# Arduino

search

# Guide to Getting Started with Arduino

You just got your Arduino in the mail and now you're ready to start having fun with Physical Computing? Here is the best place to start, with tutorials and guides to help you get started with Arduino hardware and software! From how to turn it on and plug it in to installing the driver and uploading your very first sketch program. These getting started guides are required reading for all new users!

### In This Quick-Start Guide...

This guide tell you everything you need to get started with Arduino and how to configure your setup so you can upload your first sketch. It consists of the following pages:

- Introduction: Read this introduction for an explanation of what Arduino is and why you'd want to use it.
- **Step-by-step instructions for how to get set up with your Arduino hardware and software. This is the most important part of the getting started guide!** Click whichever link matches your computer set-up:
    - Windows
    - Mac OS X
    - Linux
- If you're using one of the following Arduino variants or shields, they have their own introductory guides as well:
    - Arduino Nano
    - Arduino Mini
    - Arduino BT
    - LilyPad Arduino
    - Xbee shield
- Troubleshooting: HELP!!! Having problems? Read the troubleshooting guide for advice on what to do if things don't work.
- Sofware environment: A more detailed description of the Arduino software (development environment) and its parts.

### Other guides on the Arduino website

- Arduino booklet: an fun introduction to physical computing, electronics, and Arduino. Printable PDF with hand-drawn illustrations.

### More detailed guides, from our friends

- Learn electronics using Arduino: a great multi-part tutorial with tons of pictures, code, circuits, even video. Highly recommended.
- TodBot's course guides Longer presentation-format documents introducing Arduino from a Halloween hacking class taught by TodBot: class 1 (getting started), class 2 (input and sensors), class 3 (communication, servos, and pwm), class 4 (piezo sound & sensors, arduino+processing, stand-alone operation).
- Wiring electronics reference: circuit diagrams for connecting a variety of basic electronic components.
- Schematics to circuits: from Wiring, a guide to transforming circuit diagrams into physical circuits.
- Tom Igoe's Physical Computing Site: lots of information on electronics, microcontrollers, sensors, actuators, books, etc.

## What is Arduino?

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

## Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino
- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

## How do I use Arduino?

To get started, follow the instructions for your operating system: Windows, Mac OS X or Linux; or the additional instructions for your board: Arduino Mini, Arduino BT, or shield: Xbee.

## How To Get Arduino Running on Windows

*This document explains how to connect your Arduino board to the computer and upload your first sketch.*

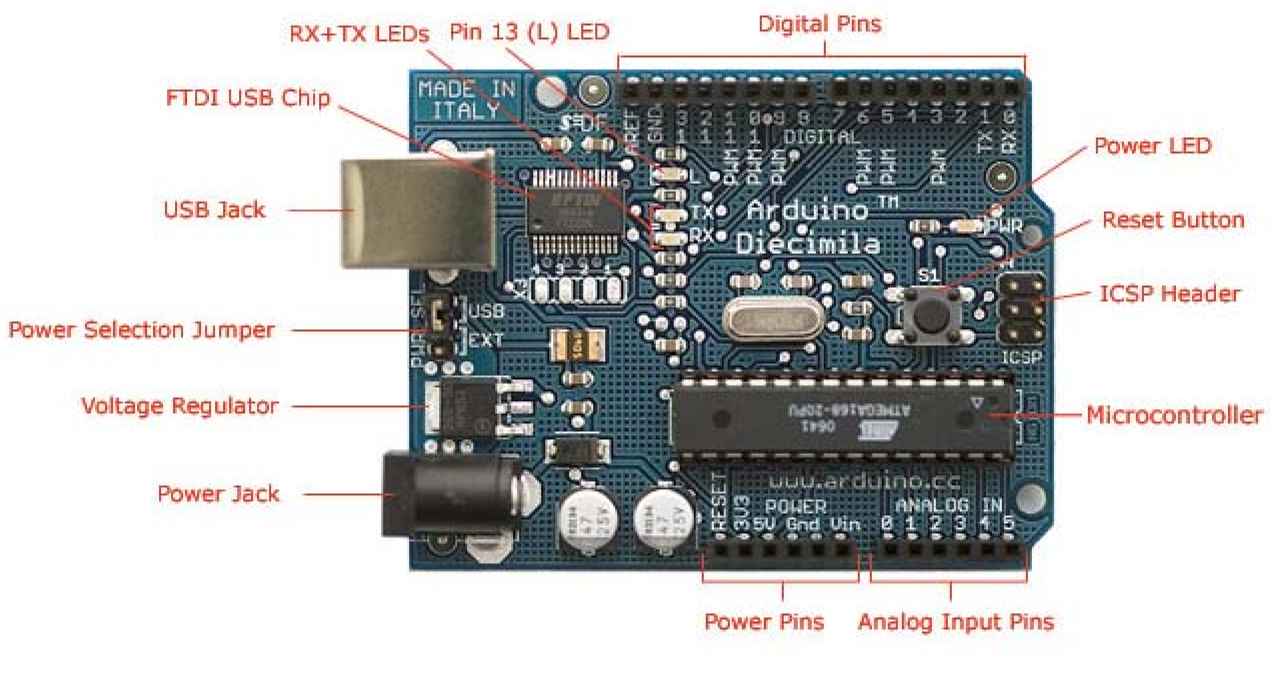
These are the steps that we'll go through:

1. Get an Arduino board and cable
2. Download the Arduino environment
3. Install the USB drivers
4. Connect the board
5. Connect an LED
6. Run the Arduino environment
7. Upload a program
8. Look for the blinking LED
9. Learn to use Arduino

### 1 | Get an Arduino board and cable

In this tutorial, we assume you're using an Arduino Diecimila. If you have another board, read the corresponding page in this getting started guide.

The Arduino Diecimila is a simple board that contains everything you need to start working with electronics and microcontroller programming. This diagram illustrates the major components of the board.



*Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.*

You also need a standard USB cable (A plug to B plug): the kind you would connect to a USB printer, for example.

### 2 | Download the Arduino environment

To program the Arduino board you need the Arduino environment.

**Download**: the latest version from the download page.

When the download 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.

### 3 | Locate the USB drivers

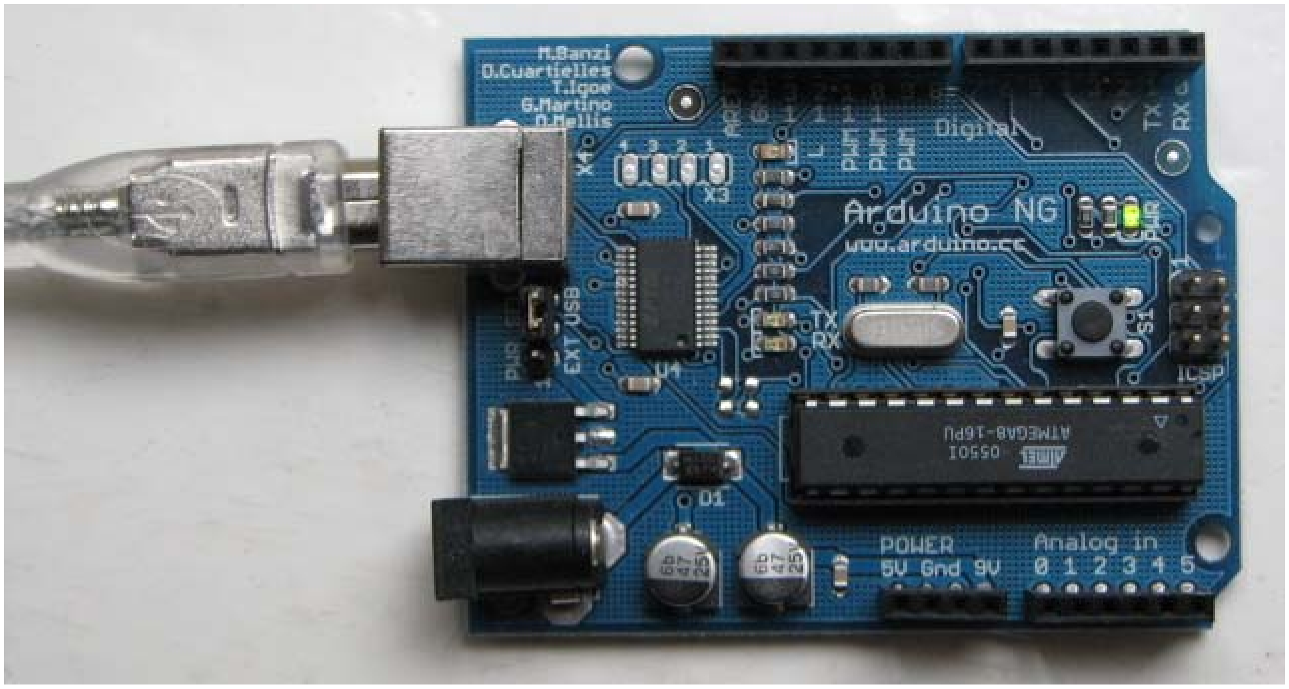
If you are using a USB Arduino, you will need to install the drivers for the FTDI chip on the board. These can be found in the `drivers/FTDI USB Drivers` directory of the Arduino distribution. In the next step ("Connect the board"), you will point Window's Add New Hardware wizard to these drivers.

The latest version of the drivers can be found on the FTDI website.

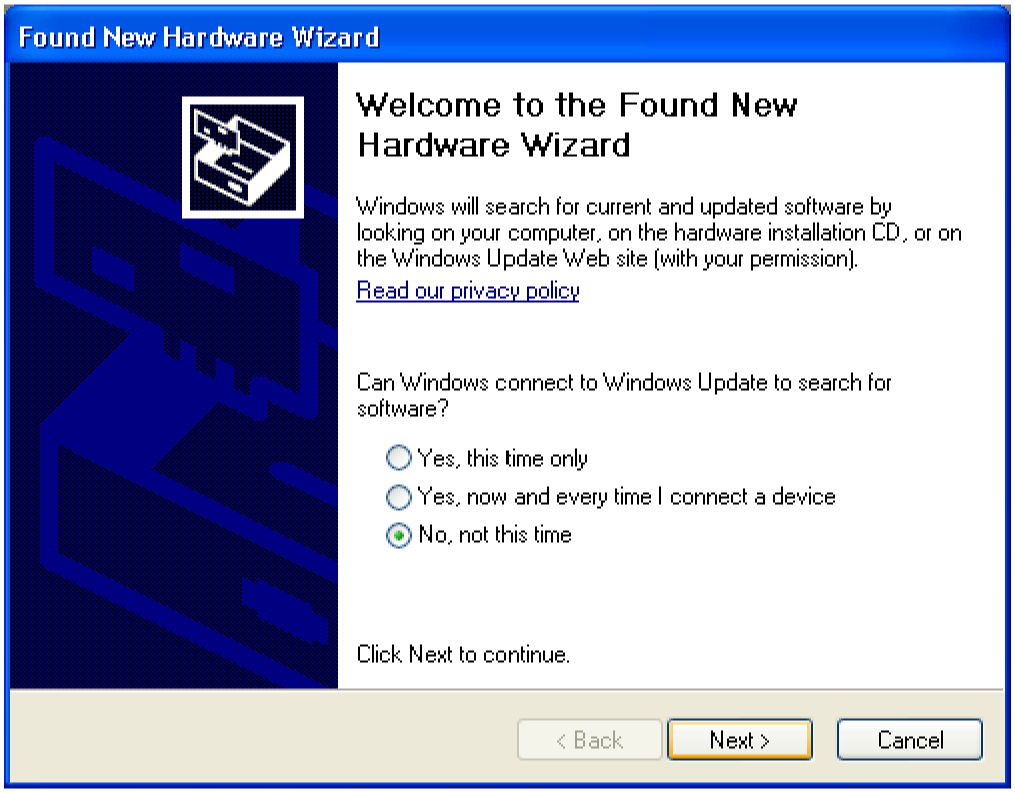
### 4 | Connect the board

The power source is selected by the jumper between the USB and power plugs. To power the board from the USB port (good for controlling low power devices like LEDs), place the jumper on the two pins closest to the USB plug. To power the board from an external power supply (6-12V), place the jumper on the two pins closest to the power plug. Either way, connect the board to a USB port on your computer.

The power LED should go on.



The Add New Hardware wizard will open. Tell it not to connect to Windows update and click next.


Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

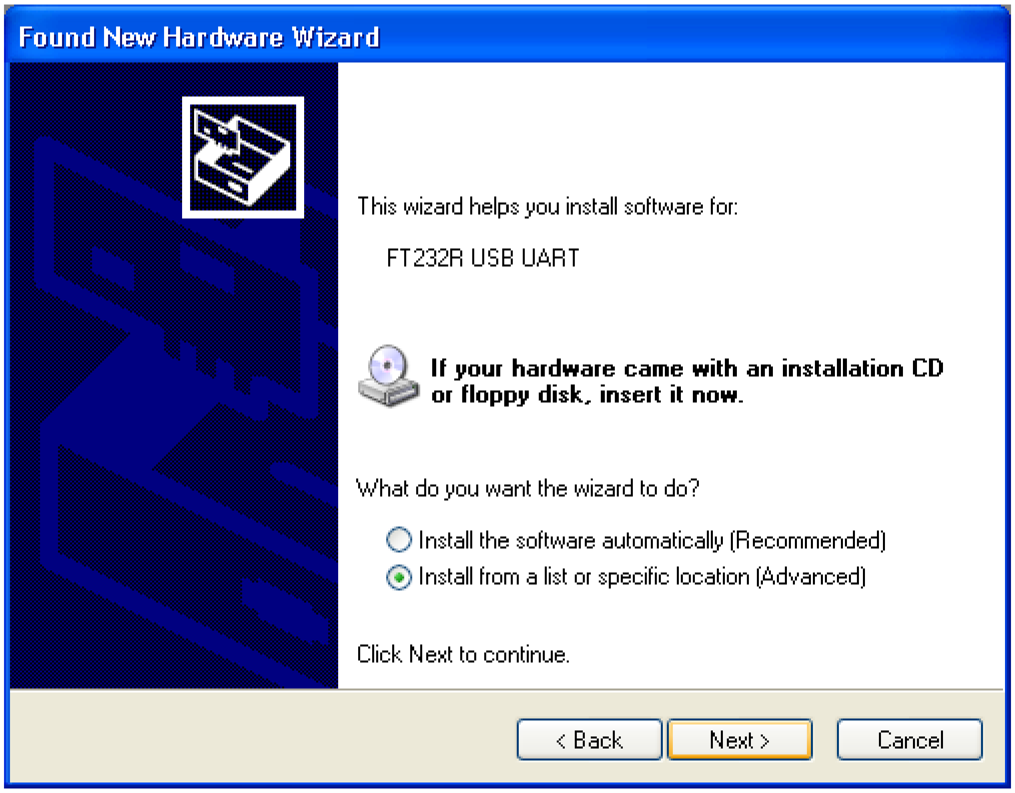
Read our privacy policy

Can Windows connect to Windows Update to search for software?

- Yes, this time only
- Yes, now and every time I connect a device
- [✓ No, not this time]

Click Next to continue.

Then select "Install from a list or specified location (Advanced)" and click next.


Found New Hardware Wizard

This wizard helps you install software for:

FT232R USB UART

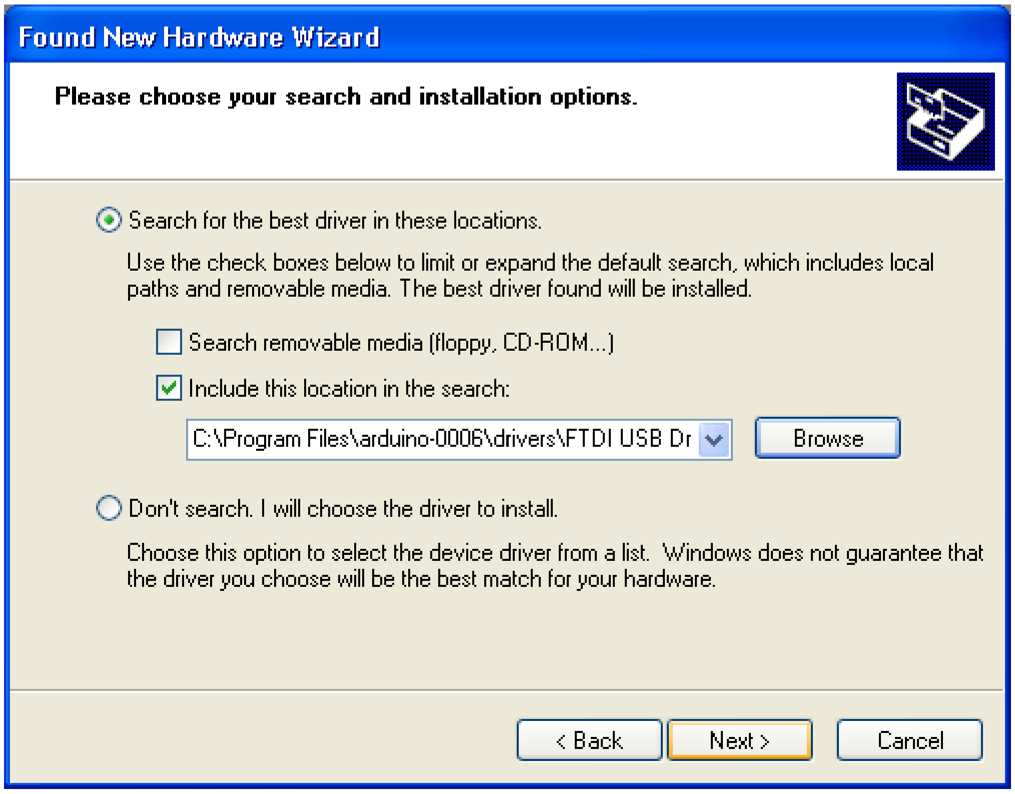
**If your hardware came with an installation CD or floppy disk, insert it now.**

What do you want the wizard to do?

- Install the software automatically (Recommended)
- [✓ Install from a list or specific location (Advanced)]

Click Next to continue.

Make sure that "Search for the best driver in these locations is checked"; uncheck "Search removable media"; check "Include this location in the search" and browse to the location you unzipped the USB drivers to in the previous step. Click next.


Found New Hardware Wizard

**Please choose your search and installation options.**

[✓ Search for the best driver in these locations.]

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

☐ Search removable media (floppy, CD-ROM...)

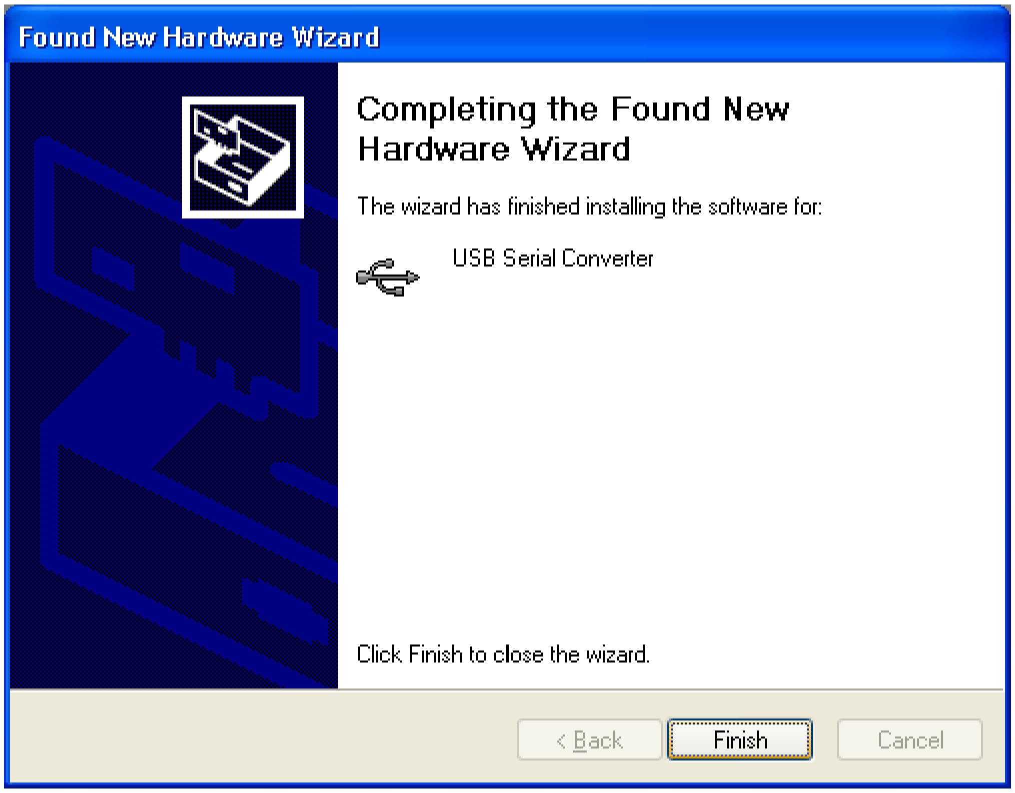
☑ Include this location in the search:

C:\Program Files\arduino-0006\drivers\FTDI USB Dr

○ Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

The wizard will search for the driver and then tell you that a "USB Serial Converter" was found. Click finish.


