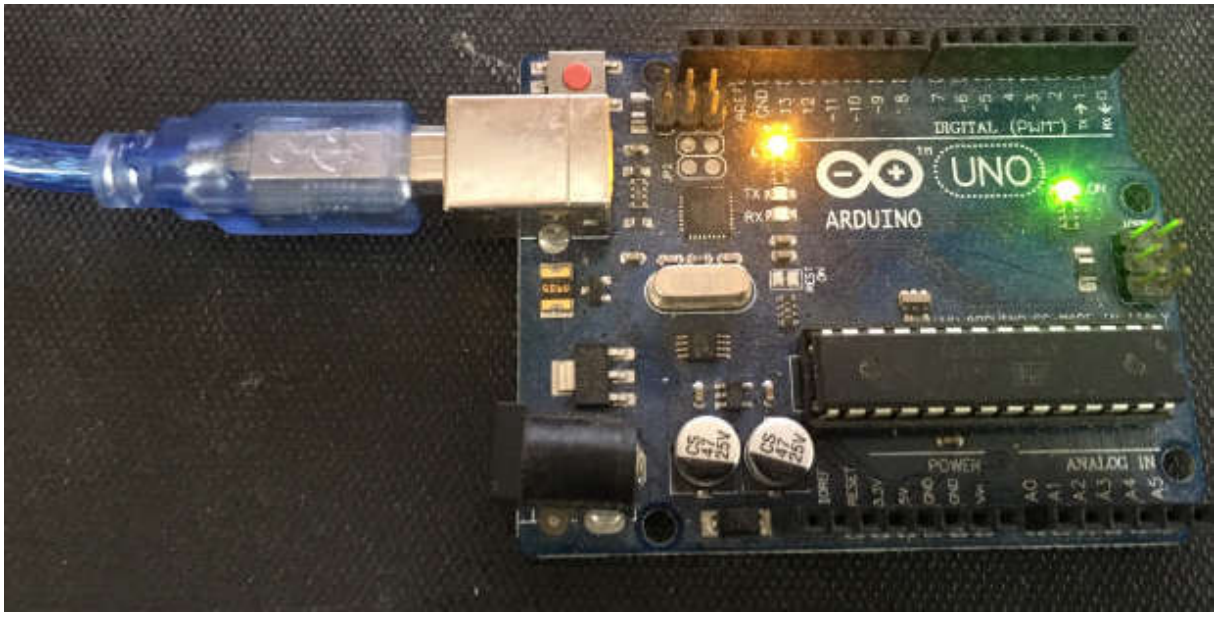


Image credit: Acoptex

Uploading The Sketch

Install the board drivers

If you used the Installer, Windows - from XP up to 10 - will install drivers automatically as soon as you connect your board. If you downloaded and expanded the Zip package or, for some reason, the board wasn't properly recognized, please follow the procedure below.

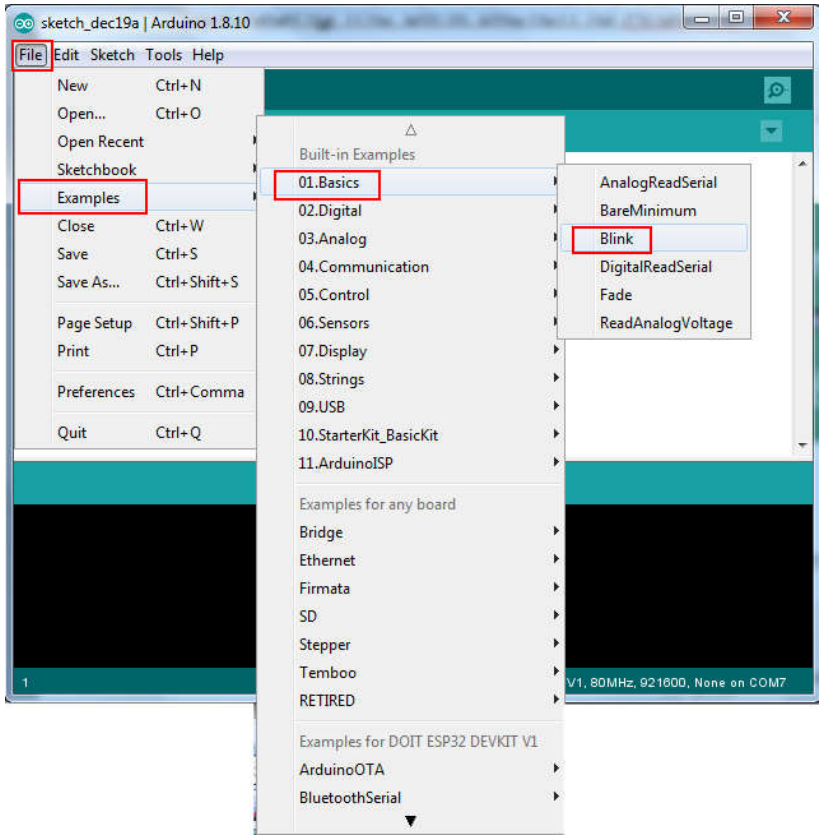
- Click on the Start Menu, and open up the Control Panel.
- While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.
- Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)". If there is no COM & LPT section, look under "Other Devices" for "Unknown Device".
- Right click on the "Arduino UNO (COmxx)" port and choose the "Update Driver Software" option.
- Next, choose the "Browse my computer for Driver software" option.
- Finally, navigate to and select the driver file named "arduino.inf", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno driver file named "Arduino UNO.inf"
- Windows will finish up the driver installation from there.

