00

Everyone, every day, uses technology. Most of us leave the programming to engineers because we think coding and electronics are complicated and difficult; actually, they can be fun and exciting activities. Thanks to Arduino, designers, artists, hobbyists and students of all ages are learning to create things that light up, move, and respond to people, animals, plants, and the rest of the world.

Over the years Arduino has been used as the "brain" in thousands of projects, one more creative than the last. A worldwide community of makers has gathered around this open-source platform, moving from personal computing to personal fabrication, and contributing to a new world of participation, cooperation and sharing.

Arduino is open and simple. It's founded on lessons we've learned teaching our own classes: if you start with the assumption that learning to make digital technologies is simple and accessible, you can make it so. Suddenly electronics and code become creative tools that anyone can use – like brushes and paint. This book walks you through the basics in a hands-on way, with creative projects you build by learning. Once you've mastered the basics, you'll have a palette of software and circuits that you can use to create something beautiful, and make someone smile with what you invent.

WELCOME TO ARDUINO!

ARDUINO MAKES IT AS EASY AS POSSIBLE TO PROGRAM TINY COMPUTERS CALLED MICROCONTROLLERS, WHICH ARE WHAT MAKE OBJECTS INTERACTIVE

You are surrounded by dozens of them every day: they are embedded in timers, thermostats, toys, remote controls, microwave ovens, even some toothbrushes. They just do one specific task, and if you hardly notice them – which is often the case – it's because they are doing it well. They have been programmed to sense and control activity using sensors and actuators.

Sensors listen to the physical world. They convert energy that you give off when you press buttons, or wave your arms, or shout, 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.

Microcontrollers listen to sensors and talk to actuators. They decide what to do based on a program that you write.

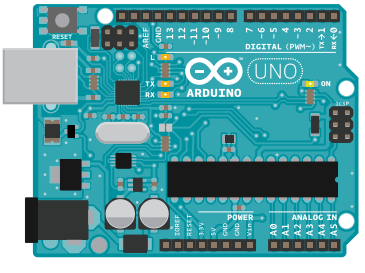
Microcontrollers and the electronics you attach to them are just the skeleton of your projects, though. You'll need to bring skills you probably already have to put some flesh on the bones.

For example, in one of the projects we suggest, you'll make an arrow and attach it to a motor, and put them both in a box with a knob, so you can make a meter to tell people whether you're busy or not. In another, you'll put some lights and a tilt switch on a cardboard frame to make an hourglass.

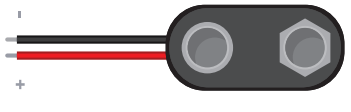
Arduino can make your projects responsive, but only you can make them beautiful. We'll provide some suggestions along the way as to how you might do that.

Arduino was designed to help you get things done. To make that happen, we 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 guides available. We'll provide a couple of references, and you can find more online at: arduino.cc/starterkit

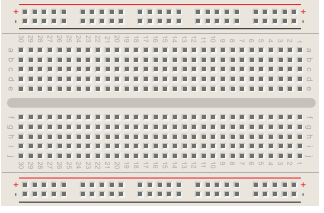
PARTS IN YOUR KIT



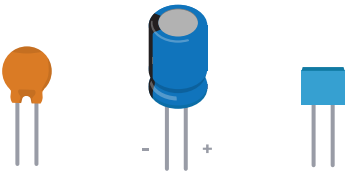
Arduino Uno - The microcontroller development board that will be at the heart of your projects. It's a simple computer, but one that has no way for you to interact with it yet. You will be building the circuits and interfaces for interaction, and telling the microcontroller how to interface with other components.



Battery Snap - Used to connect a 9V battery to power leads that can be easily plugged into a breadboard or your Arduino.



Breadboard - A board on which you can build electronic circuits. It's like a patch panel, with rows of holes that allow you to connect wires and components together. Versions that require soldering are available, as well as the solder-less type used here.



Capacitors - These components store and release electrical energy in a circuit. When the circuit's voltage is higher than what is stored in the capacitor, it allows current to flow in, giving the capacitor a charge. When the circuit's voltage is lower, the stored charge is released. Often placed across power and ground close to a sensor or motor to help smooth fluctuations in voltage.