Found New Hardware Wizard

Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

USB Serial Converter

Click Finish to close the wizard.

The new hardware wizard will appear again. Go through the same steps. This time, a "USB Serial Port" will be found.

### 5 | Connect an LED (if you're using an older board)

The first sketch you will upload to the Arduino board blinks an LED. The Arduino Diecimila (and the original Arduino NG) has a built-in resistor on pin 13. On Arduino NG Rev. C and pre-NG Arduino boards, however, pin 13 does not have a built-in LED. On these boards, you'll need to connect the positive (longer) leg of an LED to pin 13 and the negative (shorter) leg to ground (marked "GND"). The LED will typically be flat on the side with the negative leg. Normally, you also need to use a resistor with the LED, but these boards have a resistor built-in on pin 13.

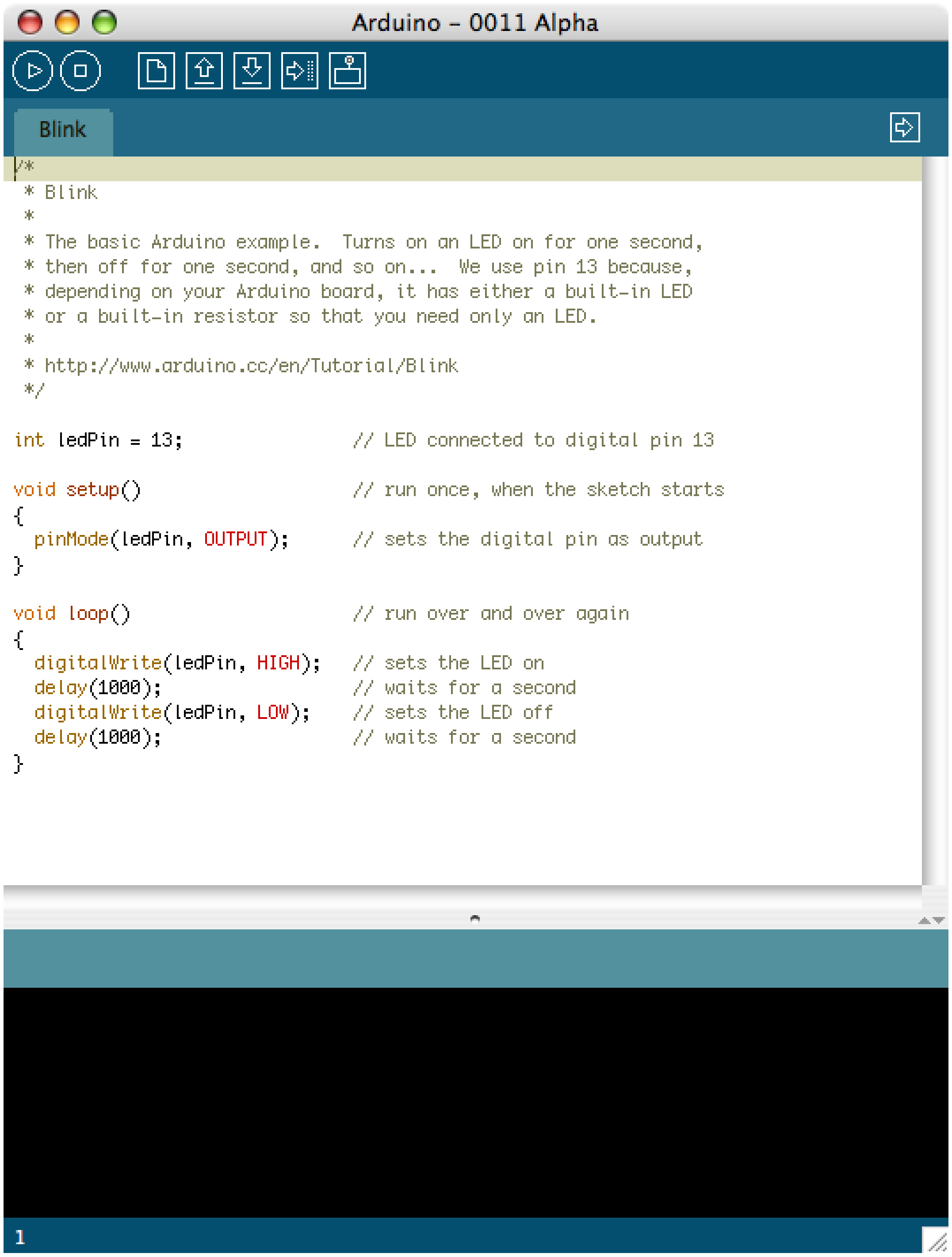
## 6 | Run the Arduino environment

Open the Arduino folder and double-click the Arduino application.

## 7 | Upload a program

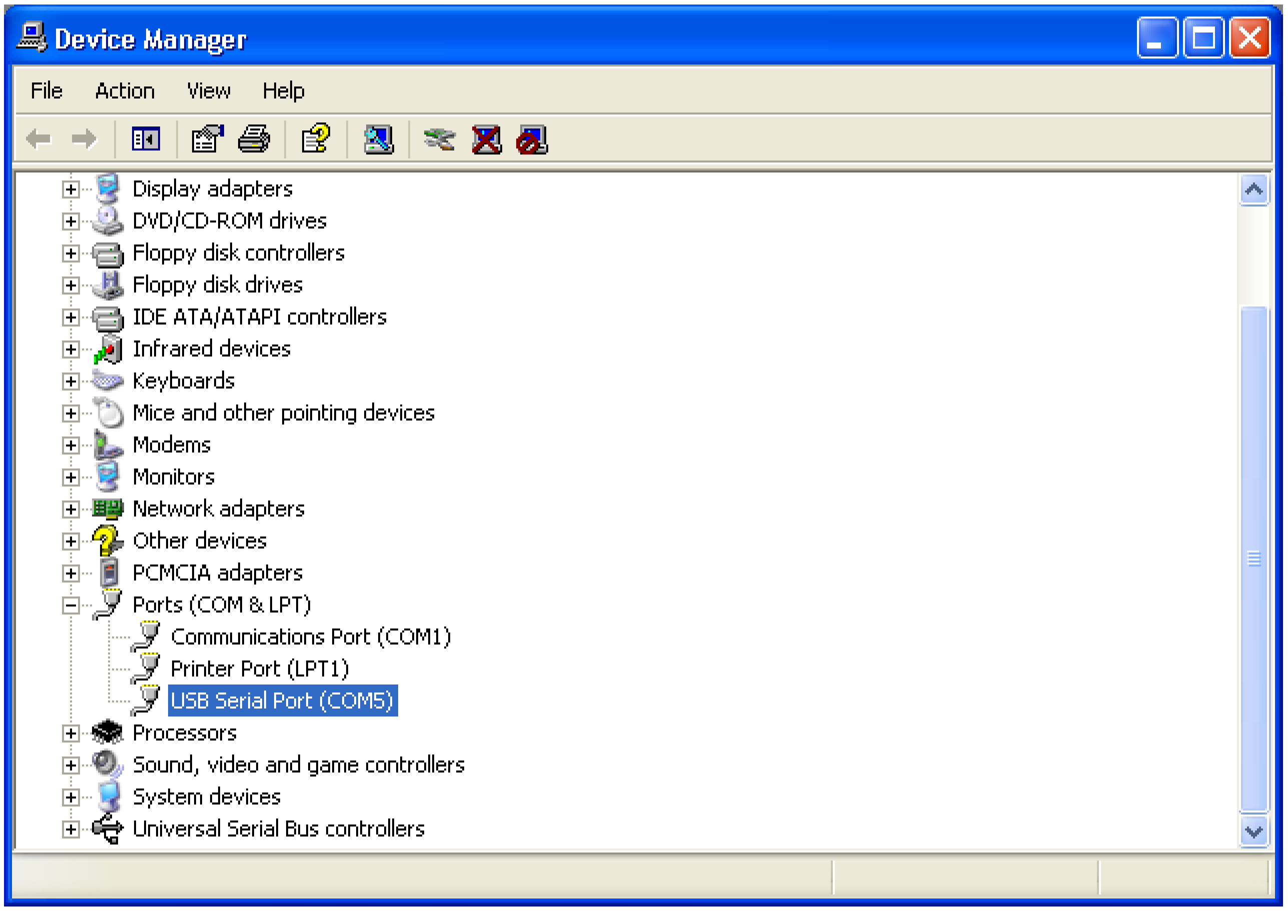
Open the LED blink example sketch: **File > Sketchbook > Examples > Digital > Blink**.

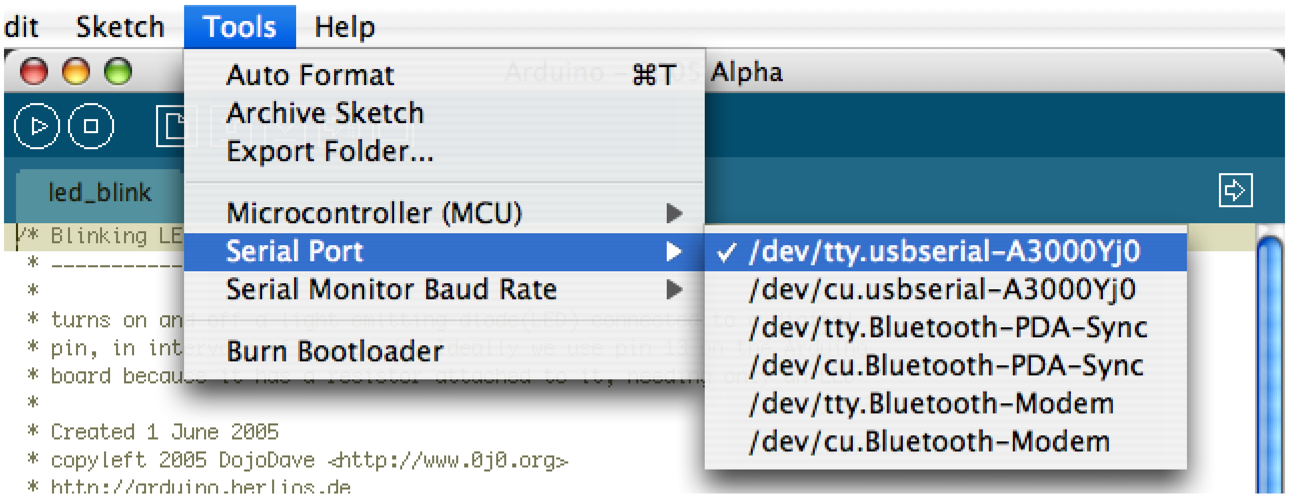
Here's what the code for the LED blink example looks like.



```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

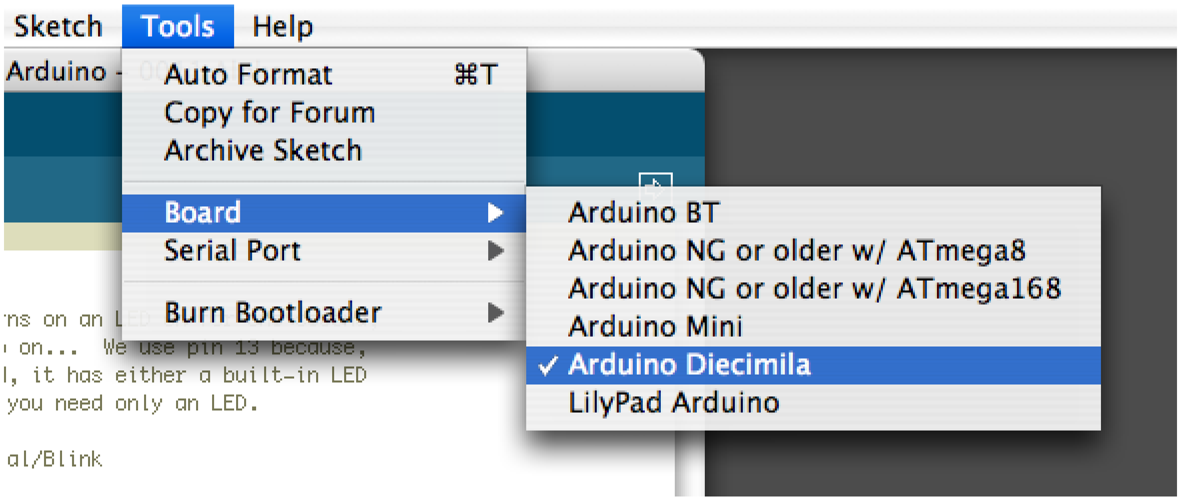
Select the serial device of the Arduino board from the Tools | Serial Port menu. On Windows, this should be COM1 or COM2 for a serial Arduino board, or COM3, COM4, or COM5 for a USB board. To find out, open the Windows Device Mananger (in the Hardware tab of System control panel). Look for a "USB Serial Port" in the Ports section; that's the Arduino board.





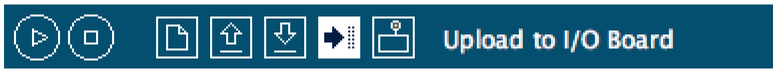
```
/* Blinking LE
 * ----------
 *
 * turns on an
 * pin, in int
 * board becau
 *
 * Created 1 June 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
```

Make sure that "Arduino Diecimila" is selected in the **Tools > Board** menu.



```
ns on an L
 on... We use pin 13 because,
l, it has either a built-in LED
you need only an LED.