Before use non-official Arduino Board, you need to download the manufacture's driver (CH340) for this chip and install it in your PC - http://www.wch.cn/download/CH341SER_EXE.html . See the description of driver installation package below: CH340/CH341 USB to serial WINDOWS driver installation package that supports 32/64 bit Windows 10 / 8.1 / 8/7 / VISTA / XP, SERVER 2016/2012/2008/2003, 2000 / ME / 98, through Microsoft digital signature authentication, support USB to 3-wire and 9-wire serial port, with the product release To the end user. Applicable scope: CH340G, CH340C, CH340B, CH340E, CH340T, CH340R, CH341A, CH341T, CH341H chips.

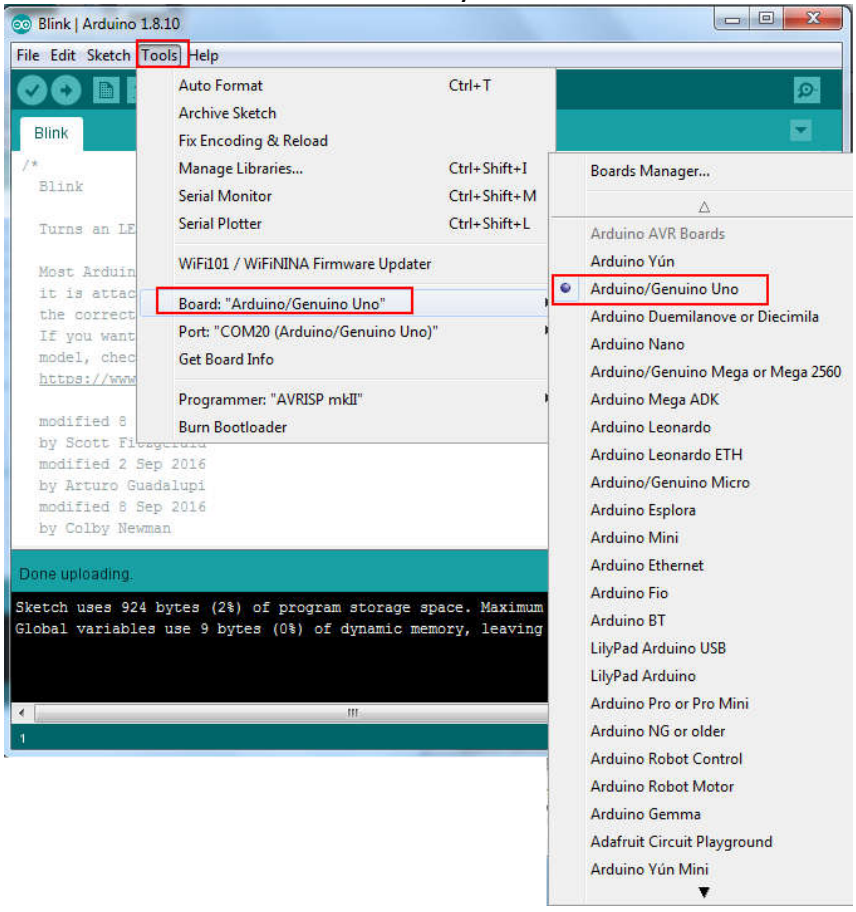
If you have CP2102 chip then you need to download the manufacture's driver for this chip and install it in your PC. You can download them here: https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