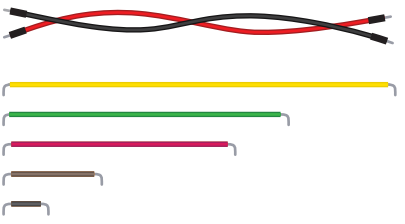
DC motor - Converts electrical energy into mechanical energy when electricity is applied to its leads. Coils of wire inside the motor become magnetized when current flows through them.

These magnetic fields attract and repel magnets, causing the shaft to spin. If the direction of the electricity is reversed, the motor will spin in the opposite direction.

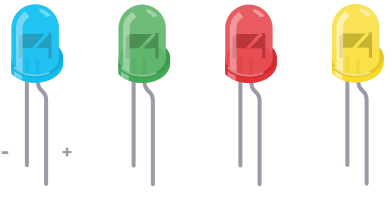
Diode - Ensures electricity only flows in one direction. Useful when you have a motor or other high current/voltage load in your circuit. Diodes are polarized, meaning that the direction that they're placed in a circuit matters. Placed one way, they allow current to pass through. Placed the other way, they block it. The anode side generally connects to the point of higher energy in your circuit. The cathode typically connects to the point of lower energy, or to ground. The cathode is usually marked with a band on one side of the component's body.

Gels (red, green, blue) - These filter out different wavelengths of light. When used in conjunction with photoresistors, they cause the sensor to only react to the amount of light in the filtered color.

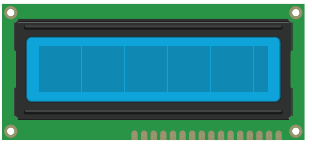
H-bridge - A circuit that allows you to control the polarity of the voltage applied to a load, usually a motor. The H-bridge in the kit is an integrated circuit, but it could also be constructed with a number of discrete components.



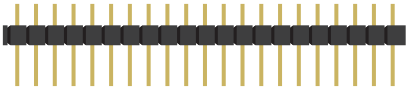
Jumper wires - Use these to connect components to each other on the breadboard, and to the Arduino.



Light Emitting Diodes (LEDs) - A type of diode that illuminates when electricity passes through it. Like all diodes, electricity only flows in one direction through these components. You're probably familiar with these as indicators on a variety of electronic devices. The anode, which typically connects to power, is usually the longer leg, and the cathode is the shorter leg.

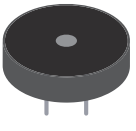


Liquid Crystal Display (LCD) - A type of alphanumeric or graphic display based on liquid crystals. LCDs are available in a many sizes, shapes, and styles. Yours has 2 rows with 16 characters each.



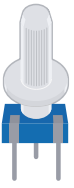
Male header pins - These pins fit into female sockets, like those on a breadboard. They help make connecting things much easier.

Optocoupler - This allows you to connect two circuits that do not share a common power supply. Internally there is a small LED that, when illuminated, causes a photoreceptor inside to close an internal switch. When you apply voltage to the + pin, the LED lights and the internal switch closes. The two outputs replace a switch in the second circuit.



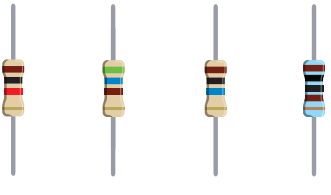
Piezo - An electrical component that can be used to detect vibrations and create noises.

Photoresistor - (also called a photocell, or light-dependent resistor). A variable resistor that changes its resistance based on the amount of light that falls on its face.

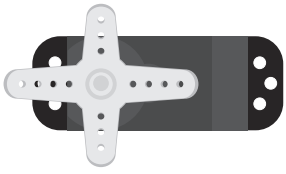


Potentiometer - A variable resistor with three pins. Two of the pins are connected to the ends of a fixed resistor. The middle pin, or wiper, moves across the resistor, dividing it into two halves. When the external sides of the potentiometer are connected to voltage and ground, the middle leg will give the difference in voltage as you turn the knob. Often referred to as a pot.