al/Blink
```

Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar. (*Note:* If you have an Arduino Mini, NG, or other board, you'll need to physically present the reset button on the board immediately before pressing the upload button.)



### 8 | Look for the blinking LED

A few seconds after the upload finishes, you should see the amber (yellow) LED on the board start to blink. If it does, congratulations! You've gotten Arduino up-and-running.

If you have problems, please see the troubleshooting suggestions.

### 9 | Learn to use Arduino

- Tutorials: try these example programs.
- Reference: read the reference for the Arduino language.

## How To Get Arduino Running on Mac OS X (10.3.9 or later)

*This document explains how to connect your Arduino board to the computer and upload your first sketch.*

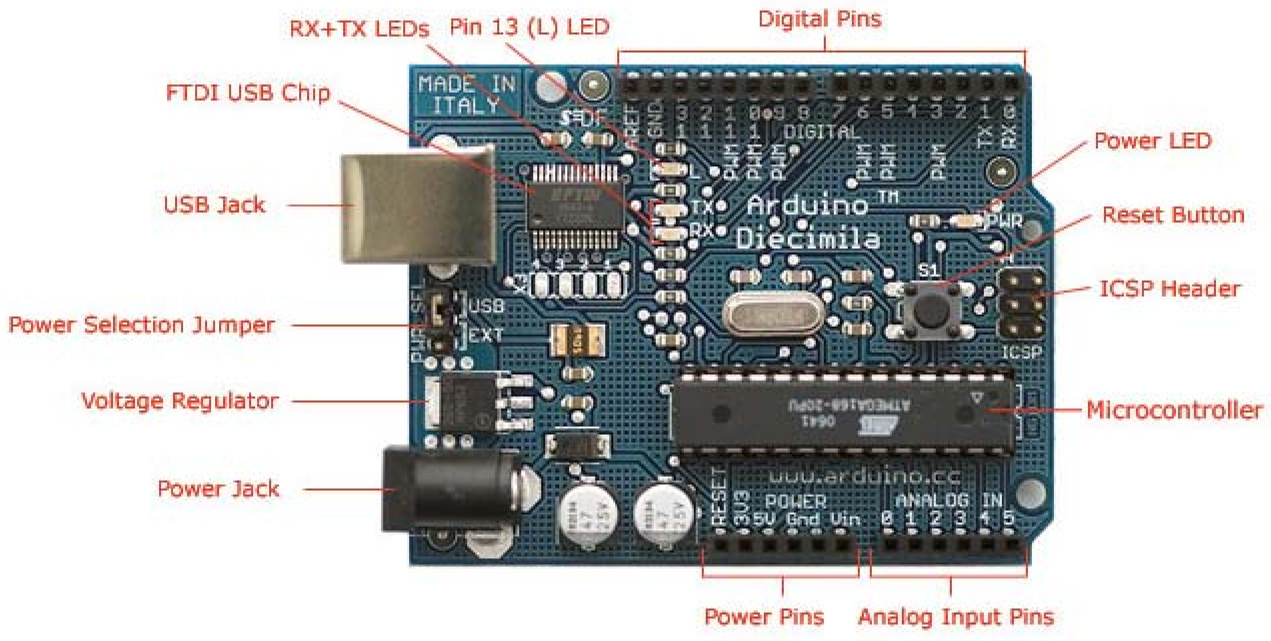
These are the steps that we'll go through:

1. Get an Arduino board and cable
2. Download the Arduino environment
3. Install the USB drivers
4. Connect the board
5. Connect an LED
6. Run the Arduino environment
7. Upload a program
8. Look for the blinking LED
9. Learn to use Arduino

### 1 | Get an Arduino board and cable

In this tutorial, we assume you're using an Arduino Diecimila. If you have another board, read the corresponding page in this getting started guide.

The Arduino Diecimila is a simple board that contains everything you need to start working with electronics and microcontroller programming. This diagram illustrates the major components of the board.



*Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.*

You also need a standard USB cable (A plug to B plug): the kind you would connect to a USB printer, for example.

### 2 | Download the Arduino environment

To program the Arduino board you need the Arduino environment.

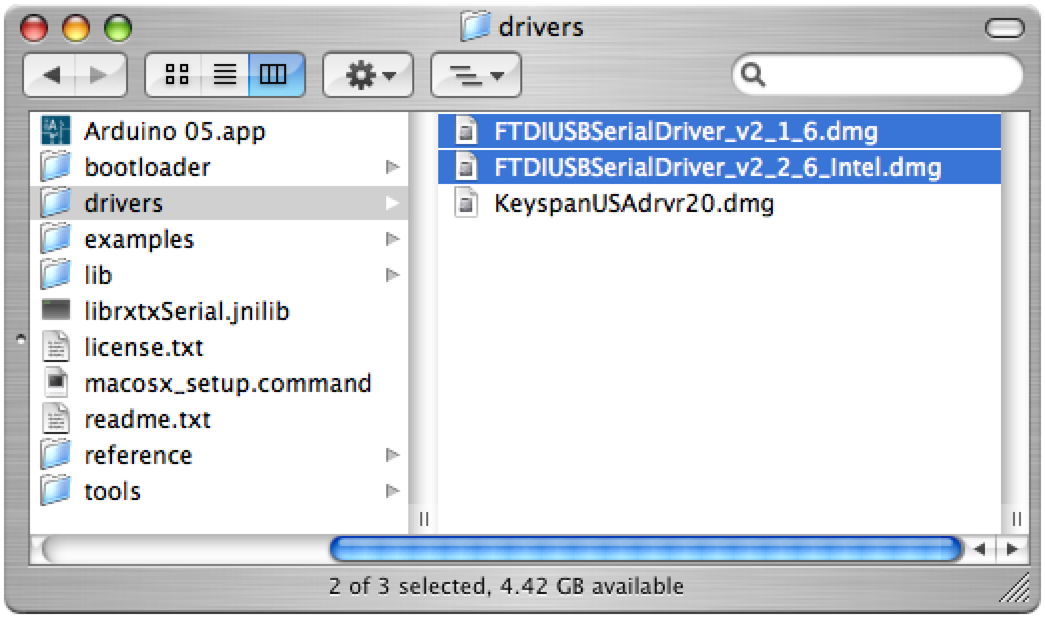
**Download**: the latest version from the download page.

When the download 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.

### 3 | Install the USB drivers

If you are using a USB Arduino, you will need to install the drivers for the FTDI chip on the board. These can be found in the **drivers** directory of the Arduino distribution.

If you have an older Mac like a Powerbook, iBook, G4 or G5, you should use the the PPC drivers: `FTDIUSBSerialDriver_v2_1_9.dmg`. If you have a newer Mac like an MacBook, MacBook Pro, or Mac Pro, you need the Intel drivers: `FTDIUSBSerialDriver_v2_2_9_Intel.dmg`. Double-click to mount the disk image and run the included `FTDIUSBSerialDriver.pkg`.

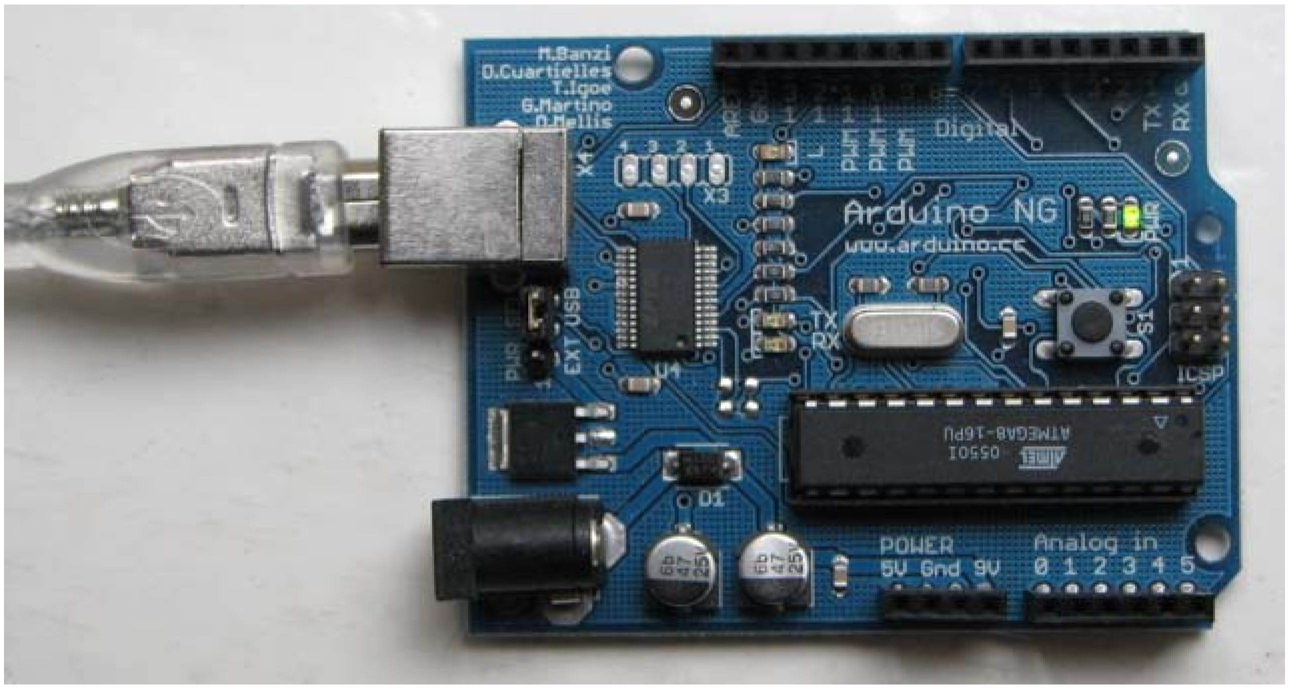


(The latest version of the drivers can be found on the FTDI website.)

### 4 | Connect the board

The power source is selected by the jumper between the USB and power plugs. To power the board from the USB port (good for controlling low power devices like LEDs), place the jumper on the two pins closest to the USB plug. To power the board from an external power supply (6-12V), place the jumper on the two pins closest to the power plug. Either way, connect the board to a USB port on your computer.

The power LED should go on.



### 5 | Connect an LED (if you're using an older board)

The first sketch you will upload to the Arduino board blinks an LED. The Arduino Diecimila (and the original Arduino NG) has a built-in resistor on pin 13. On Arduino NG Rev. C and pre-NG Arduino boards, however, pin 13 does not have a built-in LED. On these boards, you'll need to connect the positive (longer) leg of an LED to pin 13 and the negative (shorter) leg to ground (marked "GND"). The LED will typically be flat on the side with the negative leg. Normally, you also need to use a resistor with the LED, but these boards have a resistor built-in on pin 13.

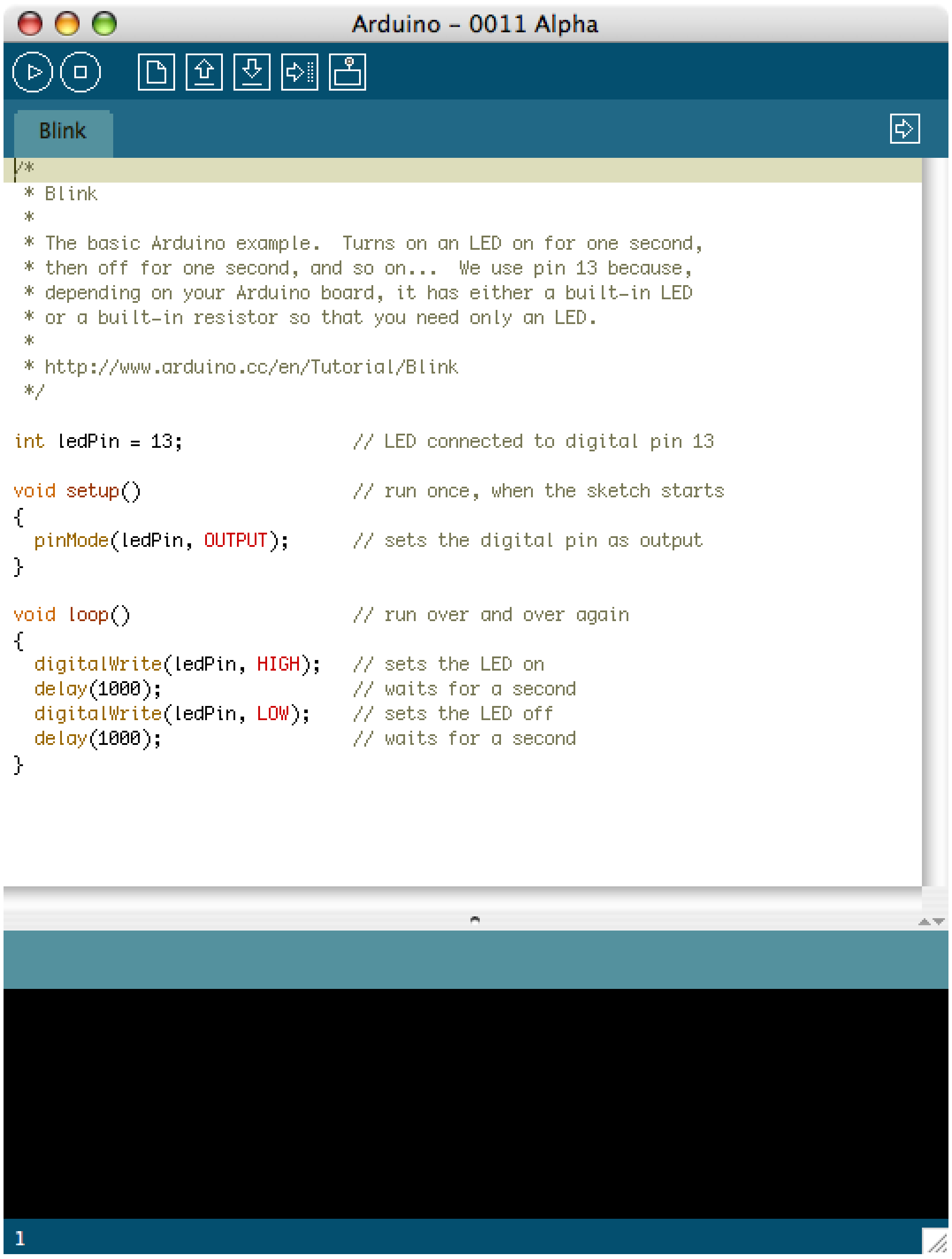
### 6 | Run the Arduino environment

Open the Arduino folder and double-click the Arduino application.

### 7 | Upload a program

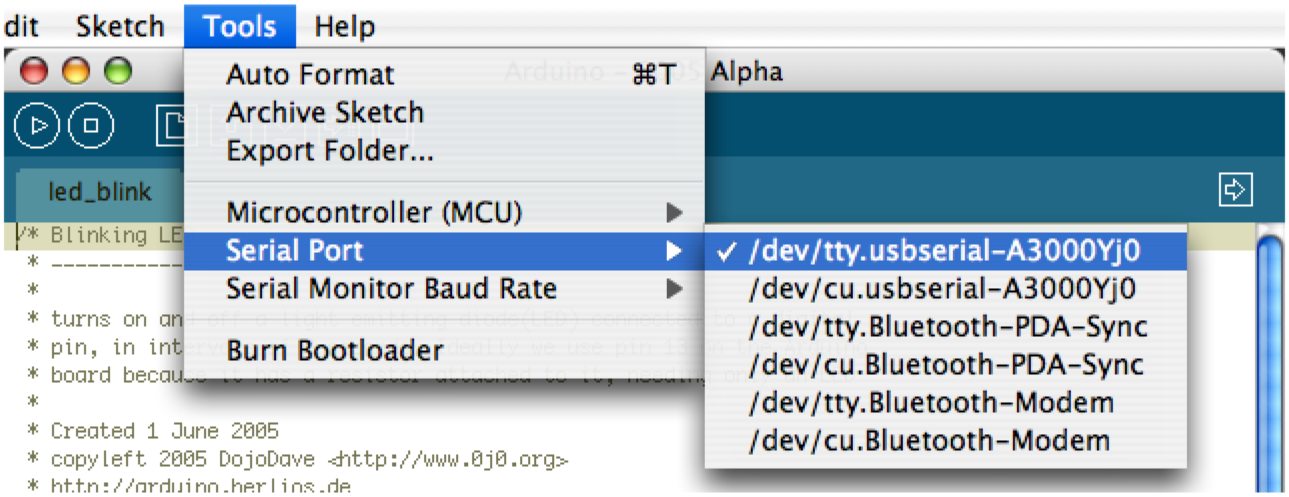
Open the LED blink example sketch: **File > Sketchbook > Examples > Digital > Blink**.

Here's what the code for the LED blink example looks like.



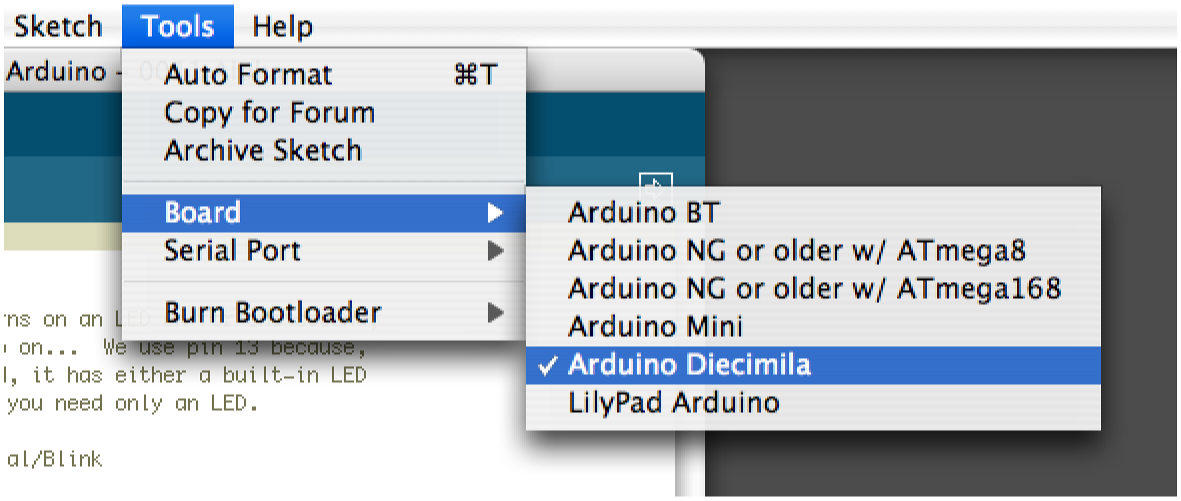
```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

Select the serial device of the Arduino board from the Tools | Serial Port menu. On the Mac, this should be something with `/dev/tty.usbserial` in it.



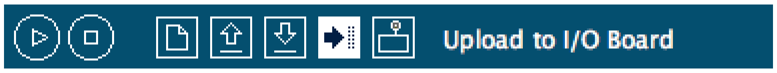
```
/* Blinking LE
 * ----------
 *
 * turns on an
 * pin, in int
 * board becau
 *
 * Created 1 June 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
```

Make sure that "Arduino Diecimila" is selected in the **Tools > Board** menu.



```
ns on an L
 on...  We use pin 13 because,
l, it has either a built-in LED
you need only an LED.

al/Blink
```

Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar. (*Note:* If you have an Arduino Mini, NG, or other board, you'll need to physically present the reset button on the board immediately before pressing the upload button.)



## 8 | Look for the blinking LED

A few seconds after the upload finishes, you should see the amber (yellow) LED on the board start to blink. If it does, congratulations! You've gotten Arduino up-and-running.

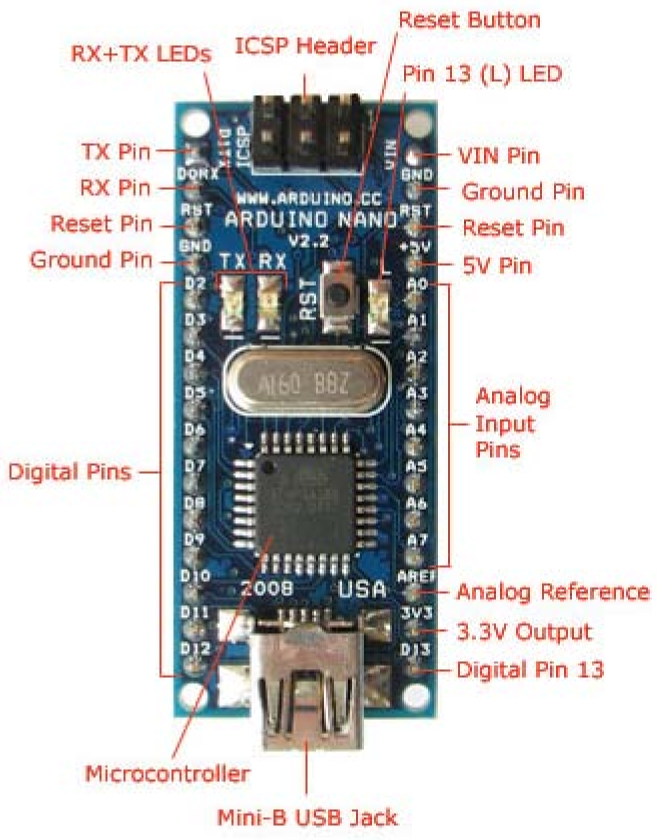
If you have problems, please see the troubleshooting suggestions.

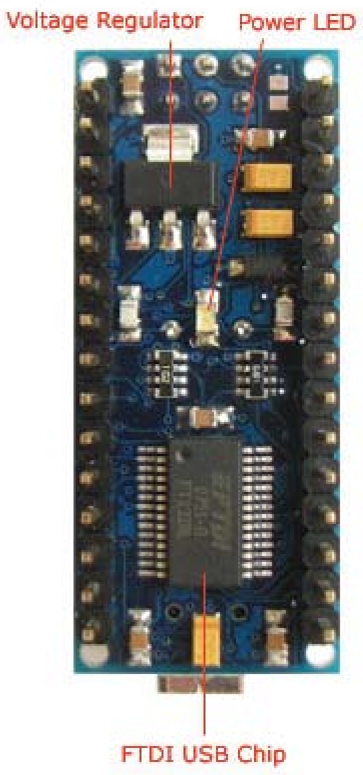
## 9 | Learn to use Arduino

- Tutorials: try these example programs.
- Reference: read the reference for the Arduino language.

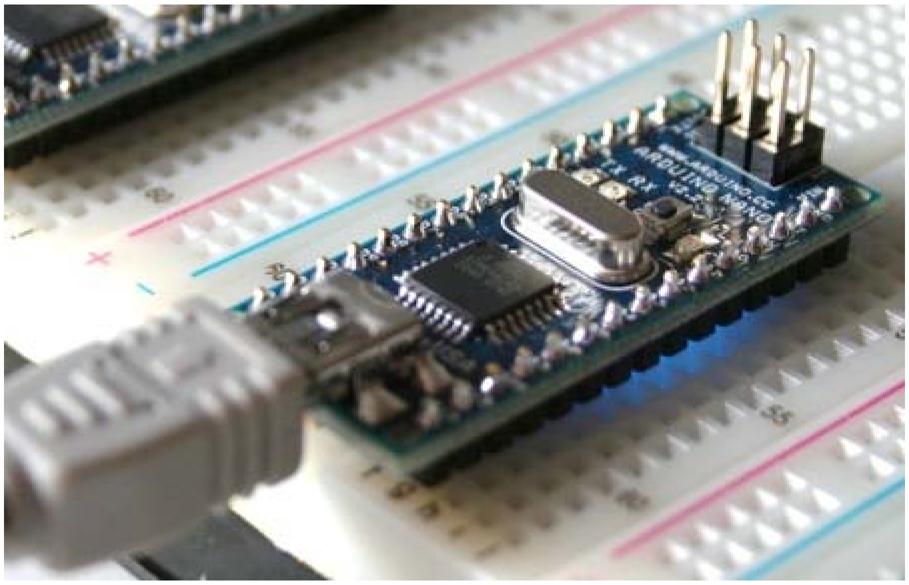
## Guide to the Arduino Nano

*This is a preliminary guide and will be updated and expanded in the next few days.*





*The various components of the Arduino Nano.*



*Connecting the Arduino Nano to a computer with a Mini-B USB cable. Note the blue power LED underneath the board.*

To connect the Arduino Nano to your computer, you'll need a Mini-B USB cable. This also provides power to the board, as indicated by the blue LED on the bottom.

To upload a sketch to the Nano, make sure you have the **Arduino Diecimila** option selected from the **Tools > Board** menu and the correct serial port selected from the **Tools > Serial Port** menu. Then simply press the upload button in the Arduino environment. The board will automatically reset and the sketch will be uploaded, as with the Arduino Diecimila. If you have any problems, see the troubleshooting guide.

For more details on the Arduino Nano, see the hardware page.

# Guide to the Arduino Mini

To get started with the Arduino Mini, follow the directions for the regular Arduino on your operating system (Windows, Mac OS X, Linux), with the following modifications:

- Connecting the Arduino Mini is a bit more complicated than a regular Arduino board (see below for instructions and photos).
- You need to select **Arduino Mini** from the **Tools | Board** menu of the Arduino environment.

## Information about the Arduino Mini

The microcontroller (an ATmega168) on the Arduino Mini is a physically smaller version of the chip on the USB Arduino boards, with the following small difference:

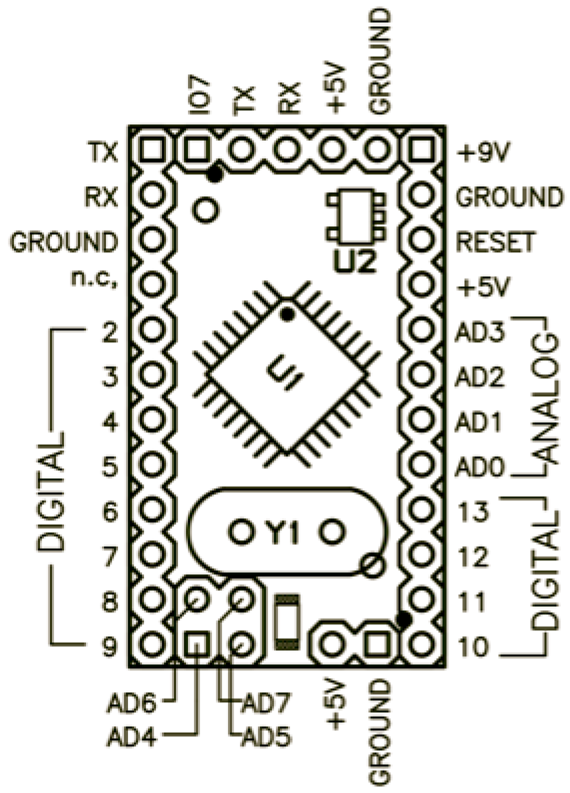
- There are two extra analog inputs on the Mini (8 total). Four of these, however, are not connected to the legs that come on the Arduino Mini, requiring you to solder wires to their holes to use them. Two of these unconnected pins are also used by the Wire library (I2C), meaning that its use will require soldering as well.

Also, the Arduino Mini is more **fragile and easy to break** than a regular Arduino board.