Uploading The Blink Sketch to Arduino Uno

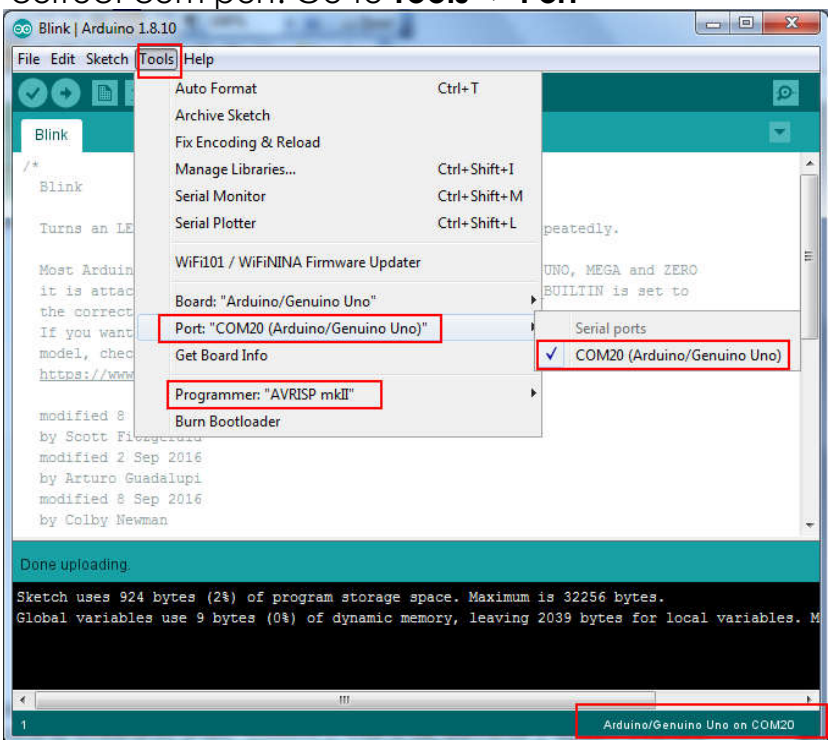
1. Plug your Arduno Uno to your PC USB port.
2. Re-open your Arduino IDE.
3. Open the LED blink example sketch. Go to **File -> Examples ->01.Basics -> Blink**