Pushbuttons - Momentary switches that close a circuit when pressed. They snap into breadboards easily. These are good for detecting on/off signals.



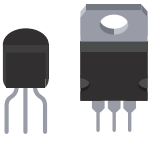
Resistors - Resist the flow of electrical energy in a circuit, changing the voltage and current as a result. Resistor values are measured in ohms (represented by the Greek omega character: Ω). The colored stripes on the sides of resistors indicate their value (see resistor color code table).



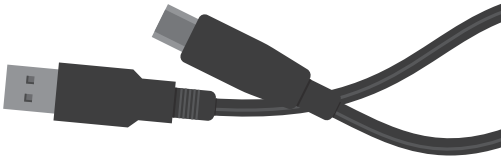
Servo motor - A type of geared motor that can only rotate 180 degrees. It is controlled by sending electrical pulses from your Arduino. These pulses tell the motor what position it should move to.

Temperature sensor - Changes its voltage output depending on the temperature of the component. The outside legs connect to power and ground. The voltage on the center pin changes as it gets warmer or cooler.

Tilt sensor - A type of switch that will open or close depending on its orientation. Typically they are hollow cylinders with a metal ball inside that will make a connection across two leads when tilted in the proper direction.



Transistor - A three legged device that can operate as an electronic switch. Useful for controlling high current/high voltage components like motors. One pin connects to ground, another to the component being controlled, and the third connects to the Arduino. When the component receives voltage on the pin connected to an Arduino, it closes the circuit between the ground and the other component.



USB Cable - This allows you to connect your Arduino Uno to your personal computer for programming. It also provides power to the Arduino for most of the projects in the kit.

UNCONNECTED WIRES

TRANSISTOR

PUSHBUTTON

TILT SWITCH

MOSFET

MOTOR

RESISTOR

PHOTO RESISTOR

POTENTIOMETER

LED

DIODE

PIEZO

POLARIZED CAPACITOR

CAPACITOR

BATTERY

GROUND

CONNECTED WIRES

In this book we will show you circuits both with realistic illustrations and with schematic diagrams.

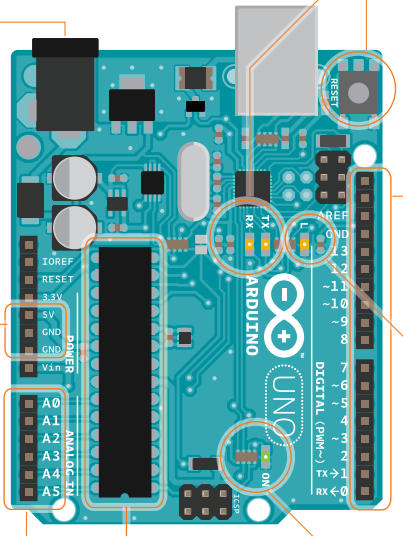
Illustrations will give you an idea of what the breadboard might look like in one possible implementation of the project. Schematics, instead, use symbols to capture the essence of circuits: they present the components and the ways they are connected in a clear, succinct, and unambiguous form, but not their physical organization. Schematics and schematic symbols are how we communicate about circuits. As you explore the world of electronics you will discover that some books and websites only provide schematic diagrams, so learning to read circuits this way is a valuable skill.

Here are the symbols we will be using throughout the book.

THE BOARD

Power connector
This is how you power your Arduino when it's not plugged into a USB port for power. Can accept voltages between 7-12V.

USB port
Used for powering your Arduino Uno, uploading your sketches to your Arduino, and for communicating with your Arduino sketch (via Serial.println() etc.)



GND and 5V pins
Use these pins to provide +5V power and ground to your circuits.

Analog in
Use these pins with analogRead().

Reset Button
Resets the ATmega microcontroller.

TX and RX LEDs
These LEDs indicate communication between your Arduino and your computer. Expect them to flicker rapidly during sketch upload as well as during serial communication. Useful for debugging.

Digital pins
Use these pins with digitalRead(), digitalWrite(), and analogWrite(). analogWrite() works only on the pins with the PWM symbol.