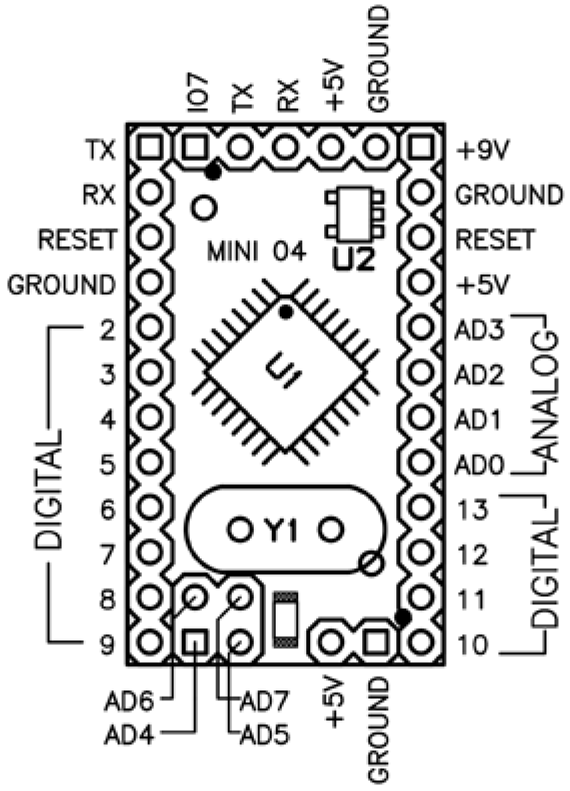
- Don't connect more than 9 volts to the +9V pin or reverse the power and ground pins of your power supply, or you might kill the ATmega168 on the Arduino Mini.
- You can't remove the ATmega168, so if you kill it, you need a new Mini.

## Connecting the Arduino Mini

Here's a diagram of the pin layout of the Arduino Mini:



*Mini 03 pinout* (compatible with earlier revisions)



*Mini 04 pinout* (the ground on the left has moved down one pin)

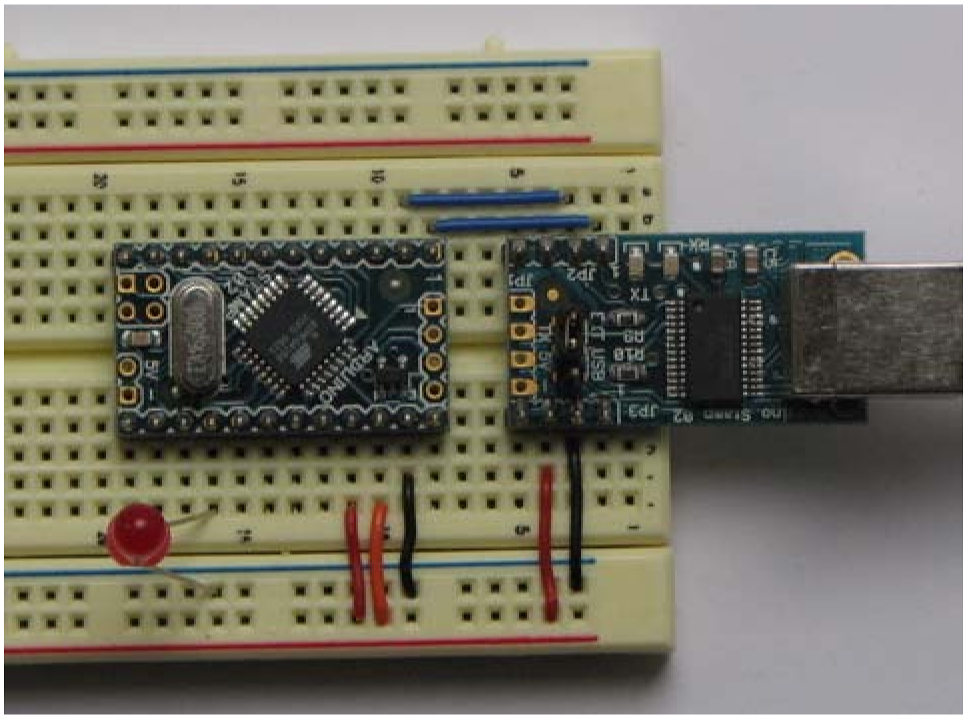
To use the Arduino Mini, you need to connect:

- Power. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or an Arduino NG) connected to the +5V pin of the Arduino Mini. Or, a +9V power source (e.g. a 9 volt battery) connected to the +9V pin of the Arduino Mini.
- Ground. One of the ground pins on the Arduino Mini must be connected to ground of the power source.
- TX/RX. These pins are used both for uploading new sketches to the board and communicating with a computer or other device.
- Reset. Whenever this pin is connected to ground, the Arduino Mini resets. You can wire it to a pushbutton, or connect it to +5V to prevent the Arduino Mini from resetting (except when it loses power). If you leave the reset pin unconnected, the Arduino Mini will reset randomly.
- An LED. While not technically necessary, connecting an LED to the Arduino Mini makes it easier to check if it's working. Pin 13 has a 1 KB resistor on it, so you can connect an LED to it directly between it and ground. When using another pin, you will need an external resistor.

You have a few options for connecting the board: the Mini USB Adapter, a regular Arduino board, or your own power supply and USB/Serial adapter.

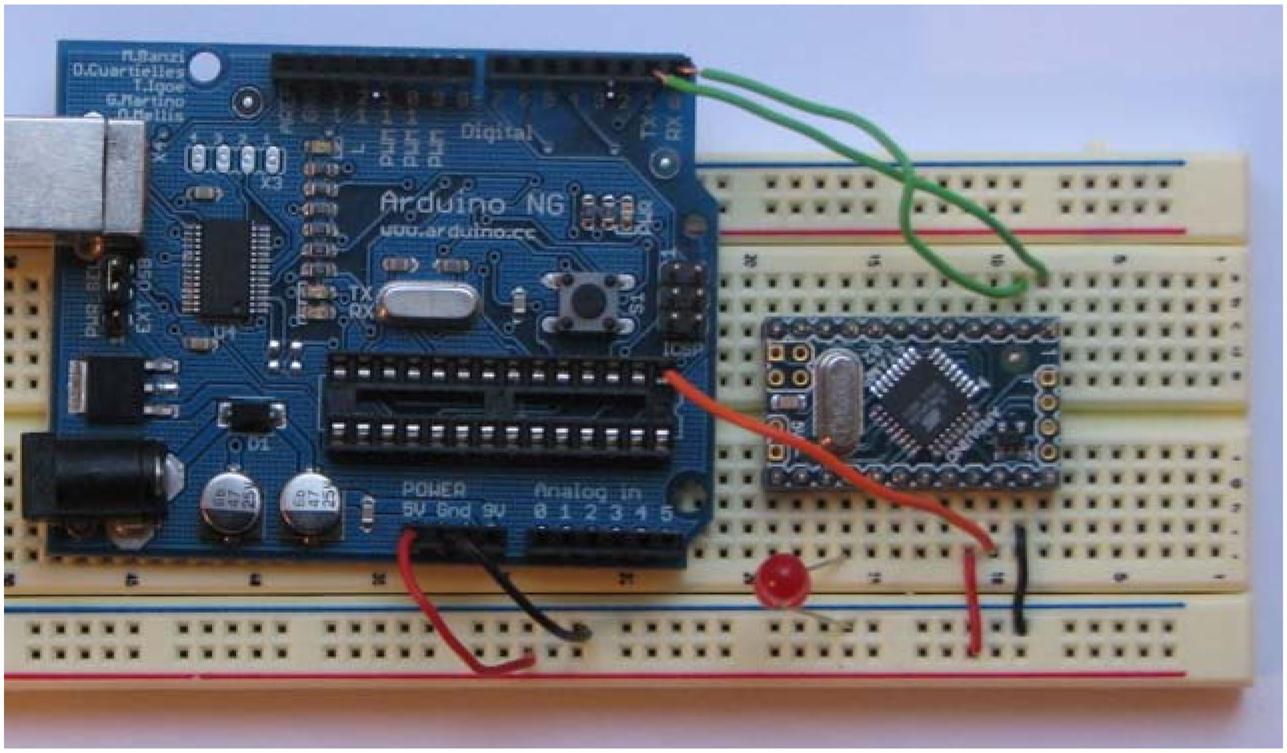
**Connecting the Arduino Mini and Mini USB Adapter**

Here is a photo showing the Arduino Mini connected to the Mini USB adapter. Notice that the reset pin is connected directly to +5V (the orange wire), without a pushbutton. Thus, to reset the Arduino Mini, you will need to unplug and reconnect the USB cable to the Mini USB Adapter, or manually move the orange wire connected to the reset pin from +5V to ground and back.



**Connecting the Arduino Mini and a regular Arduino**

Here's a photo of the Arduino Mini connected to an Arduino NG. The NG has its ATmega8 removed and is being used for its USB connection, power source, and reset button. Thus, you can reset the Arduino Mini just by pressing the button on the NG.



# ArduinoBT

The Arduino BT is an Arduino board with built-in bluetooth module, allowing for wireless communication. To get started with the Arduino BT, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux), with the following modifications:

- First, pair the Arduino BT with your computer and create a virtual serial port for it. Look for a bluetooth device called **ARDUINOBT** and the pass code is **12345**.
- Select **Arduino BT** from the **Tools | Board** menu of the Arduino environment.

## Information about the Arduino BT

In most respects, the Arduino BT is similar to the Arduino Diecimila. Here are the main differences of BT board (besides the fact that it communicates over bluetooth instead of USB):

- The Arduino BT is more fragile and easy to break than a regular Arduino board.
- **Don't power the board with more than 5.5 volts to the or reverse the polarity (power and ground pins) of your power supply, or you might kill the ATmega168 on the Arduino BT.** The Arduino BT can, however, run with a minimum of 1.2 volts, making it easier to power with batteries.
- The microcontroller (an ATmega168) on the Arduino BT is a physically smaller version of the chip on the USB Arduino boards. You can't remove it, so if you kill it, you need a new Arduino BT.
- There are two extra analog inputs on the Arduino BT (8 total). Two of these, however, are not connected to the pin headers on the board; you'll need to solder something to the pads next to the numbers "6" and "7".
- Pin 7 is connected to the reset pin of the bluetooth module; **don't use it for anything** (except resetting the module).

For more details, see the Arduino BT hardware page.

## Using the Arduino BT

The on-board serial communication between the bluetooth module and the Arduino sketch (running on the ATmega168) needs to be at 115200 baud (i.e. call Serial.begin(115200) in your setup() function). Communication between the bluetooth module and the computer can be at any baud rate.

Communication between the BT module and the computer can be temperamental. You might want to open the serial monitor a couple of seconds after resetting the board.

## Guide to the LilyPad Arduino

To get started with the LilyPad Arduino, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux. Connecting the LilyPad Arduino is a bit more complicated than a regular Arduino board (see below for instructions and photos).

The LilyPad Arduino is more **fragile and easy to break** than a regular Arduino board. Don't connect more than 5.5 volts to the + tab or reverse the power and ground pins of your power supply, or you will very likely kill the ATmega168V on the LilyPad Arduino. You can't remove the ATmega168V, so if you kill it, you need a new LilyPad.

Note: More information about getting started with the LilyPad Arduino can be found here:
http://www.cs.colorado.edu/~buechley/LilyPad

### Connecting the LilyPad Arduino

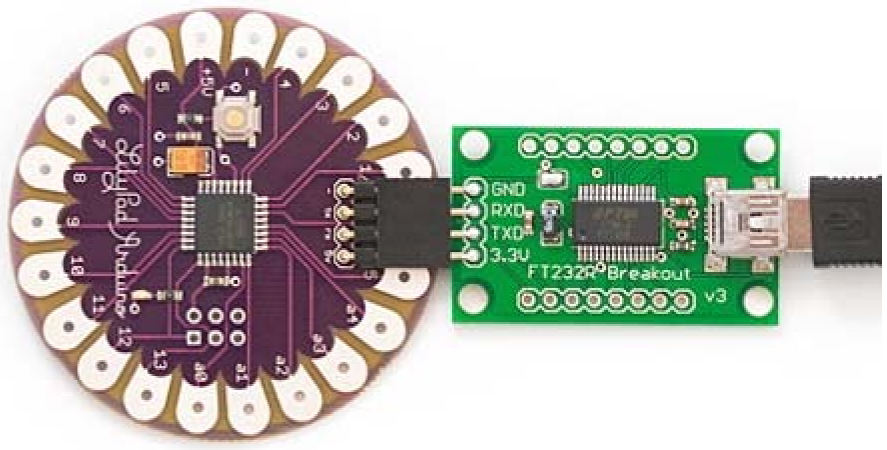
To program the LilyPad Arduino, you need to connect it to your computer. To do this, you'll need to connect:

- Power. Power should be connected to the + tab on the LilyPad Arduino. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or the + tab of a LilyPad power supply) or, another 2.7-5.5V power source (e.g. a 3.7V rechargeable Lithium Ion battery or 2 AA batteries in series).
- Ground. The ground tab on the LilyPad Arduino must be connected to ground of the power source.
- TX/RX. These tabs are used both for uploading new sketches to the board and communicating with a computer or other device.

You have a few options for connecting the board to your computer: the SparkFun LilyPad USB Link, the Mini USB Adapter, a regular Arduino board, or your own power supply and USB/Serial adapter.

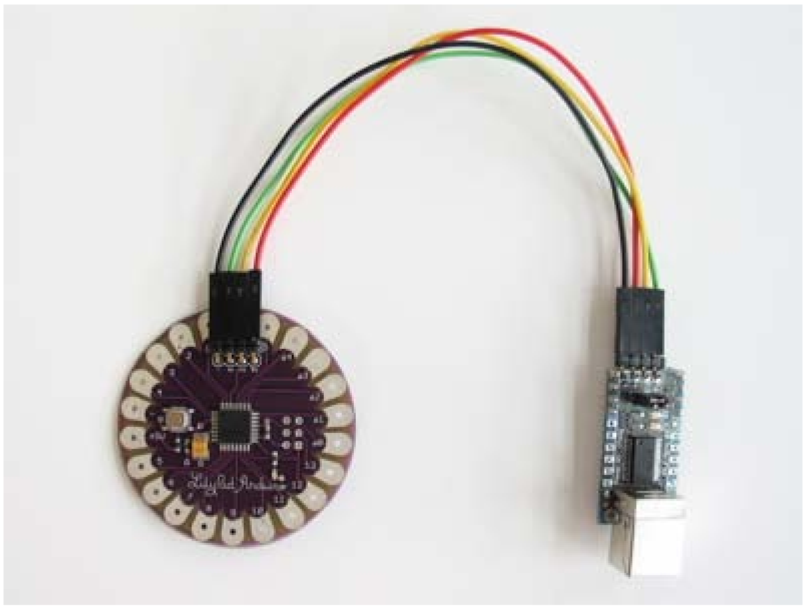
#### Use the SparkFun LilyPad USB Link

The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to the -, tx, rx, 5v labeled holes at the top of your LilyPad to make the connection. The LilyPad USB Link is available here and right angle male headers are available here.

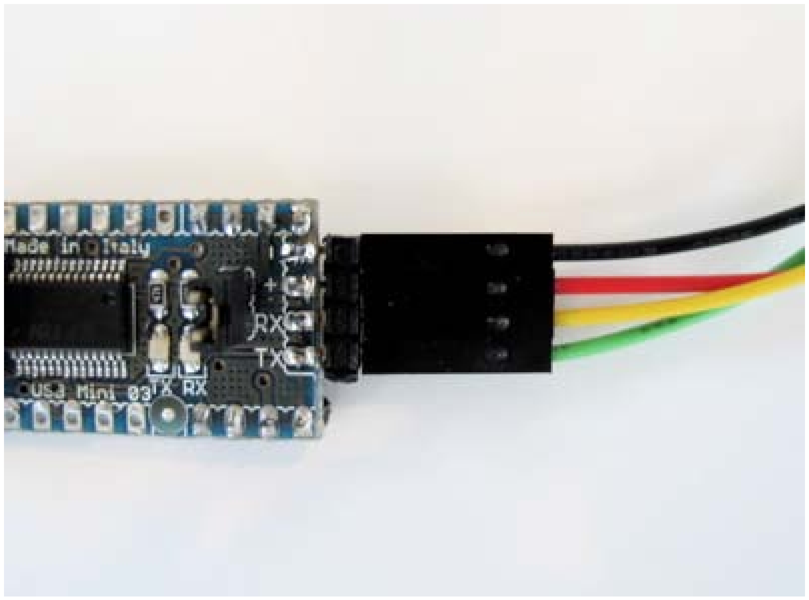


#### Modifying the Mini USB Adapter to Connect to the LilyPad Arduino

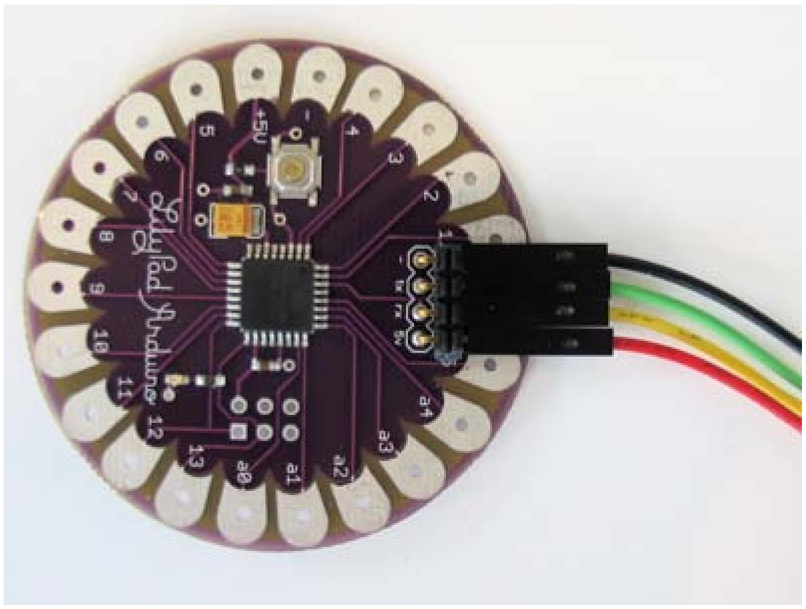
Solder a right angle male header to the Arduino mini USB adapter and then use female-female jumper cables to connect +,-,tx, and rx on the two boards. Right angle male headers are available here and female-female jumper cables are available here. On the version 3 Arduino mini USB adapter you want to connect tx to tx and rx to rx. We're using a red jumper for +, black for -, green for TX and yellow for RX.



Here is a close up view of the miniusb side of the connection:

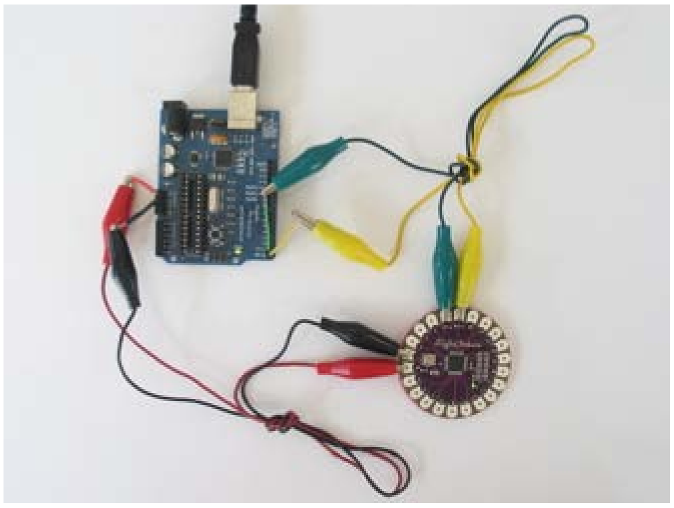


And a close up of the LilyPad side of the connection:



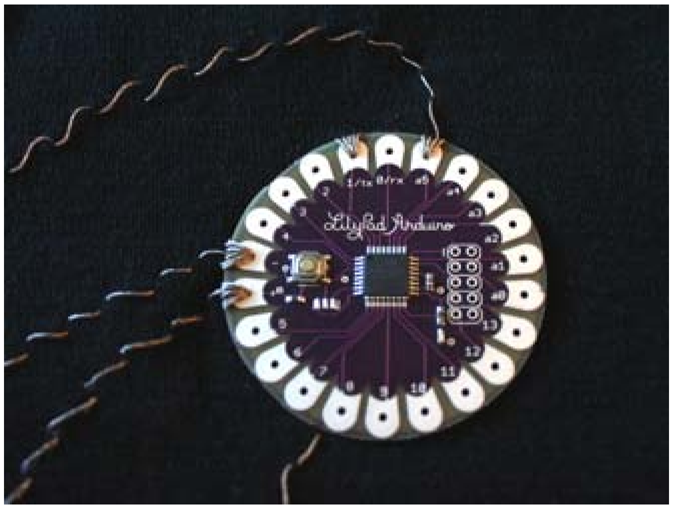
**Connecting the LilyPad Arduino and a regular Arduino**

You can also use an Arduino NG to connect the LilyPad Arduino to your computer, using a regular Arduino as a power supply and USB/Serial connection. Just remove the ATmega8 or ATmega168 from the regular Arduino and then use jumper wires and alligator clips to attach the TX, RX, +, and - tabs on the LilyPad to the corresponding pins on the NG. Here's a photo.