4. Go to **Tools->Board** and select **Arduino/Genuino Uno**.

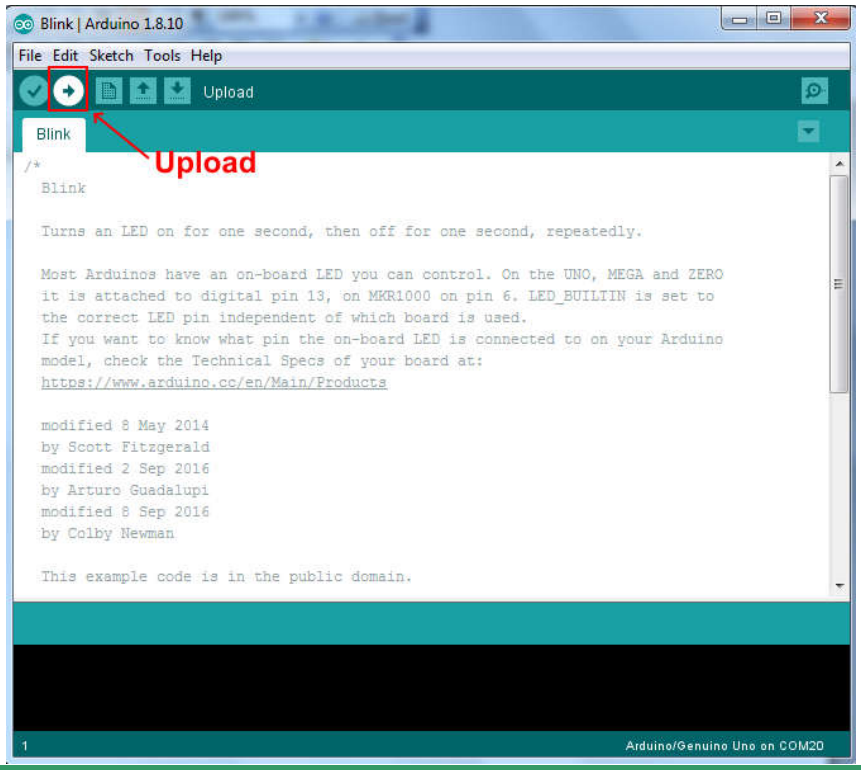


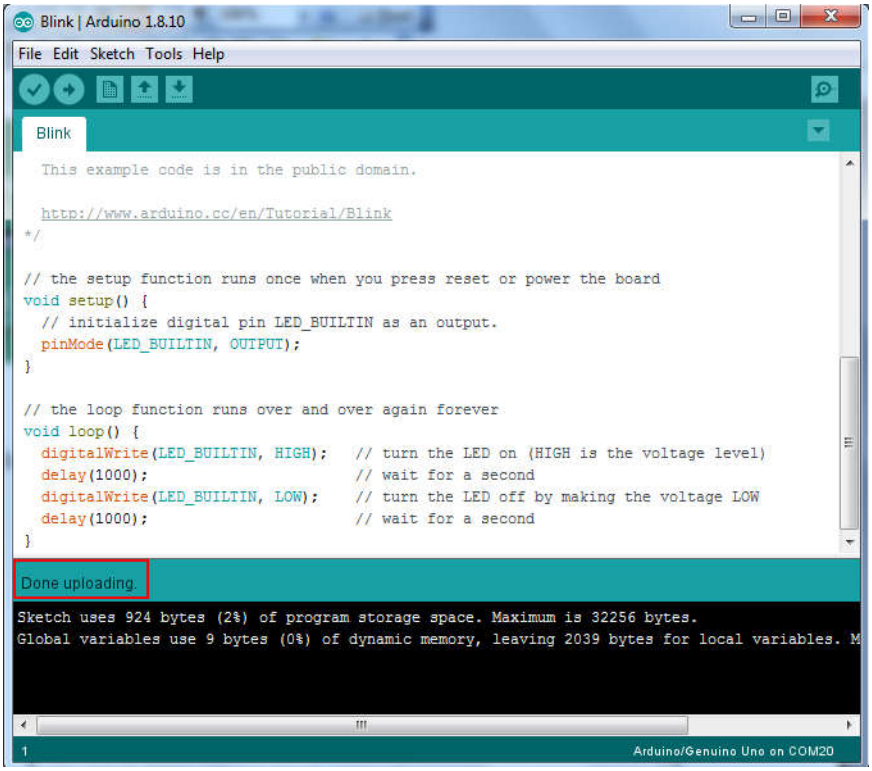
5. Select the correct com port. Go to **Tools -> Port**



Please note that your COM port is very likely to be different from the preceding screenshot (Port: "COM20"). That is ok, because it doesn't interfere with anything. On the other hand, all the other configurations should look exactly like mine.

6. After checking the configurations, click on **Upload** button in the Arduino IDE and wait a few seconds until you see the message **Done uploading** in the bottom left corner.