Pin 13 LED
The only actuator built-it to your Arduino Uno. Besides being a handy target for your first blink sketch, this LED is very useful for debugging.

ATmega microcontroller
The heart of your Arduino Uno.

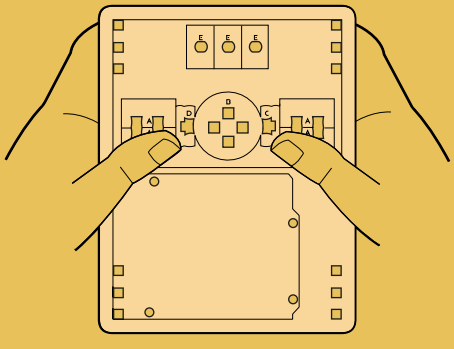
Power LED
Indicates that your Arudino is receiving power. Useful for debugging.

Your Starter Kit includes a pre-cut, easy-to-assemble wooden base that will make working on all your projects – whether they are from this book or not – even easier.

To build it, take the wood sheet out of the box and follow the instructions on the right.

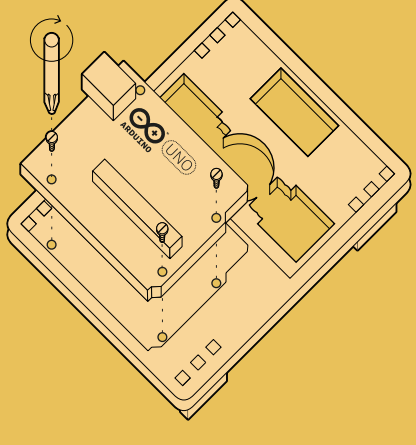
Be careful to use only the parts that are shown, but don't misplace any of the other pieces: you'll need them for some of the projects later.

Let's start!



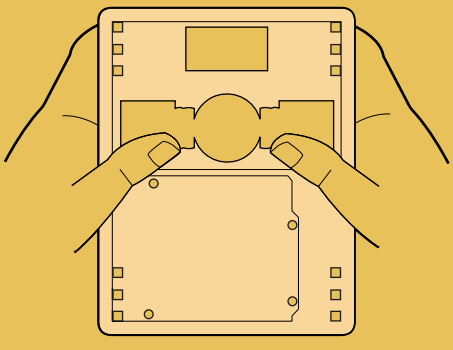
❶

Take the wood sheet and carefully separate the pieces.



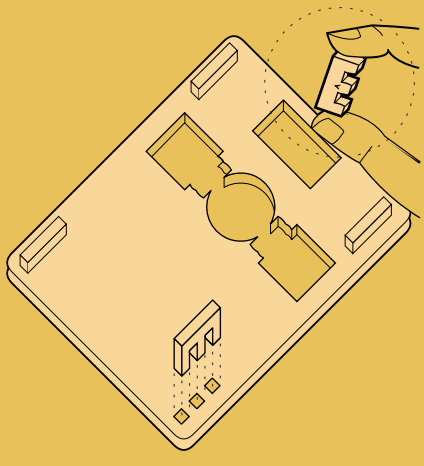
❹

Secure your Arduino Uno to the base using 3 screws. Be careful not to overtightens.



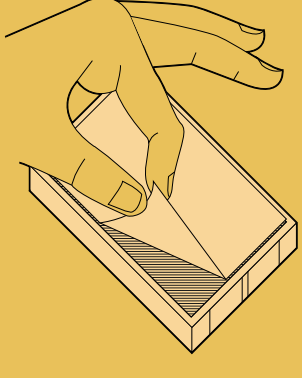
2

Go on until you've separated all the parts.



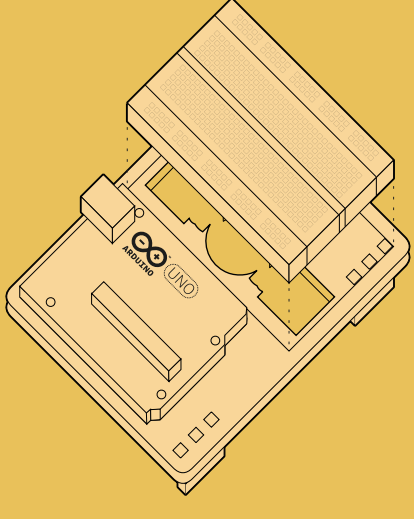
3

Place the pieces marked with an "A" into the holes in the corners, in order to create the feet of the base.