**Sewing the LilyPad Arduino**

The hole on each tab of the LilyPad is large enough for a sewing needle to pass through. You can make both electrical and physical connections with stitching in conductive thread. Sew through the holes several times to insure good contact. Here's a picture showing a sewn LilyPad:



See the LilyPad Arduino tutorial on Leah's website for more information about building a working wearable. See SparkFun for more stitchable modules that you can use with your LilyPad Arduino.

## Arduino Xbee Shield

The Arduino Xbee shield allows your Arduino board to communicate wirelessly using Zigbee. It was developed in collaboration with Libelium.

### A Simple Example

You should be able to get two Arduino boards with Xbee shields talking to each other without any configuration, using just the standard Arduino serial commands (described in the reference).

To upload a sketch to an Arduino board with a Xbee shield, you'll need to put both jumpers on the shield to the "USB" setting (i.e. place them on the two pins closest to the edge of the board) or remove them completely (but be sure not to lose them!). Then, you can upload a sketch normally from the Arduino environment. In this case, upload the **Communication | Physical Pixel** sketch to one of the boards. This sketch instructs the board to turn on the LED attached to pin 13 whenever it receives an 'H' over its serial connection, and turn the LED off when it gets an 'L'. You can test it by connecting to the board with the Arduino serial monitor (be sure it's set at 9600 baud), typing an H, and pressing enter (or clicking send). The LED should turn on. Send an L and the LED should turn off. If nothing happens, you may have an Arduino board that doesn't have a built-in LED on pin 13 (see the board index to check for sure), in this case you'll need to supply your own.

Once you've uploaded the Physical Pixel sketch and made sure that it's working, unplug the first Arduino board from the computer. Switch the jumpers to the Xbee setting (i.e. place each on the center pin and the pin farthest from the edge of the board). Now, you need to upload a sketch to the other board. Make sure its jumpers are in the USB setting. Then upload the following sketch to the board:

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.print('H');
  delay(1000);
  Serial.print('L');
  delay(1000);
}
```

When it's finished uploading, you can check that it's working with the Arduino serial monitor. You should see H's and L's arriving one a second. Turn off the serial monitor and unplug the board. Switch the jumpers to the Xbee setting. Now connect both boards to the computer. After a few seconds, you should see the LED on the first board turn on and off, once a second. (This is the LED on the Arduino board itself, not the one on the Xbee shield, which conveys information about the state of the Xbee module.) If so, congratulations, your Arduino boards are communicating wirelessly. This may not seem that exciting when both boards are connected to the same computer, but if you connect them to different computers (or power them with an external power supply - being sure to switch the power jumper on the Arduino board), they should still be able to communicate.

### A Few Notes

You can use any of the standard Arduino serial commands with the Xbee shield. With the shield's jumpers in the Xbee position, the print and println commands will send data over the Xbee shield and the USB connection (i.e. to other Xbee shields and to the computer at the same time). In this configuration, however, the board will only receive data from the Xbee shield not from the USB connection (you'll need to switch the jumpers to allow the board to receive data from the computer).

The Xbee module on the shield is set up to work at 9600 baud by default, so unless you reconfigure it, you'll need to make sure you're passing 9600 to the Serial.begin() command in your sketch.

To allow your computer to communicate directly with the Xbee shield, connect it to an Arduino board whose microcontroller has been removed and place its jumpers in the USB configuration. Then you can send data to and receive data from the Xbee module from any terminal program. This allows you, for example, to see the data that the module is receiving from other Xbee shields (e.g. to collect sensor data wirelessly from a number of locations).

## Configuring the Xbee Module

You can configure the Xbee module from code running on the Arduino board or from software on the computer. To configure it from the Arduino board, you'll need to have the jumpers in the Xbee position. To configure it from the computer, you'll need to have the jumpers in the USB configuration and have removed the microncontroller from your Arduino board.

To get the module into configuration mode, you need to send it three plus signs: +++ and there needs to be at least one second before and after during which you send no other character to the module. Note that this includes newlines or carriage return characters. Thus, if you're trying to configure the module from the computer, you need to make sure your terminal software is configured to send characters as you type them, without waiting for you to press enter. Otherwise, it will send the plus signs immediately followed by a newline (i.e. you won't get the needed one second delay after the +++). If you successfully enter configuration mode, the module will send back the two characters 'OK', followed by a carriage return.

| Send Command | Expected Response |
|---|---|
| +++ | OK <*CR*> |

Once in configuration mode, you can send AT commands to the module. Command strings have the form ATxx (where xx is the name of a setting). To read the current value of the setting, send the command string followed by a carriage return. To write a new value to the setting, send the command string, immediately followed by the new setting (with no spaces or newlines in-between), followed by a carriage return. For example, to read the network ID of the module (which determines which other Xbee modules it will communicate with), use the *'ATID* command:

| Send Command | Expected Response |
|---|---|
| ATID <*enter*> | 3332 <*CR*> |

To change the network ID of the module:

| Send Command | Expected Response |
|---|---|
| ATID3331 <*enter*> | OK <*CR*> |

Now, check that the setting has taken effect:

| Send Command | Expected Response |
|---|---|
| ATID <*enter*> | 3331 <*CR*> |

Unless you tell the module to write the changes to non-volatile (long-term) memory, they will only be in effect until the module loses power. To save the changes permanently (until you explicitly modify them again), use the **ATWR** command:

| Send Command | Expected Response |
|---|---|
| ATWR <*enter*> | OK <*CR*> |

To reset the module to the factory settings, use the **ATRE** command:

| Send Command | Expected Response |
|---|---|
| ATRE <*enter*> | OK <*CR*> |

Note that like the other commands, the reset will not be permanent unless you follow it with the **ATWR** comamand.

## References

For more information, see: the hardware page for the Xbee shield, the Libelium SquidBee wiki, and the MaxStream Xbee page.

# Arduino Troubleshooting

**On this page...** (hide)

### Why doesn't my sketch start when I'm powering the board with an external power supply?

Because the RX pin is unconnected, the bootloader on the board may be seeing garbage data coming in, meaning that it never times out and starts your sketch. Try tying the RX pin to ground with a 10K resistor (or connecting it to the TX pin).

### Why I can't upload my programs to the Arduino board?

There are a few things that could be wrong.

- First make sure your board is on (the green LED is on) and connected to the computer (if it's not, see "what if my board doesn't turn on" above).
- Then, check that the proper port is selected in the "Tools | Serial Port" menu (if your port doesn't appear, restart the IDE with the board connected to the computer).
- Make sure there's a bootloader burned on the Atmega8 on your Arduino board. To check, connect an LED to pin 13 and reset the board. The LED should blink. If it doesn't, see the Bootloader page for instructions on burning a bootloader to the board.
- Be sure that you are resetting the board a couple of seconds before uploading (unless you have an Arduino Diecimila).
- However, note that some Diecimila were accidently burned with the wrong bootloader and may require you to physically press the reset button before uploading; see this question below.
- However, on some computers, you may need to press the reset button on the board after you hit the upload button in the Arduino environment. Try different intervals of time between the two, up to 10 seconds or more.
- Disconnect digital pins 0 and 1 while uploading (they can connected and used after the code has been uploaded).
- Try uploading with nothing connected to the board (apart from the USB cable, of course).

- Make sure the board isn't touching anything metallic or conductive.
- If you get this error: `[VP 1] Device is not responding correctly.` try uploading again (i.e. reset the board and press the download button a second time).
- Check that you're not running any programs that scan all serial ports, like PDA sync applications, Bluetooth-USB drivers (e.g. BlueSoleil), virtual daemon tools, etc.
- Make sure you don't have firewall software that blocks access to the serial port (e.g. ZoneAlarm).
- You may need to quit Processing, PD, vvvv, etc. if you're using them to read data over the USB or serial connection to the Arduino board.
- If you have a really ancient Arduino board, you may need to change the baud rate at which sketches are uploaded to 9600 (from the normal 19200). You will have to change the speed in the *preferences* file directly. See the preferences page for instructions on finding the file. Look for the file in your computer and change the **serial.download_rate** property to match the one in your board. If you have such a board, it's recommended that you burn the latest bootloader (which works at 19200 baud). This can be done with the *'Tools | Burn Bootloader* menu item.

If it still doesn't work, you can ask for help in the forum. Please include the following information:

- Your operating system.
- What kind of board you have. If it's a Mini, LilyPad or other board that requires extra wiring, include a photo of your circuit, if possible.
- Whether or not you were ever able to upload to the board. If so, what were you doing with the board before / when it stopped working, and what software have you recently added or removed from your computer?
- The messages displayed when you try to upload with verbose output enabled. To do this, you'll need to set upload.verbose to true in your Arduino preferences file.

### Why does the Arduino software freeze when I try to upload a program? (on Windows)?

This might be caused by a conflict with the Logitech process 'LVPrcSrv.exe'. Open the Task Manager and see if this program is running, and if so, kill it before attempting the upload. more information

### What if my board doesn't turn on (the green power LED doesn't light up)?

If you're using a USB board, make sure that the jumper (little plastic piece near the USB plug) is on the correct pins. If you're powering the board with an external power supply (plugged into the power plug), the jumper should be on the two pins closest to the power plug. If you're powering the board through the USB, the jumper should be on the two pins closest to the USB plug. This picture shows the arrangment for powering the board from the USB port.

Attach:jumper.jpg Δ

(thanks to mrbbp for report and picture)

### Why does my Diecimila take such a long time (6-8 seconds) to start my sketch?

Some of the Arduino Diecimila boards were accidently burned with the Arduino NG bootloader. It should work fine, but has a longer delay when the board is reset (because the NG doesn't have an automatic reset, so you have to time the uploads manually). You can recognize the NG bootloader because the LED on pin 13 will blink three times when you reset the board (as compared to once with the Diecimila bootloader). If your Diecimila has the NG bootloader on it, you may need to physically press the reset button on the board before uploading your sketch. You can burn the correct bootloader onto your Diecimila, see the bootloader page for details.

### What should I do if I get an error when launching arduino.exe on Windows?

If you get an error when double-clicking the arduino.exe executable on Windows, for example:

```
Arduino has encountered a problem and needs to close.
```

you'll need to launch Arduino using the run.bat file. Please be patient, the Arduino environment may take some time to open.

### Why won't Arduino run on old versions of Mac OS X?

If you get an error like this:

```
Link (dyld) error:
```

```
dyld: /Applications/arduino-0004/Arduino 04.app/Contents/MacOS/Arduino Undefined symbols:
/Applications/arduino-0004/librxtxSerial.jnilib undefined reference to _printf$LDBL128 expected to be
defined in /usr/lib/libSystem.B.dylib
```

you probably need to upgrade to Max OS X 10.3.9 or later. Older versions have incompatible versions of some system libraries.

Thanks to Gabe462 for the report.

### What do I do if I get an UnsatisfiedLinkError error (about native library librxtxSerial.jnilib) when launching Arduino?

If you get an error like this when launching Arduino:

```
Uncaught exception in main method: java.lang.UnsatisfiedLinkError: Native Library
/Users/anu/Desktop/arduino-0002/librxtxSerial.jnilib already loaded in another classloader
```

you probably have an old version of the communications library lying around. Search for comm.jar or jcl.jar in /System/Library/Frameworks/JavaVM.framework/ or in directories in your CLASSPATH or PATH environment variables. (reported by Anurag Sehgal)

### What about the error "Could not find the main class."?

If you get this error when launching Arduino:

```
Java Virtual Machine Launcher: Could not find the main class. Program will exit.
```

make sure that you correctly extracted the contents of the Arduino .zip file - in particular that the **lib** directory is directly inside of the Arduino directory and contains the file **pde.jar**.

### What can I do about cygwin conflicts on Windows?

If you already have cygwin installed on your machine, you might get an error like this when you try to compile a sketch in Arduino:

```
6 [main] ? (3512) C:\Dev\arduino-0006\tools\avr\bin\avr-gcc.exe: *** fatal error - C:\Dev\arduino-
0006\tools\avr\bin\avr-gcc.exe: *** system shared memory version mismatch detected - 0x75BE0084/0x75BE009C.
```

```
This problem is probably due to using incompatible versions of the cygwin DLL.
```

```
Search for cygwin1.dll using the Windows Start->Find/Search facility and delete all but the most recent
version. The most recent version *should* reside in x:\cygwin\bin, where 'x' is the drive on which you
have installed the cygwin distribution. Rebooting is also suggested if you are unable to find another
cygwin DLL.
```

If so, first make sure that you don't have cygwin running when you use Arduino. If that doesn't help, you can try deleting cygwin1.dll from the Arduino directory and replacing it with the cygwin1.dll from your existing cygwin install (probably in c:\cygwin\bin).

*Thanks to karlcswanson for the suggestion.*

### Why does the Arduino software run really slowly (on Windows)?

If the Arduino software takes a long time to start up and appears to freeze when you try to open the Tools menu, there by a conflict with another device on your system. The Arduino software, on startup and when you open the Tools menu, tries to get a list of all the COM ports on your computer. It's possible that a COM port created by one of the devices on your computer slows down this process. Take a look in the Device Manager. Try disabling the devices that provide COM ports (e.g. Bluetooth devices).

### Why doesn't my board show in the Tools | Serial Port menu ?

If you're using a USB Arduino board, make sure you installed the FTDI drivers (see the Howto for directions). If you're using a USB-to-Serial adapter with a serial board, make sure you installed its drivers.

Make sure that the board is plugged in: the serial port menu refreshes whenever you open the **Tools** menu, so if you just unplugged the board, it won't be in the menu.

Check that you're not running any programs that scan all serial ports, like PDA sync applications, Bluetooth-USB drivers (e.g. BlueSoleil), virtual daemon tools, etc.

On Windows, the COM port assigned to the board may be too high. From zeveland:

"One little note if you aren't able to export and your USB board is trying to use a high COM port number: try changing the FTDI chip's COM port assignment to a lower one.

"I had a bunch of virtual COM ports set up for Bluetooth so the board was set to use COM17. The IDE wasn't able to find the board so I deleted the other virtual ports in Control Panel (on XP) and moved the FTDI's assignment down to COM2. Make sure to set Arduino to use the new port and good luck."

On the Mac, if you have an old version of the FTDI drivers, you may need to remove them and reinstall the latest version. See this forum thread for directions (thanks to gck).

### What if I get a gnu.io.PortInUseException when uploading code or using the serial monitor (on the Mac)?