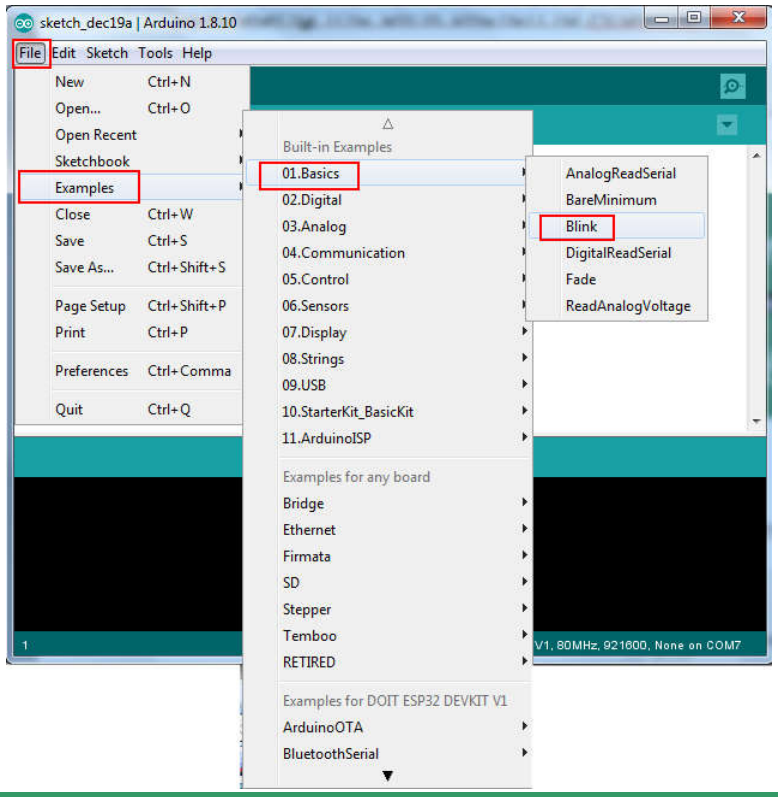




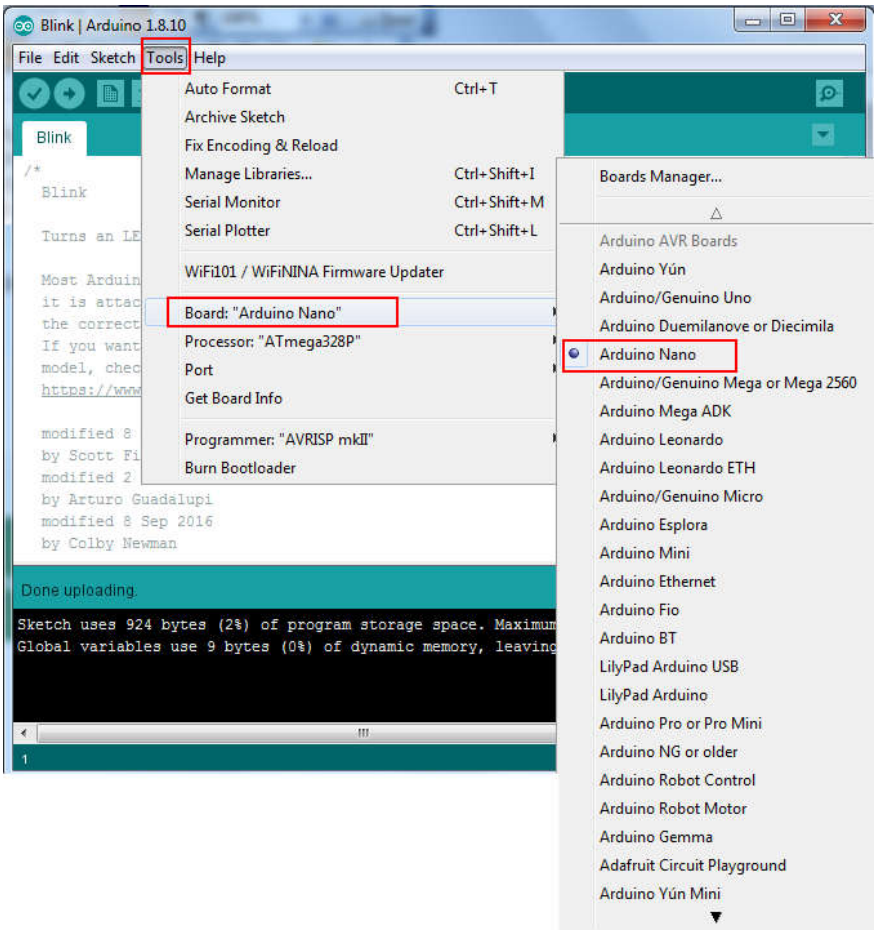
7. You will see that on-board (L) LED is blinking with 1 second intervals.