5

Carefully peel the backing from the breadboard.



6

Stick the breadboard on the wooden sheet, next to the Arduino UNO.

THINGS YOU NEED TO SUPPLY

9V battery

Small light source like a flashlight

Conductive material like aluminum foil or copper mesh

Colored paper

Scissors

An old CD or DVD

Tape and glue

A box that you can make holes into

Basic tools like a screwdriver

9V battery powered component
Any battery powered electronic device with at least one switch or pushbutton that you're willing to hack into will do the job.

Soldering iron and solder
(necessary only in Project 15)

SETTING UP

WELCOME TO ARDUINO! BEFORE YOU START CONTROLLING THE WORLD AROUND YOU, YOU'LL NEED TO DOWNLOAD THE IDE TO PROGRAM YOUR BOARD

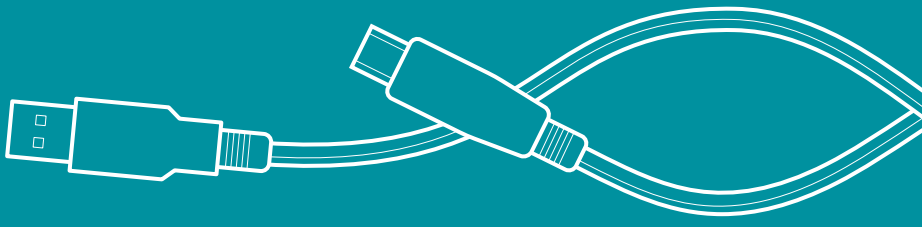
The Arduino IDE allows you to write programs and upload them to your Arduino.

Download the latest version of the IDE from:
arduino.cc/download

Have your Arduino board and USB cable near your computer. Don't plug them in just yet.

Follow the appropriate procedures in the next pages for Windows, Mac OS X or Linux.

The online version of this guide is available at:
arduino.cc/guide



WINDOWS INSTALLATION

Online version
arduino.cc/windows

INSTRUCTION FOR: WINDOWS 7, VISTA, AND XP

1. When the download of the IDE finishes, unzip the downloaded file. Make sure to preserve the folder structure. Double-click the folder to open it. There should be a few files and sub-folders inside.

2. Connect the Arduino to your computer with the USB cable. Your Arduino will automatically draw power from either the USB connection to the computer or an external power supply.
The green power light (labeled PWR) should turn on.

3. Windows should initiate its driver installation process when the board is plugged in. Your computer won't be able to find the drivers by itself, so you'll need to tell it where they are located.

— Click on the Start Menu and open the Control Panel.
— Navigate to "System and Security". Open the Device Manager.
— In Windows XP, look for the listing named "Ports (COM & LPT)" and right click on the "USB device" port; in Vista and Windows 7, right click on "Unknown device" under "Other devices".
— Choose "Update Driver Software".
— On Windows XP and Windows 7, you will be asked whether to install automatically or "with a path". Chose the second option, "with a path". On Windows Vista proceed directly to the next step.
— Select the "Browse my computer for Driver software" option.
— Navigate to the folder you unzipped in the earlier step. Locate and select the "Drivers" folder in the main Arduino folder (not the "FTDI USB Drivers" sub-directory). Press "OK" and "Next" to proceed.
— If you are prompted with a warning dialog about not passing Windows Logo testing, click "Continue Anyway".
— Windows now will take over the driver installation.

In the Device Manager, you should now see a port listing similar to "Arduino UNO (COM4)".

Congratulations! You've installed the Arduino IDE on your computer.