```
Error inside Serial.<init>()
gnu.io.PortInUseException: Unknown Application
     at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:354)
     at processing.app.Serial.<init>(Serial.java:127)
     at processing.app.Serial.<init>(Serial.java:72)
```

This probably means that the port is actually in use by another application. Please make sure that you're not running other programs that access serial or USB ports, like PDA sync application, bluetooth device managers, certain firewalls, etc. Also, note that some programs (e.g. Max/MSP) keep the serial port open even when not using it - you may to need to close any patches that use the serial port or quit the application entirely.

If you get this error with Arduino 0004 or earlier, or with Processing, you'll need to run the macosx_setup.command, and then restart your computer. Arduino 0004 includes a modified version of this script that all users need to run (even those who ran the one that came with Arduino 0003). You may also need to delete the contents of the **/var/spool/uucp** directory.

### I'm having trouble with the FTDI USB drivers.

Try installing the latest drivers from FTDI or contacting their support at support1@ftdichip.com.

### Why doesn't my sketch start when I power up or reset the Arduino board?

Most likely because you are sending serial data to the board when it firsts turns on. During the first few seconds, the bootloader (a program pre-burned onto the chip on the board) listens for the computer to send it a new sketch to be uploaded to the board. After a few seconds without communication, the bootloader will time out and start the sketch that's already on the board. If you continue to send data to the bootloader, it will never time out and your sketch will never start. You'll either need to find a way to stop serial data from arriving for the first few seconds when the board powers (e.g. by enabling the chip that sends the data from within your setup() function) or burn your sketch onto the board with an external programmer, replacing the bootloader.

### Why does my sketch appear to upload successfully but not do anything?

You have selected the wrong item from the Tools > Microcontroller menu. Make sure the selected microcontroller corresponds to the one on your board (either ATmega8 or ATmega168) - the name will be written on the largest chip on the board.

Check for a noisy power supply. It's possible this could cause the chip to lose its sketch.

Alternatively, the sketch may be too big for the board. When uploading your sketch, Arduino 0004 checks if it's too big for the ATmega8, but it bases its calculation on a 1 Kb bootloader. You may have a older bootloader that takes up 2 Kb of the 8 Kb of program space (flash) on the ATmega8 instead of the 1 Kb used by the current bootloader. If yours is bigger, only part of the sketch will be uploaded, but the software won't know, and your board will continually reset, pause, reset.

If you have access to an AVR-ISP or parallel port programmer, you can burn the latest version of the bootloader to your board with the **Tools | Burn Bootloader** menu item. Otherwise, you can tell the Arduino environment the amount of space available for sketches by editing the upload.maximum_size variable in your preferences file (see: instructions on finding the file). Change 7168 to 6144, and the environment should correctly warn you when your sketch is too big.

### How can I reduce the size of my sketch?

The ATmega168 chip on the Arduino board is cheap, but it has only 16 Kb of program code, which isn't very much (and 2 Kb is used by the bootloader).

If you're using floating point, try to rewrite your code with integer math, which should save you about 2 Kb. Delete any **#include** statements at the top of your sketch for libraries that you're not using.

Otherwise, see if you can make your program shorter.

We're always working to reduce the size of the Arduino core to leave more room for your sketches.

### Why don't I get a PWM (an analog output) when I call analogWrite() on pins other than 3, 5, 6, 9, 10, or 11?

The microcontroller on the Arduino board (the ATmega168) only supports PWM/analogWrite() on certain pins. Calling analogWrite() on any other pins will give high (5 volts) for values greater than 128 and low (0 volts) for values less than 128. (Older Arduino boards with an ATmega8 only support PWM output on pins 9, 10, and 11.)

### Why do I get errors about undeclared functions or undeclared types?

The Arduino environment attempts to automatically generate prototypes for your functions, so that you can order them as you like in your sketch. This process, however, isn't perfect, and sometimes leads to obscure error messages.

If you declare a custom type in your code and create a function that accepts or returns a value of that type, you'll get an error when you try to compile the sketch. This is because the automatically-generated prototype for that function will appear above the type definition.

If you declare a function with a two-word return type (e.g. "unsigned int") the environment will not realize it's a function and will not create a prototype for it. That means you need to provide your own, or place the definition of the function above any calls to it.

Guide Home

# Introduction to the Arduino Environment

## Toolbar

*Verify/Compile*

Checks your code for errors.

*Stop*

Stops the serial monitor, or unhighlight other buttons.

*New*

Creates a new sketch.

*Open*

Presents a menu of all the sketches in your sketchbook. Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the **File | Sketchbook** menu instead.

*Save*

Saves your sketch.

*Upload to I/O Board*

Uploads your code to the Arduino I/O board. Make sure to save or verify your sketch before uploading it.

*Serial Monitor*

Displays serial data being sent from the Arduino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to **Serial.begin** in your sketch. Note that on Mac or Linux, the Arduino board will reset (rerun your sketch from the

## Menus

### Sketch

*Verify/Compile*

Checks your sketch for errors.

*Import Library*

Uses a library in your sketch. Works by adding **#include**s to the top of your code. This makes extra functionality available to your sketch, but increases its size. To stop using a library, delete the appropriate **#include**s from the top of your sketch. For more details, see the page on Libraries.

*Show Sketch Folder*

Opens the sketch folder on the desktop.

*Add File...*

Adds another source file to the sketch. The new file appears in a new tab in the sketch window. This facilitates and larger projects with multiple source files. Files can be removed from a sketch using the tab menu.

### Tools

*Auto Format*

This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements instead curly braces are indented more.

*Copy for Discourse*

Copies the code of your sketch to the clipboard in a forum suitable for posting to the forum, complete with syntax coloring.

*Board*

Select the board that you're using. This controls the way that your sketch is compiled and uploaded as well as the behavior of the Burn Bootloader menu items.

*Serial Port*

This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

Before uploading your sketch, you need to select the item beginning) when you connect with the serial monitor.

You can also talk to the board from Processing, Flash, MaxMSP, etc (see the interfacing page for details).

**Tab Menu**

⇨

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no extension), C files (.c extension), C++ files (.cpp), or header files (.h). See the description of the build process for details of how these are handled.

from this menu that represents your Arduino board. On the Mac, this is probably something like **/dev/tty.usbserial-1B1** (for a USB board), or **/dev/tty.USA19QW1b1P1.1** (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably **COM1** or **COM2** (for a serial board) or **COM4**, **COM5**, **COM7**, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager.

*Burn Bootloader*

The items in this menu allow you to burn a bootloader onto your board with a variety of programmers. This is not required for normal use of an Arduino board, but may be useful if you purchase additional ATmega's or are building a board yourself. Ensure that you've selected the correct board from the Boards menu beforehand. To burn a bootloader with the AVR ISP, you need to select the item corresponding to your programmer from the Serial Port menu. Instructions are available for building a parallel programmer.

## Preferences

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preference files.

## Login to Arduino

| | |
|---|---|
| Username: | |
| Password: | |
| Keep me logged in: | ✓ |

Login

## Guide.Guide History

Hide minor edits - Show changes to markup

October 22, 2006, at 12:19 PM by David A. Mellis -
Deleted lines 0-30:

## Guide to Arduino

This guide tell you everything you need to get started with Arduino. It consists of the following pages:

- **Introduction**: this page, which explains what it is and why you'd want to use it
- How To: step-by-step instructions on getting your first Arduino program working
- Board: description of the Arduino board and its parts
- Environment: description of the Arduino development environment and its parts
- References: pointers to other sources of information, on this site and elsewhere, for learning more

### What is Arduino?

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

### Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino

- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Restore
October 22, 2006, at 12:18 PM by David A. Mellis -
Changed lines 3-4 from:

This guide tell you everything you need to get started with Arduino. It explains what it is and why you'd want to use it. Then, it offers step-by-step instructions on getting your first Arduino program working, followed by more details on the Arduino hardware and software. Finally, it points to other sources of information, on this site and elsewhere, for learning more.

to:

This guide tell you everything you need to get started with Arduino. It consists of the following pages:

- **Introduction**: this page, which explains what it is and why you'd want to use it
- How To: step-by-step instructions on getting your first Arduino program working
- Board: description of the Arduino board and its parts
- Environment: description of the Arduino development environment and its parts
- References: pointers to other sources of information, on this site and elsewhere, for learning more

Restore
October 22, 2006, at 12:02 PM by David A. Mellis -
Changed lines 7-12 from:

Arduino is an open-source physical computing platform based on a simple i/o board, and a development environment for writing Arduino software. The Arduino programming language is an implementation of Wiring, itself built on Processing.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand? or purchased? preassembled; the open-source IDE can be downloaded? for free.

Arduino received an Honory Mention in the Digital Communities section of the 2006 Ars Electronica Prix. Credits?

to:

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

Restore
October 22, 2006, at 12:01 PM by David A. Mellis - tom's what is arduino and why would you want to use it
Added lines 1-25:

## Guide to Arduino

This guide tell you everything you need to get started with Arduino. It explains what it is and why you'd want to use it. Then, it offers step-by-step instructions on getting your first Arduino program working, followed by more details on the Arduino hardware and software. Finally, it points to other sources of information, on this site and elsewhere, for learning more.

### What is Arduino?

Arduino is an open-source physical computing platform based on a simple i/o board, and a development environment for writing Arduino software. The Arduino programming language is an implementation of Wiring, itself built on Processing.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand? or purchased? preassembled; the open-source IDE can be downloaded? for free.

Arduino received an Honory Mention in the Digital Communities section of the 2006 Ars Electronica Prix. Credits?

## Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino
- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Restore

# Arduino : Guide / Guide

# Arduino

search

## Login to Arduino

|  |  |
|---|---|
| Username: |  |
| Password: |  |
| Keep me logged in: | ✓ |

Login

## Guide.HomePage History

Hide minor edits - Show changes to markup

June 18, 2008, at 09:37 AM by David A. Mellis -
Added line 23:

- Arduino Nano

Restore
April 23, 2008, at 10:28 PM by David A. Mellis -
Changed lines 6-7 from:

(:table width=90% border=0 cellpadding=5 cellspacing=0:)

to:

(:table width=100% border=0 cellpadding=5 cellspacing=0:)

Restore
February 13, 2008, at 09:20 PM by David A. Mellis -
Deleted lines 29-30:

- Hardware board: A more detailed description of the Arduino hardware board and its parts.

Restore
November 08, 2007, at 01:05 PM by Limor Fried -
Changed lines 4-5 from:

You just got your Arduino in the mail and now you're ready to start having fun with Physical Computing? Here is the best place to start, with tutorials and guides to help you get started with Arduino! From how to turn it on and plug it in to installing the driver and uploading your very first sketch program. These getting started guides are required reading for all new Arduino users!

to:

You just got your Arduino in the mail and now you're ready to start having fun with Physical Computing? Here is the best place to start, with tutorials and guides to help you get started with Arduino hardware and software! From how to turn it on and plug it in to installing the driver and uploading your very first sketch program. These getting started guides are required reading for all new users!

Restore
November 08, 2007, at 01:01 PM by Limor Fried - font noodling
Added lines 4-5:

You just got your Arduino in the mail and now you're ready to start having fun with Physical Computing? Here is the best place to start, with tutorials and guides to help you get started with Arduino! From how to turn it on and plug it in to installing the driver and uploading your very first sketch program. These getting started guides are required reading for all new Arduino users!

Added line 7:
Added line 10:
Changed line 17 from:

- How to get set up using the following operating systems

to:

- **Step-by-step instructions for how to get set up with your Arduino hardware and software. This is the most important part of the getting started guide!** Click whichever link matches your computer set-up:

Restore
November 08, 2007, at 12:52 PM by Limor Fried -
Changed lines 38-39 from:

### More Getting Started guides, from our friends

to:

### More detailed guides, from our friends

Changed lines 42-43 from:

- Course guides: Longer documents introducing Arduino: class 1 (getting started), class 2 (input and sensors), class 3 (communication, servos, and pwm), class 4 (piezo sound & sensors, arduino+processing, stand-alone operation).

to:

- TodBot's course guides Longer presentation-format documents introducing Arduino from a Halloween hacking class taught by TodBot: class 1 (getting started), class 2 (input and sensors), class 3 (communication, servos, and pwm), class 4 (piezo sound & sensors, arduino+processing, stand-alone operation).

Restore
November 08, 2007, at 12:49 PM by Limor Fried - rearranging, reformatting, etc
Changed lines 11-13 from:

- Introduction: an explanation of what Arduino is and why you'd want to use it.
- How To get set up using the following operating systems

to:

- Introduction: Read this introduction for an explanation of what Arduino is and why you'd want to use it.
- How to get set up using the following operating systems

Changed lines 26-33 from:

- Board: A more detailed description of the Arduino hardware board and its parts.
- Environment: A more detailed description of the Arduino development environment and its parts.

### Elsewhere on the Arduino website

- Arduino booklet: an introduction to physical computing, electronics, and Arduino.

to:

- Hardware board: A more detailed description of the Arduino hardware board and its parts.
- Sofware environment: A more detailed description of the Arduino software (development environment) and its parts.

Changed lines 33-34 from:

### External Resources

to:

### Other guides on the Arduino website

- Arduino booklet: an fun introduction to physical computing, electronics, and Arduino. Printable PDF with hand-drawn illustrations.

### More Getting Started guides, from our friends

Restore
November 08, 2007, at 12:43 PM by Limor Fried - reformatting
Changed lines 1-3 from:

(:title Guide:)

## Guide to Arduino

to:

(:title Getting started with Arduino:)

# Guide to Getting Started with Arduino

Changed lines 7-10 from:

### In This Guide

This guide tell you everything you need to get started with Arduino. It consists of the following pages:

to:

### In This Quick-Start Guide...

This guide tell you everything you need to get started with Arduino and how to configure your setup so you can upload your first sketch. It consists of the following pages:

Changed lines 12-16 from:

- How To (Windows, Mac OS X, Linux; Arduino Mini, Arduino BT, LilyPad Arduino; Xbee shield): step-by-step instructions on getting your first Arduino program working.
- Troubleshooting: what to do if things don't work.
- Board: description of the Arduino board and its parts.
- Environment: description of the Arduino development environment and its parts.

to:

- How To get set up using the following operating systems
  - Windows
  - Mac OS X
  - Linux

- If you're using one of the following Arduino variants or shields, they have their own introductory guides as well:
  - Arduino Mini
  - Arduino BT
  - LilyPad Arduino
  - Xbee shield

- Troubleshooting: HELP!!! Having problems? Read the troubleshooting guide for advice on what to do if things don't work.

- Board: A more detailed description of the Arduino hardware board and its parts.

- Environment: A more detailed description of the Arduino development environment and its parts.

Restore
November 01, 2007, at 11:37 PM by David A. Mellis - adding ladyada's arduino tutorial.
Added lines 25-26:

- Learn electronics using Arduino: a great multi-part tutorial with tons of pictures, code, circuits, even video. Highly recommended.

Restore
October 04, 2007, at 08:53 AM by David A. Mellis - adding link to lilypad guide
Changed line 12 from:

- How To (Windows, Mac OS X, Linux; Arduino Mini, Arduino BT; Xbee shield): step-by-step instructions on getting your first Arduino program working.

to:

- How To (Windows, Mac OS X, Linux; Arduino Mini, Arduino BT, LilyPad Arduino; Xbee shield): step-by-step instructions on getting your first Arduino program working.

Restore
August 31, 2007, at 11:04 PM by David A. Mellis -
Changed line 11 from:

- Introduction (this page): an explanation of what Arduino is and why you'd want to use it.

to:

Introduction: an explanation of what Arduino is and why you'd want to use it.

Changed lines 33-59 from:

(:tableend:)

### What is Arduino?

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

### Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino
- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

### How do I use Arduino?

To get started, follow the instructions for your operating system: Windows, Mac OS X or Linux; or the additional instructions for your board: Arduino Mini, Arduino BT, or shield: Xbee.

to:

(:tableend:)

Restore
August 31, 2007, at 11:00 PM by David A. Mellis - (removing references which shouldn't have been there anyway)
Changed lines 16-17 from:

- References: links to and descriptions of other source of information on Arduino (on this site and elsewhere).

to:
Restore
August 12, 2007, at 07:19 PM by David A. Mellis -
Changed line 60 from:

To get started, follow the instructions for your operating system: Windows, Mac OS X or Linux; or the additional instructions for your board: Arduino Mini, Arduino BT, or shield: Xbee shield.

to:

To get started, follow the instructions for your operating system: Windows, Mac OS X or Linux; or the additional instructions for your board: Arduino Mini, Arduino BT, or shield: Xbee.

Restore
August 12, 2007, at 07:18 PM by David A. Mellis -
Changed line 12 from:

- How To (Windows, Mac OS X, Linux; Arduino Mini, Arduino BT): step-by-step instructions on getting your first Arduino program working.

to:

- How To (Windows, Mac OS X, Linux; Arduino Mini, Arduino BT; Xbee shield): step-by-step instructions on getting your first Arduino program working.

Changed line 60 from:

To get started, follow the instructions for your operating system: Windows, Mac OS X or Linux; or the additional instructions for your board: Arduino Mini, Arduino BT.

to:

To get started, follow the instructions for your operating system: Windows, Mac OS X or Linux; or the additional instructions for your board: Arduino Mini, Arduino BT, or shield: Xbee shield.

Restore
June 15, 2007, at 05:33 PM by David A. Mellis - removing broken link to videos
Deleted lines 25-26:

- Video lectures: Tom Igoe introduces Arduino. (thanks to Pollie Barden).

Restore
January 27, 2007, at 08:23 AM by David A. Mellis -
Changed line 12 from:

- How To (Windows, Mac OS X, Linux; Arduino Mini): step-by-step instructions on getting your first Arduino program working.

to:

- How To (Windows, Mac OS X, Linux; Arduino Mini, Arduino BT): step-by-step instructions on getting your first Arduino program working.

Changed line 62 from:

To get started, follow the instructions for your operating system: Windows, Mac OS X or Linux.

to:

To get started, follow the instructions for your operating system: Windows, Mac OS X or Linux; or the additional instructions for your board: Arduino Mini, Arduino BT.

Restore
January 07, 2007, at 07:25 AM by David A. Mellis -
Changed line 12 from:

- How To (Windows, Mac OS X, Linux): step-by-step instructions on getting your first Arduino program working.

to:

- How To (Windows, Mac OS X, Linux; Arduino Mini): step-by-step instructions on getting your first Arduino program working.

Restore
December 25, 2006, at 01:47 PM by David A. Mellis -
Changed line 11 from:

- Introduction (this page): an explaination of what Arduino is and why you'd want to use it.

to:

- Introduction (this page): an explanation of what Arduino is and why you'd want to use it.

Restore
December 04, 2006, at 04:39 PM by David A. Mellis - adding links to the howtos.
Changed lines 58-62 from:

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

to:

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

### How do I use Arduino?

To get started, follow the instructions for your operating system: Windows, Mac OS X or Linux.

Restore
December 02, 2006, at 10:43 AM by David A. Mellis -
Added line 1:

(:title Guide:)

Restore
November 18, 2006, at 02:10 AM by David A. Mellis - fixing link to Arduino booklet
Changed lines 19-20 from:

- Arduino booklet?: an introduction to physical computing, electronics, and Arduino.

to:

- Arduino booklet: an introduction to physical computing, electronics, and Arduino.

Restore
November 18, 2006, at 02:09 AM by David A. Mellis - adding link to Arduino booklet
Added lines 17-20:

### Elsewhere on the Arduino website

- Arduino booklet?: an introduction to physical computing, electronics, and Arduino.

Restore
November 04, 2006, at 01:16 PM by David A. Mellis -
Added lines 25-30:

- Wiring electronics reference: circuit diagrams for connecting a variety of basic electronic components.
- Schematics to circuits: from Wiring, a guide to transforming circuit diagrams into physical circuits.
- Tom Igoe's Physical Computing Site: lots of information on electronics, microcontrollers, sensors, actuators, books, etc.

Restore
November 04, 2006, at 01:14 PM by David A. Mellis -
Added lines 3-7:

(:table width=90% border=0 cellpadding=5 cellspacing=0:) (:cell width=50%:)

### In This Guide

Added lines 17-26:

(:cell width=50%:)

### External Resources

- Video lectures: Tom Igoe introduces Arduino. (thanks to Pollie Barden).

- Course guides: Longer documents introducing Arduino: class 1 (getting started), class 2 (input and sensors), class 3 (communication, servos, and pwm), class 4 (piezo sound & sensors, arduino+processing, stand-alone operation).

(:tableend:)

Restore
November 04, 2006, at 12:19 PM by David A. Mellis -
Deleted lines 2-3:

### In This Guide

Restore
November 04, 2006, at 12:19 PM by David A. Mellis -
Changed lines 12-13 from:
to:

- References: links to and descriptions of other source of information on Arduino (on this site and elsewhere).

Restore
November 04, 2006, at 12:18 PM by David A. Mellis -
Changed line 7 from:

- Introduction: an explains of what Arduino is and why you'd want to use it.

to:

- Introduction (this page): an explaination of what Arduino is and why you'd want to use it.

Changed lines 13-42 from:

### Other Parts of the Arduino Website

- Tutorials: code examples and circuits for performing many tasks.
- Reference: documentation of the Arduino programming language and functions.
- Hardware: descriptions of the various Arduino boards and other hardware, with schematics, PCB layout files, assembly instructions, etc.
- Language comparison: compares the Arduino/Wiring language (based on C/C++) with Processing (based on Java).
- FAQ: frequently asked questions about Arduino.
- Discussion forums: for questions about anything Arduino related.
- Playground: a publicly editable wiki collecting community documentation (register here).

### External Resources on Arduino

- Video lectures: Tom Igoe introduces Arduino. (thanks to Pollie Barden).
- Course guides: Longer documents introducing Arduino: class 1 (getting started), class 2 (input and sensors), class 3 (communication, servos, and pwm), class 4 (piezo sound & sensors, arduino+processing, stand-alone operation).

### Related Sites

- Instant Soup is an introduction to electronics through a series of beautifully-documented fun projects.
- Make magazine has some great links in its electronics archive.
- hack a day has links to interesting hacks and how-to articles on various topics.

to:

### What is Arduino?

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

### Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino

- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Restore
November 04, 2006, at 12:16 PM by David A. Mellis -
Added lines 19-20:

- Hardware: descriptions of the various Arduino boards and other hardware, with schematics, PCB layout files, assembly instructions, etc.

Restore
November 04, 2006, at 12:14 PM by David A. Mellis -
Deleted lines 2-3:

(:table width=90% border=0 cellpadding=5 cellspacing=0:) (:cell width=50%:)

Deleted lines 12-13:

(:cell width=50%:)

Deleted lines 26-27:

(:cellnr width=50%:)

Deleted lines 32-33:

(:cell width=50%:)

Deleted lines 40-41:

(:tableend:)

Restore
November 04, 2006, at 12:11 PM by David A. Mellis -
Changed lines 9-15 from:

- Introduction: this page, which explains what it is and why you'd want to use it
- How To (Windows, Mac OS X, Linux): step-by-step instructions on getting your first Arduino program working
- Troubleshooting: what to do if things don't work
- Board: description of the Arduino board and its parts
- Environment: description of the Arduino development environment and its parts

- References: pointers to other sources of information, on this site and elsewhere, for learning more

to:

- Introduction: an explains of what Arduino is and why you'd want to use it.
- How To (Windows, Mac OS X, Linux): step-by-step instructions on getting your first Arduino program working.
- Troubleshooting: what to do if things don't work.
- Board: description of the Arduino board and its parts.
- Environment: description of the Arduino development environment and its parts.

Changed lines 50-72 from:

(:tableend:)

### What is Arduino?

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

### Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino

- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

to:

(:tableend:)

Restore
November 04, 2006, at 12:09 PM by David A. Mellis -
Added lines 32-33:

(:cellnr width=50%:)

Added lines 40-41:

(:cell width=50%:)

Restore
November 04, 2006, at 12:08 PM by David A. Mellis -
Changed lines 18-19 from:

### Other Resources

to:

### Other Parts of the Arduino Website

Added lines 26-33:

- FAQ: frequently asked questions about Arduino.
- Discussion forums: for questions about anything Arduino related.
- Playground: a publicly editable wiki collecting community documentation (register here).

### External Resources on Arduino

Deleted line 35:
Added lines 38-46:

### Related Sites

- Instant Soup is an introduction to electronics through a series of beautifully-documented fun projects.
- Make magazine has some great links in its electronics archive.
- hack a day has links to interesting hacks and how-to articles on various topics.

Restore
November 04, 2006, at 12:03 PM by David A. Mellis - adding external resources
Added lines 3-6:

(:table width=90% border=0 cellpadding=5 cellspacing=0:) (:cell width=50%:)

### In This Guide

Added lines 16-32:

(:cell width=50%:)

### Other Resources

- Tutorials: code examples and circuits for performing many tasks.
- Reference: documentation of the Arduino programming language and functions.
- Language comparison: compares the Arduino/Wiring language (based on C/C++) with Processing (based on Java).
- Video lectures: Tom Igoe introduces Arduino. (thanks to Pollie Barden).
- Course guides: Longer documents introducing Arduino: class 1 (getting started), class 2 (input and sensors), class 3 (communication, servos, and pwm), class 4 (piezo sound & sensors, arduino+processing, stand-alone operation).

(:tableend:)

Restore
November 04, 2006, at 10:01 AM by David A. Mellis -
Changed line 6 from:

- How To (Windows, Mac OS X): step-by-step instructions on getting your first Arduino program working

to:

- How To (Windows, Mac OS X, Linux): step-by-step instructions on getting your first Arduino program working

Restore
November 04, 2006, at 08:05 AM by David A. Mellis -
Added line 7:

- Troubleshooting: what to do if things don't work

Restore
November 04, 2006, at 06:56 AM by David A. Mellis -
Changed line 6 from:

- How To: step-by-step instructions on getting your first Arduino program working

to:

- How To (Windows, Mac OS X): step-by-step instructions on getting your first Arduino program working

Restore
October 22, 2006, at 12:19 PM by David A. Mellis -
Added lines 1-31:

## Guide to Arduino

This guide tell you everything you need to get started with Arduino. It consists of the following pages:

- Introduction: this page, which explains what it is and why you'd want to use it
- How To: step-by-step instructions on getting your first Arduino program working
- Board: description of the Arduino board and its parts
- Environment: description of the Arduino development environment and its parts
- References: pointers to other sources of information, on this site and elsewhere, for learning more

### What is Arduino?

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

### Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino

- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Restore

# Guide to Getting Started with Arduino

You just got your Arduino in the mail and now you're ready to start having fun with Physical Computing? Here is the best place to start, with tutorials and guides to help you get started with Arduino hardware and software! From how to turn it on and plug it in to installing the driver and uploading your very first sketch program. These getting started guides are required reading for all new users!

## In This Quick-Start Guide...

This guide tell you everything you need to get started with Arduino and how to configure your setup so you can upload your first sketch. It consists of the following pages:

- Introduction: Read this introduction for an explanation of what Arduino is and why you'd want to use it.

- **Step-by-step instructions for how to get set up with your Arduino hardware and software. This is the most important part of the getting started guide!** Click whichever link matches your computer set-up:
  - Windows
  - Mac OS X
  - Linux

- If you're using one of the following Arduino variants or shields, they have their own introductory guides as well:
  - Arduino Nano
  - Arduino Mini
  - Arduino BT
  - LilyPad Arduino
  - Xbee shield

- Troubleshooting: HELP!!! Having problems? Read the troubleshooting guide for advice on what to do if things don't work.

- Sofware environment: A more detailed description of the Arduino software (development environment) and its parts.

## Other guides on the Arduino website

- Arduino booklet: an fun introduction to physical computing, electronics, and Arduino. Printable PDF with hand-drawn illustrations.

## More detailed guides, from our friends

- Learn electronics using Arduino: a great multi-part tutorial with tons of pictures, code, circuits, even video. Highly recommended.

- TodBot's course guides Longer presentation-format documents introducing Arduino from a Halloween hacking class taught by TodBot: class 1 (getting started), class 2 (input and sensors), class 3 (communication, servos, and pwm), class 4 (piezo sound & sensors, arduino+processing, stand-alone operation).

- Wiring electronics reference: circuit diagrams for connecting a variety of basic electronic components.

- Schematics to circuits: from Wiring, a guide to transforming circuit diagrams into physical circuits.

- Tom Igoe's Physical Computing Site: lots of information on electronics, microcontrollers, sensors, actuators, books, etc.

# Arduino

search

## Login to Arduino

Username:

Password:

Keep me logged in: ✓

Login

## Guide.Introduction History

Hide minor edits - Show changes to markup

August 31, 2007, at 11:01 PM by David A. Mellis -
Changed lines 21-25 from:

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

to:

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

### How do I use Arduino?

To get started, follow the instructions for your operating system: Windows, Mac OS X or Linux; or the additional instructions for your board: Arduino Mini, Arduino BT, or shield: Xbee.

Restore
November 04, 2006, at 12:11 PM by David A. Mellis -
Added lines 1-21:

### What is Arduino?

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

### Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the

Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino

- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

## What is Arduino?

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

## Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino

- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

## How do I use Arduino?

To get started, follow the instructions for your operating system: Windows, Mac OS X or Linux; or the additional instructions for your board: Arduino Mini, Arduino BT, or shield: Xbee.

# Arduino

search

## Login to Arduino

Username:

Password:

Keep me logged in: ✓

Login

## Guide.Windows History

Hide minor edits - Show changes to markup

June 01, 2008, at 06:27 PM by David A. Mellis -
Added lines 3-4:

(:include HowtoIntro:)

Restore
June 01, 2008, at 06:19 PM by David A. Mellis -
Changed lines 7-11 from:

### 2 | Download the Arduino environment

To program the Arduino board you need the Arduino environment.

**Download Arduino**: Arduino 0011 for Windows

to:

(:include HowtoDownload:)

Restore
March 28, 2008, at 05:53 PM by David A. Mellis -
Changed lines 10-11 from:

**Download Arduino**: Arduino 0010 for Windows

to:

**Download Arduino**: Arduino 0011 for Windows

Restore
March 27, 2008, at 05:37 PM by David A. Mellis -
Changed lines 10-11 from:

**Download Arduino**: Arduino 0009 for Windows

to:

**Download Arduino**: Arduino 0010 for Windows

Restore
August 06, 2007, at 08:44 PM by David A. Mellis - updating software version to 0009.
Changed lines 10-11 from:

**Download Arduino**: Arduino 0008 for Windows

to:

**Download Arduino**: Arduino 0009 for Windows

Restore
June 09, 2007, at 06:54 PM by David A. Mellis -
Changed lines 10-11 from:

**Download Arduino**: Arduino 0007 for Windows

to:

**Download Arduino**: Arduino 0008 for Windows

Restore
March 03, 2007, at 02:11 PM by David A. Mellis -
Added lines 37-38:

(:include HowtoLED:)

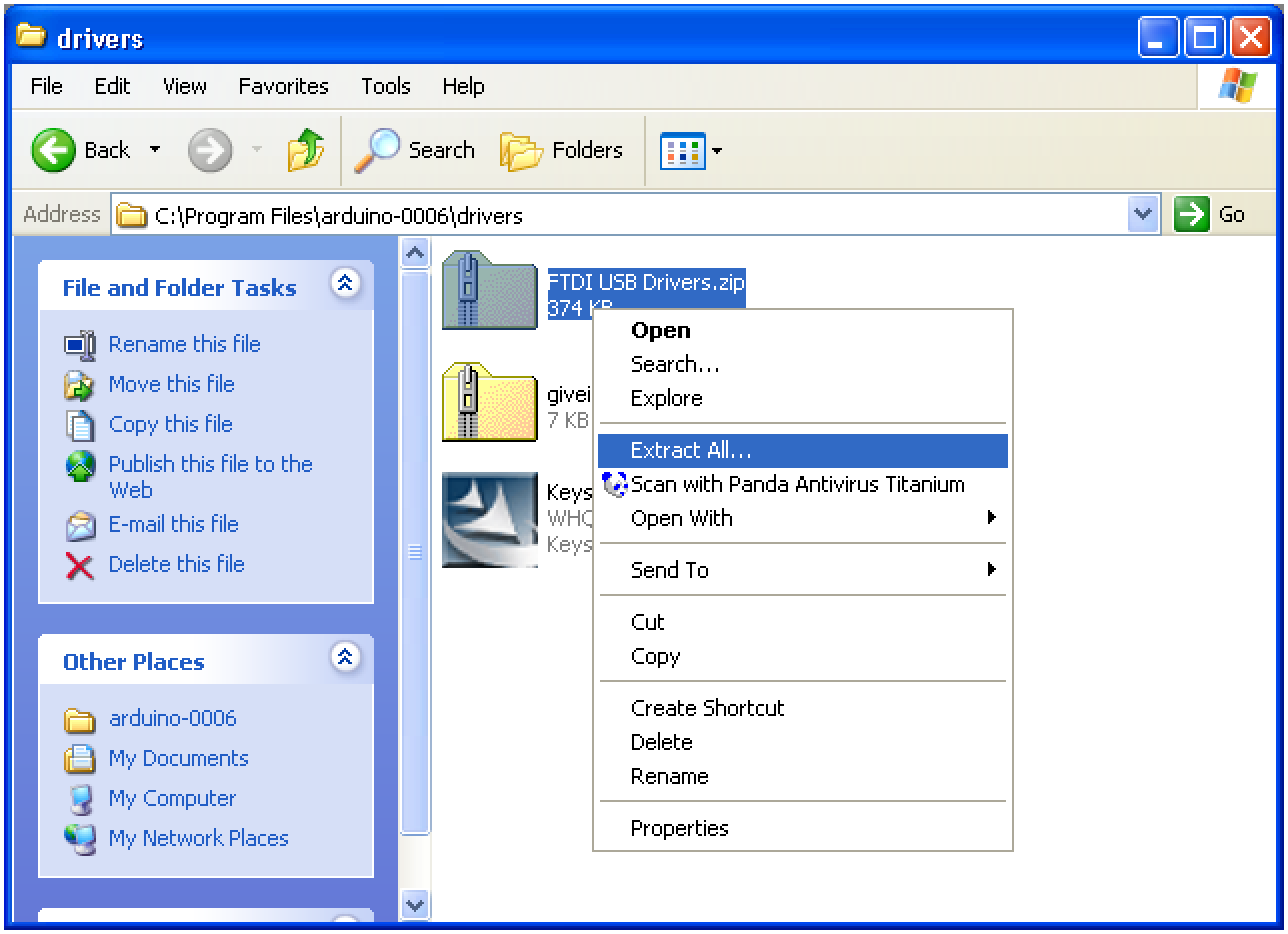
Restore
December 25, 2006, at 06:27 PM by David A. Mellis - removing directions for unzipping Windows drivers.
Changed lines 10-18 from:

**Download Arduino**: Arduino 0006 for Windows

### 3 | Unzip the USB drivers

If you are using a USB Arduino, you will need to install the drivers for the FTDI chip on the board. These can be found in the drivers directory of the Arduino distribution.

On Windows, you will need to unzip `FTDI USB Drivers.zip`. In the next step ("Connect the board"), you will point Window's Add New Hardware wizard to these drivers.



to:

**Download Arduino**: Arduino 0007 for Windows

### 3 | Locate the USB drivers

If you are using a USB Arduino, you will need to install the drivers for the FTDI chip on the board. These can be found in the `drivers/FTDI USB Drivers` directory of the Arduino distribution. In the next step ("Connect the board"), you will point Window's Add New Hardware wizard to these drivers.

Restore
December 04, 2006, at 04:37 PM by David A. Mellis -
Changed lines 7-8 from:

(:include HowtoDownload:)

to:

## 2 | Download the Arduino environment

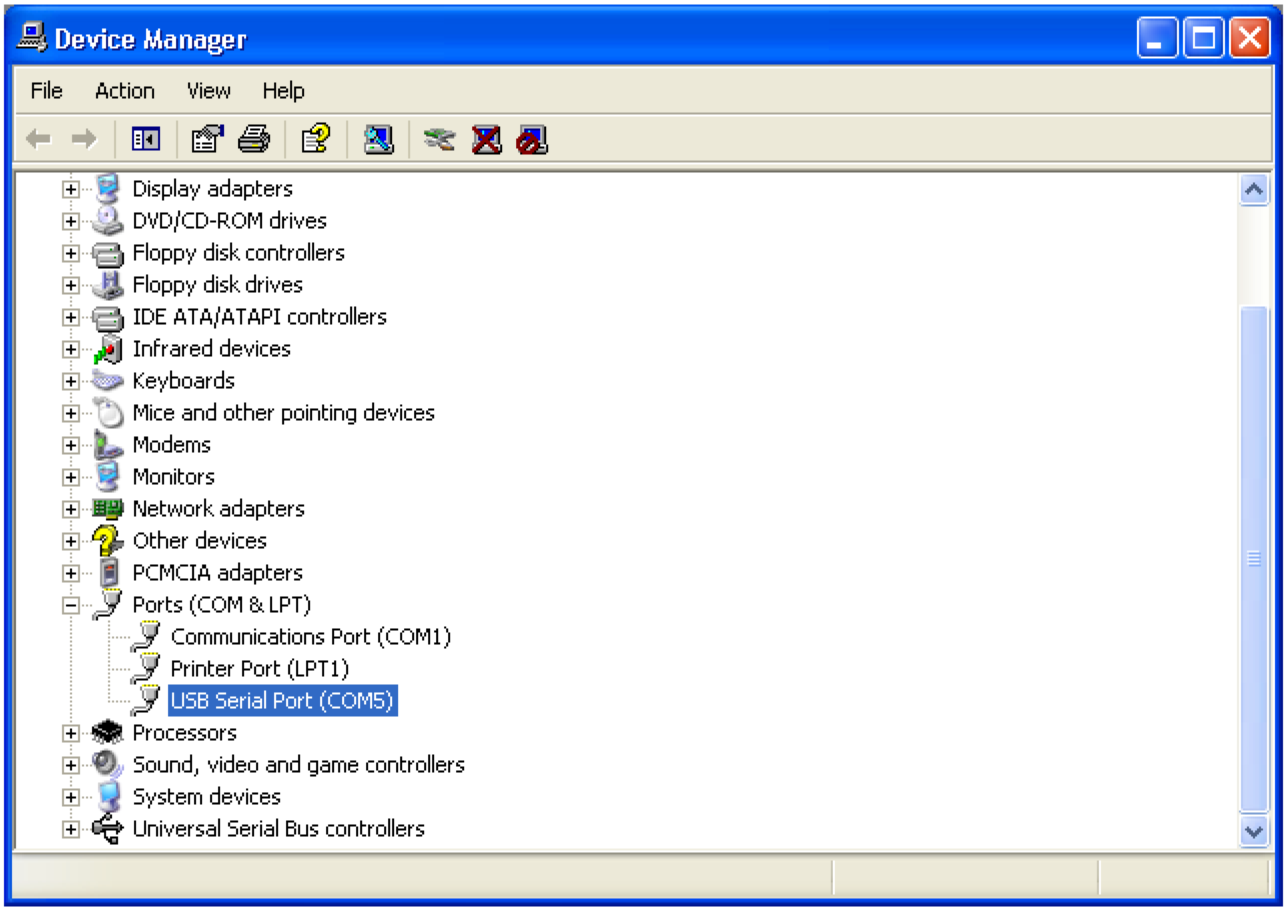
To program the Arduino board you need the Arduino environment.

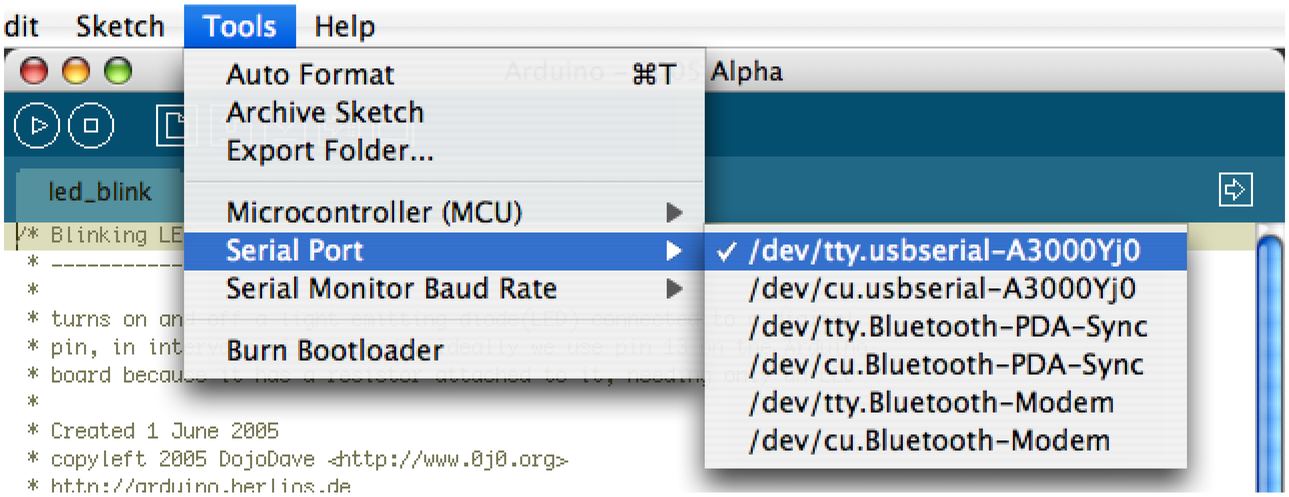
**Download Arduino**: Arduino 0006 for Windows

Restore
November 16, 2006, at 04:48 PM by David A. Mellis -
Added lines 38-45:

(:include HowtoExample:)

Select the serial device of the Arduino board from the Tools | Serial Port menu. On Windows, this should be COM1 or COM2 for a serial Arduino board, or COM3, COM4, or COM5 for a USB board. To find out, open the Windows Device Mananger (in the Hardware tab of System control panel). Look for a "USB Serial Port" in the Ports section; that's the Arduino board.