Uploading The Blink Sketch to Arduino Nano

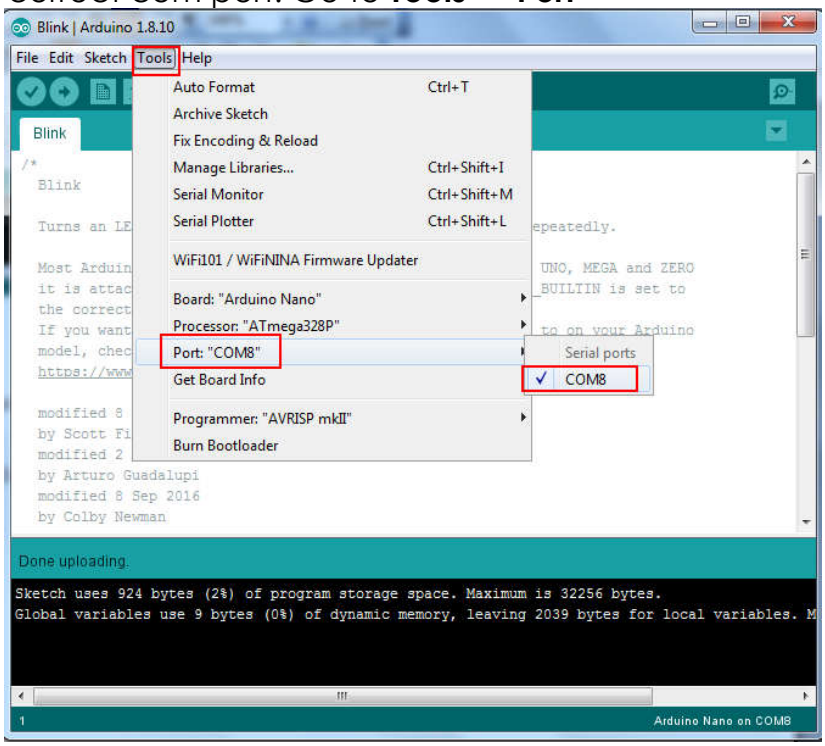
1. Plug your Arduno Nano to your PC USB port.
2. Re-open your Arduino IDE.
3. Open the LED blink example sketch. Go to **File -> Examples ->01.Basics -> Blink**