MAC OS X INSTALLATION

Online version
arduino.cc/mac

INSTRUCTION FOR:
OS X 10.5 AND LATER

1. When the download of the IDE finished, double-click the .zip fle. This will expand the Arduino application.
2. Copy the Arduino application into the Applications folder, or wherever else you wish to install the software.
3. Connect the board to the computer with the USB cable. The green power LED (labeled PWR) should turn on.
4. You do not need to install any drivers to work with the board. Depending on the version of OS X that you are running, you might get a dialog box asking if you wish to open the "Network Preferences". Click the "Network Preferences..." button, and then click "Apply".
5. The Uno will show up as "Not Configured", but it is still working. You can quit the System Preferences.

Congratulations! You have Arduino all set up and you're ready to start making projects.

LINUX INSTALLATION

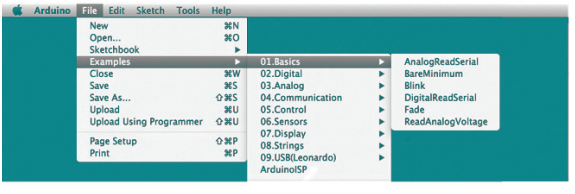
If you're using Linux, please visit the website for instructions:
arduino.cc/linux

COMMUNICATING WITH THE ARDUINO

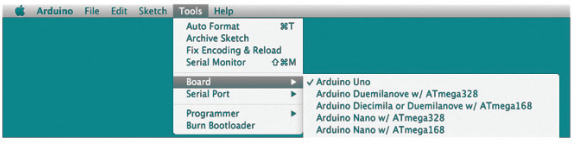
Now that you've installed the Arduino IDE and made sure your computer can talk to the board, it's time to make sure you can upload a program.

1. Double-click the Arduino application to open it. If the IDE loads in the wrong language, you can change this in the application preferences. Look for "Language Support" on this page for details: ***arduino.cc/ide***

2. Navigate to the LED blink example sketch ('sketch' is what Arduino programs are called). It's located under:
FILE > EXAMPLES > 01.BASICS > BLINK



3. A window with some text in it should have opened. Leave the window be for now, and select your board under:
TOOLS > BOARD menu



4. Choose the serial port your Arduino is connected to from the ***TOOLS > SERIAL PORT*** menu.

— *On Windows.* This is likely to be the COM with the highest number. There is no harm in guessing wrong, and if it doesn't work, try the next one. To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.

— *On Mac*. This should be something with /dev/tty.usbmodem in it. There are usually two of these; select either one.

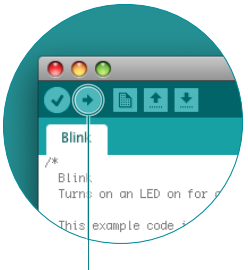


Fig. 1

5. To upload the Blink sketch to your Arduino, press the ***UPLOAD*** toggle in the top left corner of the window. See Fig. 1.

6 You should see a bar indicating the progress of the upload near the lower left corner of the Arduino IDE, and the lights labeled TX and RX on the Arduino board will be blinking. If the upload is successful, the IDE will display the message ***DONE UPLOADING***.

7 A few seconds after the upload has completed, you should see the yellow LED with an ***L*** next to it start blinking. See Fig. 2.
If this is the case, congratulations! You've successfully programmed the Arduino to blink its onboard LED!

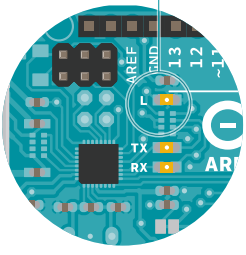


Fig. 2

Sometimes your brand new Arduino is already programmed with the Blink sketch, so you can't tell if you are truly in control. If this is the case, change the `delay` time by changing the number in the parenthesis to 100, and upload the Blink sketch again. Now the LED should blink much faster.
Congratulations! You really are in control! Now it's time to move on to Project 1. (You needn't save any changes you have made.)

ADDITIONAL INFORMATION

If you have problems with any of the steps outlined above, please see the troubleshooting suggestions:
arduino.cc/trouble

While you're getting ready to build your projects, you can look at the following page for additional information about the Arduino's programming environment:
arduino.cc/ide

You might also want to look at:

— the examples for using various sensors and actuators
arduino.cc/tutorial

— the reference for the Arduino language
arduino.cc/examples