```
/* Blinking LE
 * ----------
 *
 * turns on an
 * pin, in int
 * board becau
 *
 * Created 1 June 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
```

Restore
November 04, 2006, at 07:01 AM by David A. Mellis -
Deleted lines 38-39:

(:include HowtoReferences:)

Restore
November 04, 2006, at 06:50 AM by David A. Mellis -
Changed lines 7-15 from:

## 2 | Download the Arduino environment

To program the Arduino board you need the Arduino environment.

**Download Arduino**: From the software page.

For more information, see the guide to the Arduino environment.

## 3 | Install the USB drivers

to:

(:include HowtoDownload:)

## 3 | Unzip the USB drivers

Changed lines 12-13 from:

On Windows, you will need to unzip `FTDI USB Drivers.zip`. Then, when you plug in the Arduino board, point the Windows *Add Hardware* wizard to the `FTDI USB Drivers` directory.

to:

On Windows, you will need to unzip `FTDI USB Drivers.zip`. In the next step ("Connect the board"), you will point Window's Add New Hardware wizard to these drivers.

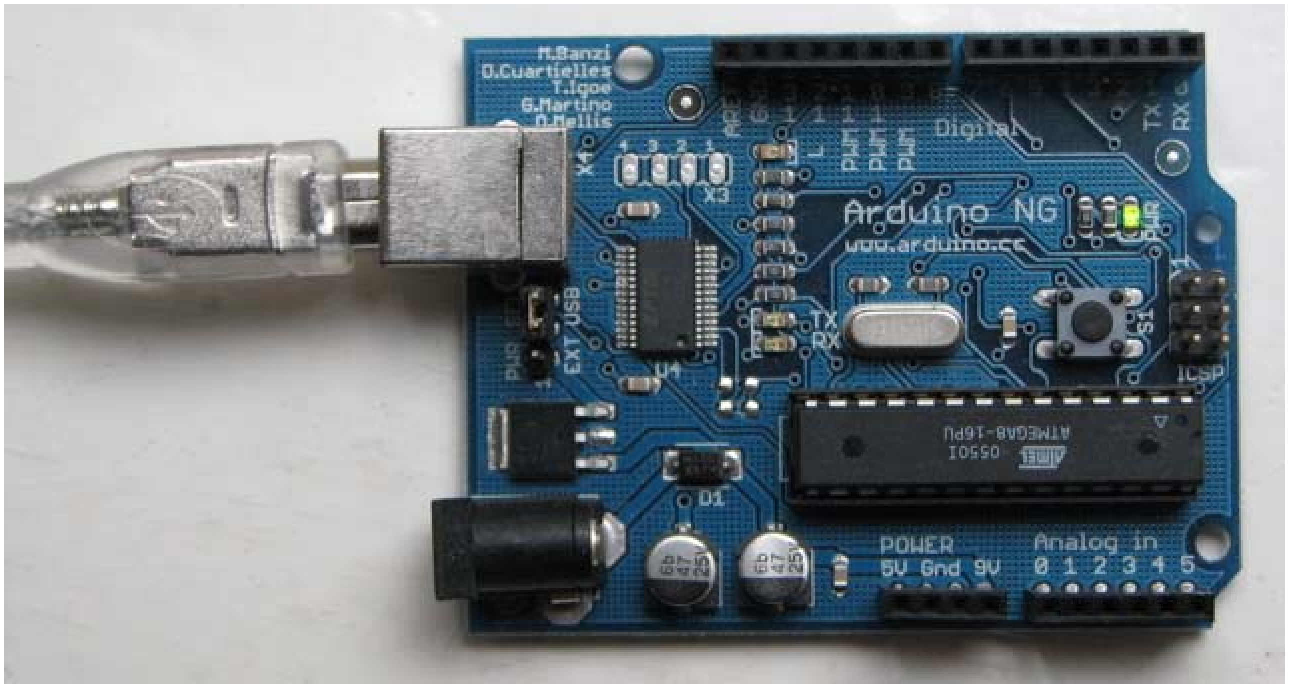
Changed lines 18-29 from:

## 4 | Connect the board

If you're using a serial board, power the board with an external power supply (6 to 25 volts DC, with the core of the connector positive). Connect the board to a serial port on your computer.

On the USB boards, the power source is selected by the jumper between the USB and power plugs. To power the board from the USB port (good for controlling low power devices like LEDs), place the jumper on the two pins closest to the USB plug. To power the board from an external power supply (needed for motors and other high current devices), place the jumper on the two pins closest to the power plug. Either way, connect the board to a USB port on your computer.

The power LED should go on.



On Windows, the Add New Hardware wizard will open. Tell it not to connect to Windows update and click next.

to:

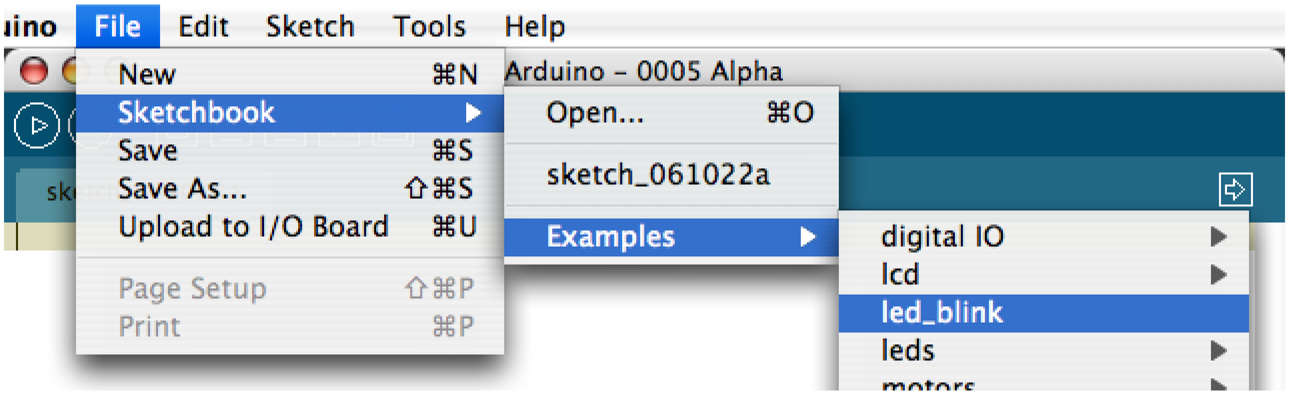
(:include HowtoConnect:)

The Add New Hardware wizard will open. Tell it not to connect to Windows update and click next.

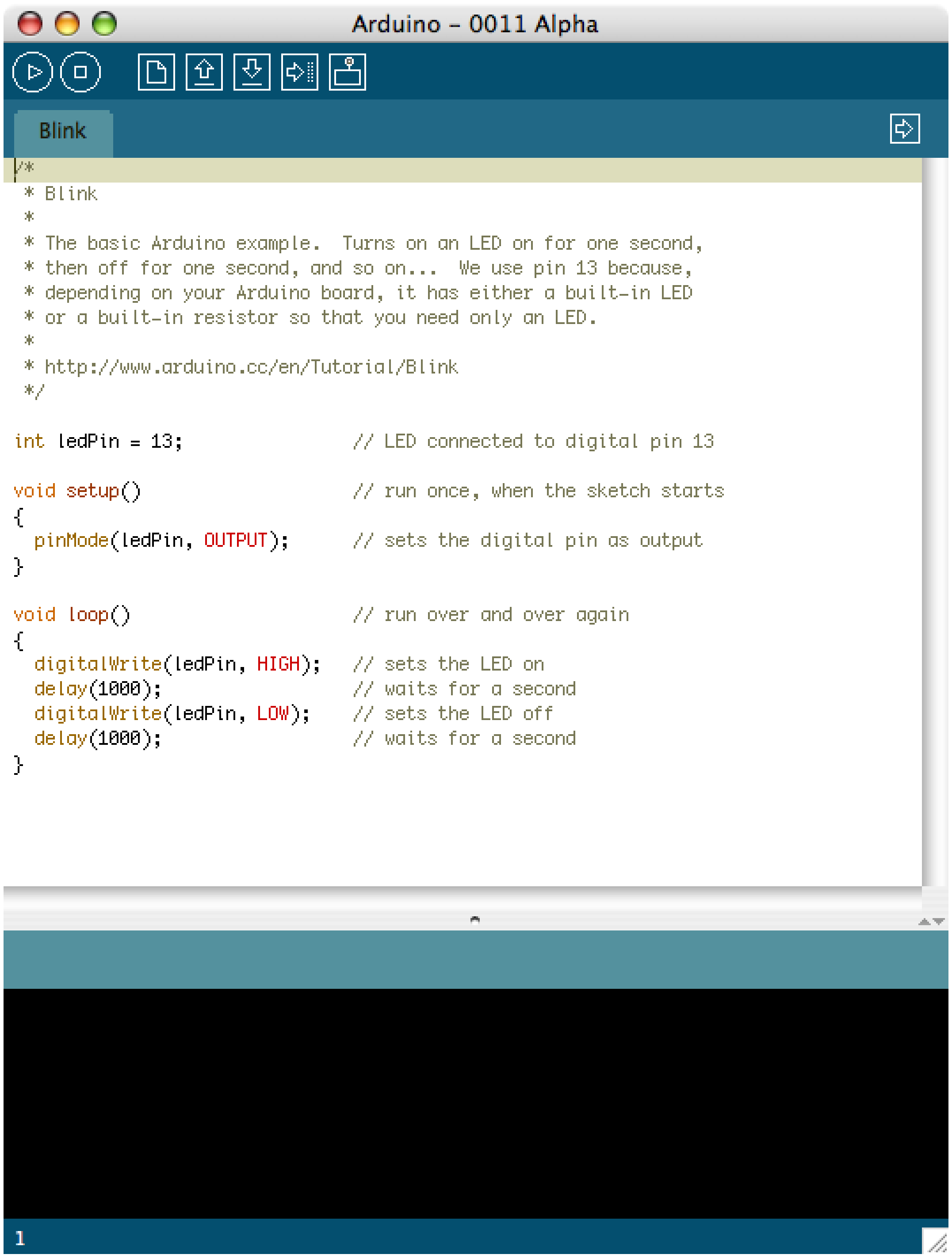
Changed lines 38-72 from:

## 5 | Upload a program

Open the LED blink example sketch: File > Sketchbook > Examples > led_blink.

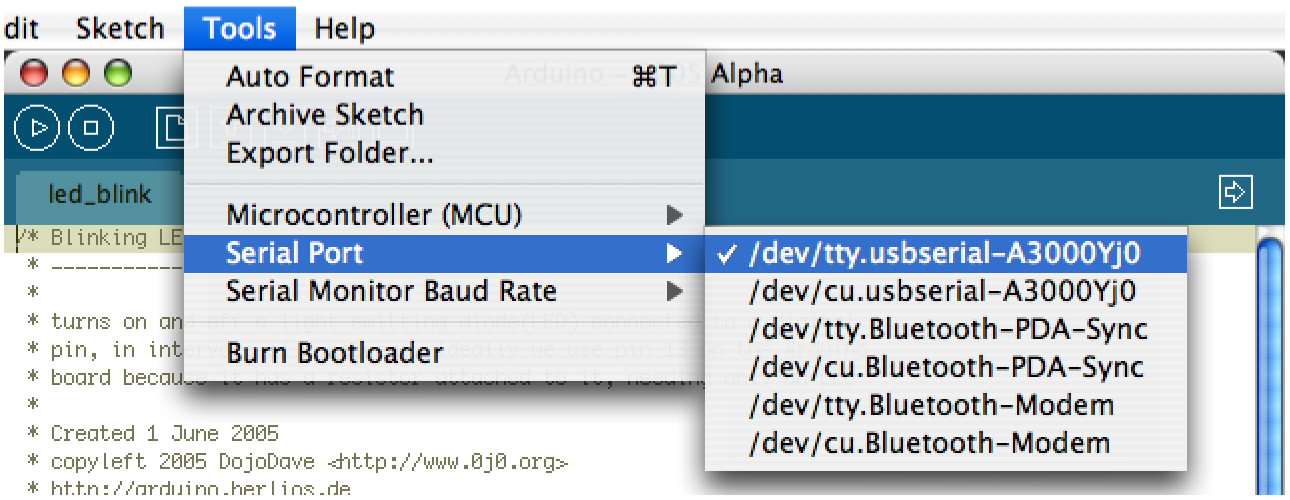


Here's what the code for the LED blink example looks like.



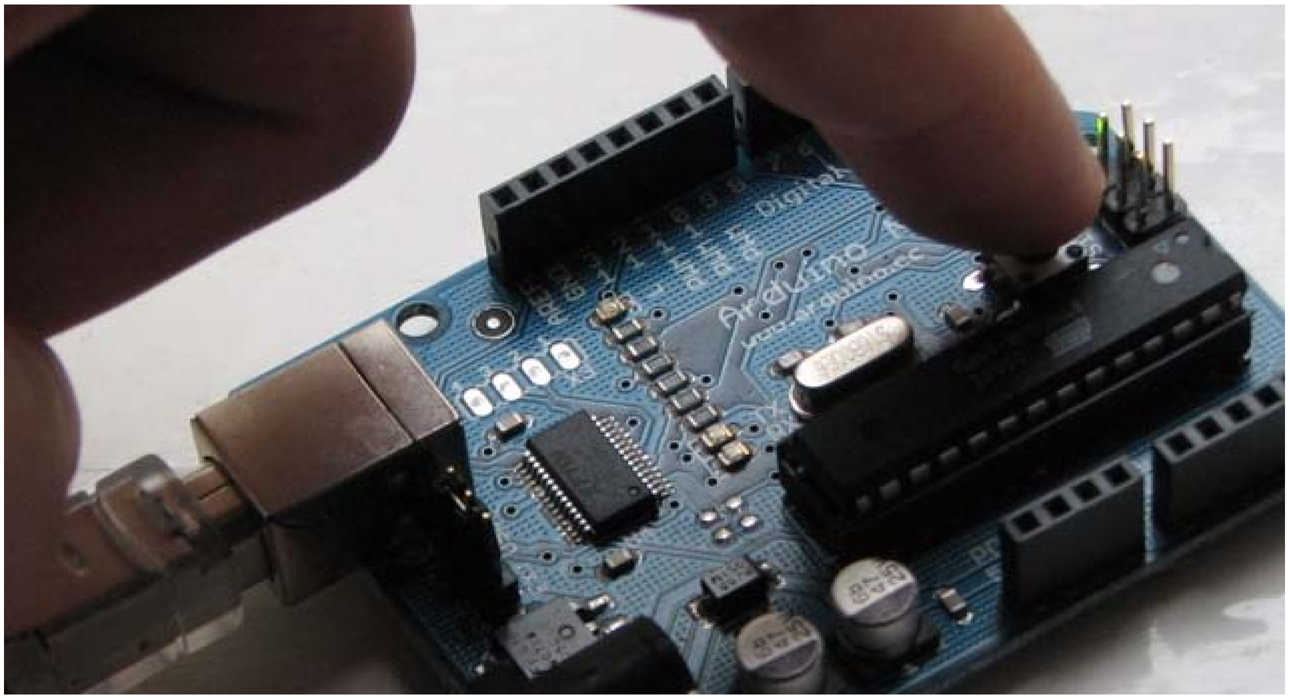
```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