4. Go to **Tools->Board** and select **Arduino Nano**.

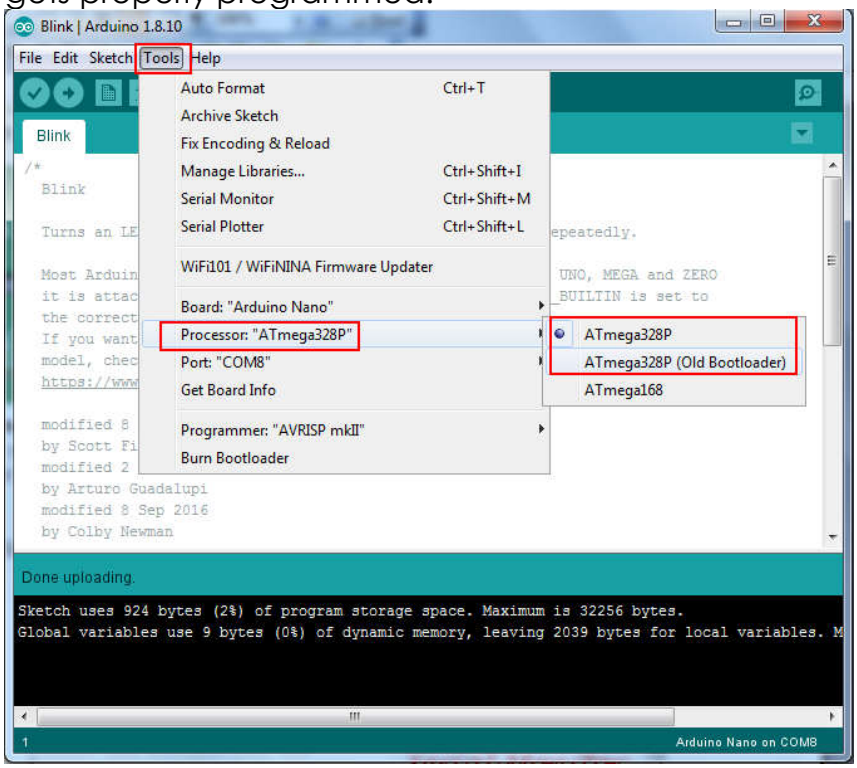


5. Select the correct com port. Go to **Tools -> Port**

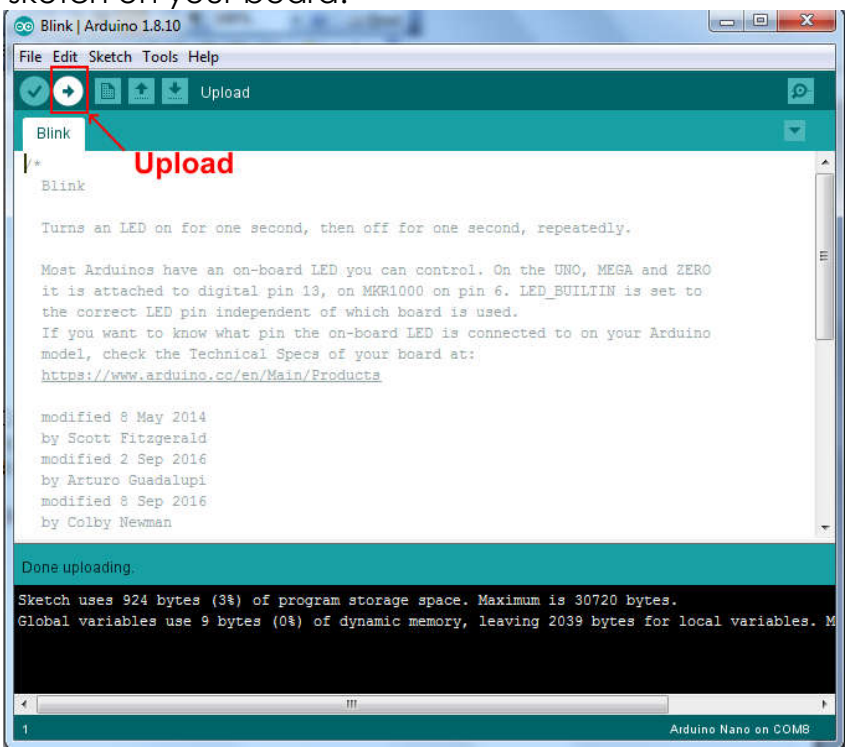


6. Go to **Tools-> Processor** and select **ATmega328P (from January 2018 sold NEW Arduino NANO boards)**. To program old boards you need to choose

ATmega328P (Old Bootloader). If you get an error while uploading or you are not sure which bootloader you have, try each type of processor 328P until your board gets properly programmed.



7. Compile and upload the sketch to your Adruino Nano. To upload the sketch to the Arduino Nano, click the **Upload** button in the upper left to load and run the sketch on your board.



Wait a few seconds. If the upload is successful, the message "**Done uploading**." will appear in the status bar. You will see the on-board (L) LED blinking (in yellow or red) every second.

The Code

The **comments** are very handy as it is quite easy to understand the code if the comments have written in the sketch. There are two ways to have comments in code. You can set them between **/* */** or after **//**

Every Arduino program has two main functions. Functions are parts of a computer program that run specific commands. Functions have unique names, and are "called" when needed. The necessary functions in an Arduino program are called **setup()** and **loop().** These functions need to be declared, which means that you need to tell the Arduino what these functions will do.
setup() and **loop()** are declared as you see on the right.

The **setup()** runs once, when the Arduino is first powered on.
This is where you configure the digital pins to be either inputs or outputs using a function named **pinMode()**. The digital pin (**LED_BUILTIN**) connected to LED will be **OUTPUT**. **pinMode()** takes two arguments: what pin to control, and set this pin as INPUT or OUTPUT.

The **loop()** runs continuously after the **setup()** has completed. The **loop()** is where you'll turn output ON and OFF.

digitalWrite() is the command that allows you to send 5V(HIGH) or 0V(LOW) to an output pin. **digitalWrite()** takes two arguments: what pin to control, and what value to set that pin, HIGH or LOW.

The **delay()** function lets you stop the Arduino from executing anything for a period of time. **delay()** takes an argument that determines the number of milliseconds before it executes the next set of code. There are 1000 milliseconds in one second.

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

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

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