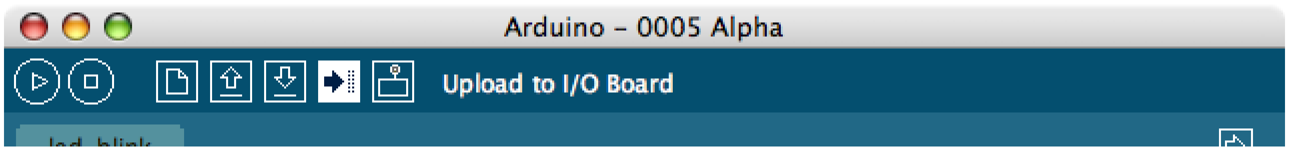
Select the serial device of the Arduino board from the Tools | Serial Port menu. On Windows, this should be COM1 or COM2 for a serial Arduino board, or COM3, COM4, or COM5 for a USB board.



```
/* Blinking LE
 * ----------
 *
 * turns on an
 * pin, in int
 * board becau
 *
 * Created 1 June 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
```

Push the reset button on the board then click the *Upload* button in the IDE. Wait a few seconds. If successful, the message "Done uploading." will appear in the status bar.





If the Arduino board doesn't show up in the Tools | Serial Port menu, or you get an error while uploading, please see the troubleshooting suggestions.

A few seconds after the upload finishes, you should see the amber (yellow) LED on the board start to blink.

## Learn More

- Read about the Arduino Environment
- Learn about the parts of the Arduino board
- See the tutorials for some example programs. (There are also some examples available in the examples directory inside the arduino directory.)
- Look up specific Arduino functions and syntax in the reference
- The Arduino programming language is compatible with the Wiring language allowing porting applications from the Wiring board to Arduino. Please note the differences between the Wiring and Processing languages.
- If you're having problems, check the FAQ.
- If you don't find a solution there, try posting in the forums.

to:

(:include HowtoUpload:)

(:include HowtoReferences:)

Restore
November 04, 2006, at 06:45 AM by David A. Mellis -
Changed lines 1-25 from:

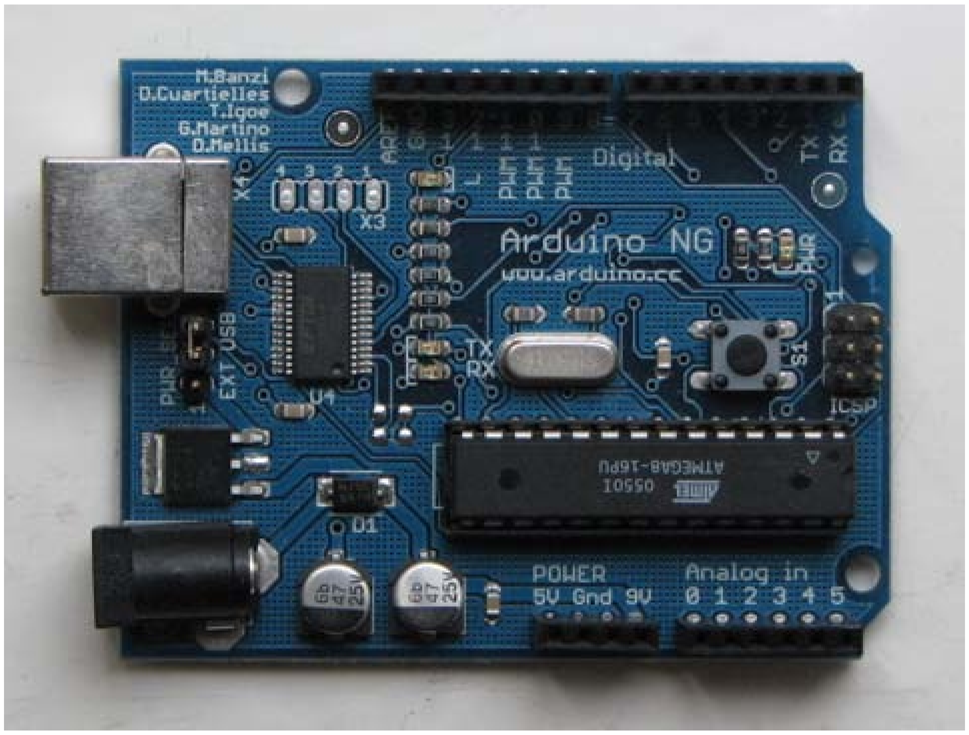
## Arduino Howto

These are the steps you need to follow in order to be up and running:

1. Get an Arduino board
2. Download the Arduino environment
3. Install the USB drivers
4. Connect the board
5. Upload a program

### 1 | Get an Arduino board

The Arduino i/o board is a simple circuit featuring the ATmega8 processor from Atmel. The board is composed of a printed circuit board (PCB) and electronic parts.



There are a few ways to get an Arduino board:

- **buy a ready made board**. See how you can buy a board or just the PCB.
    - European distributor
    - US distributor
- **build your own board**. If you want you can build your own PCB just by downloading the CAD files from the Hardware page. Extract the .brd file and send it to a PCB manufacturer. Be aware that manufacturing a single pcb will be very expensive. It's better to get together with other people and make 20 or 30 at a time. Since you get the full CAD files you can make your own customised version of Arduino. if you make modifications or fix bugs please send us your changes!
    - **purchase parts**. purchase the parts from any electronics store. The Serial version in particular has been designed to use the most basic parts that can be found anywhere in the world. The USB version on the other hand requires some advanced soldering skills because of the FTDI chip that is an smd part. Here is a list? of parts for the serial board.
    - **assemble the board**. We put together a step by step guide on how to build an arduino board. *Newbies: never soldered before? afraid of trashing thousands of boards before getting one properly soldered? fear not :) learn to master the art of soldering.*
    - **program the bootloader**. In order for the development environment to be able to program the chip, this has

to be programmed with a piece of code called *bootloader*. See the bootloader page on how to program it on your chip.

to:

## How To Get Arduino Running on Windows

(:include HowtoSteps:)

(:include HowtoGet:)

Restore
November 04, 2006, at 06:40 AM by David A. Mellis -
Added lines 1-105:

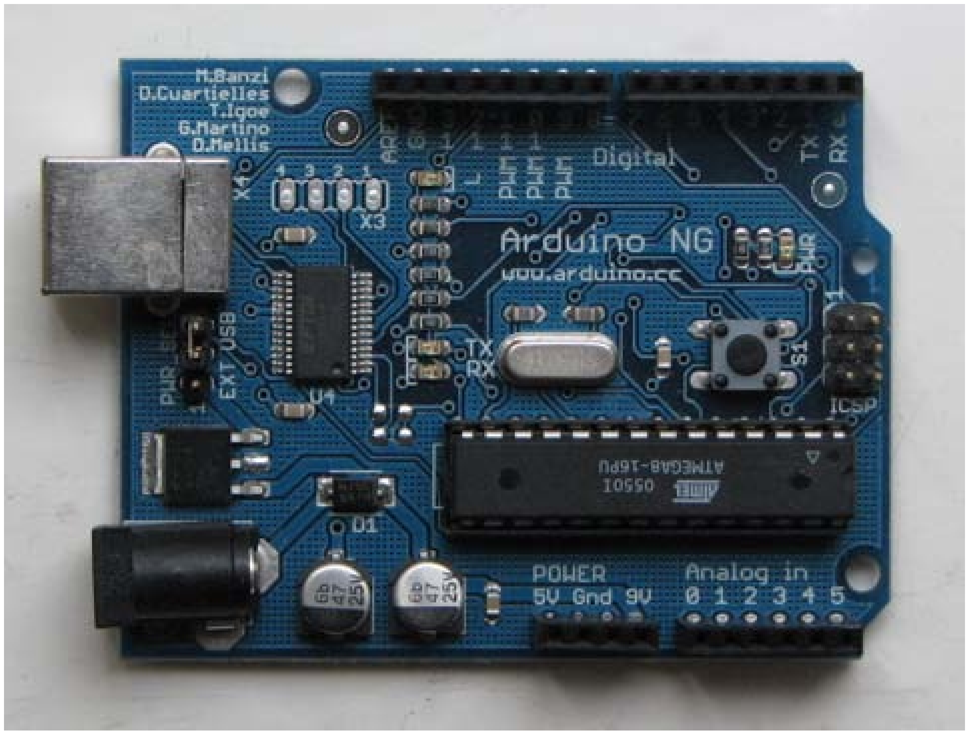
## Arduino Howto

These are the steps you need to follow in order to be up and running:

1. Get an Arduino board
2. Download the Arduino environment
3. Install the USB drivers
4. Connect the board
5. Upload a program

### 1 | Get an Arduino board

The Arduino i/o board is a simple circuit featuring the ATmega8 processor from Atmel. The board is composed of a printed circuit board (PCB) and electronic parts.



There are a few ways to get an Arduino board:

- **buy a ready made board**. See how you can buy a board or just the PCB.
    - European distributor
    - US distributor
- **build your own board**. If you want you can build your own PCB just by downloading the CAD files from the Hardware page. Extract the .brd file and send it to a PCB manufacturer. Be aware that manufacturing a single pcb will be very expensive. It's better to get together with other people and make 20 or 30 at a time. Since you get the full CAD files you can make your own customised version of Arduino. if you make modifications or fix bugs please send us your changes!
    - **purchase parts**. purchase the parts from any electronics store. The Serial version in particular has been designed to use the most basic parts that can be found anywhere in the world. The USB version on the other hand requires some advanced soldering skills because of the FTDI chip that is an smd part. Here is a list? of

parts for the serial board.

- **assemble the board**. We put together a step by step guide on how to build an arduino board. *Newbies: never soldered before? afraid of trashing thousands of boards before getting one properly soldered? fear not :) learn to master the art of soldering.*
- **program the bootloader**. In order for the development environment to be able to program the chip, this has to be programmed with a piece of code called *bootloader*. See the bootloader page on how to program it on your chip.

## 2 | Download the Arduino environment

To program the Arduino board you need the Arduino environment.

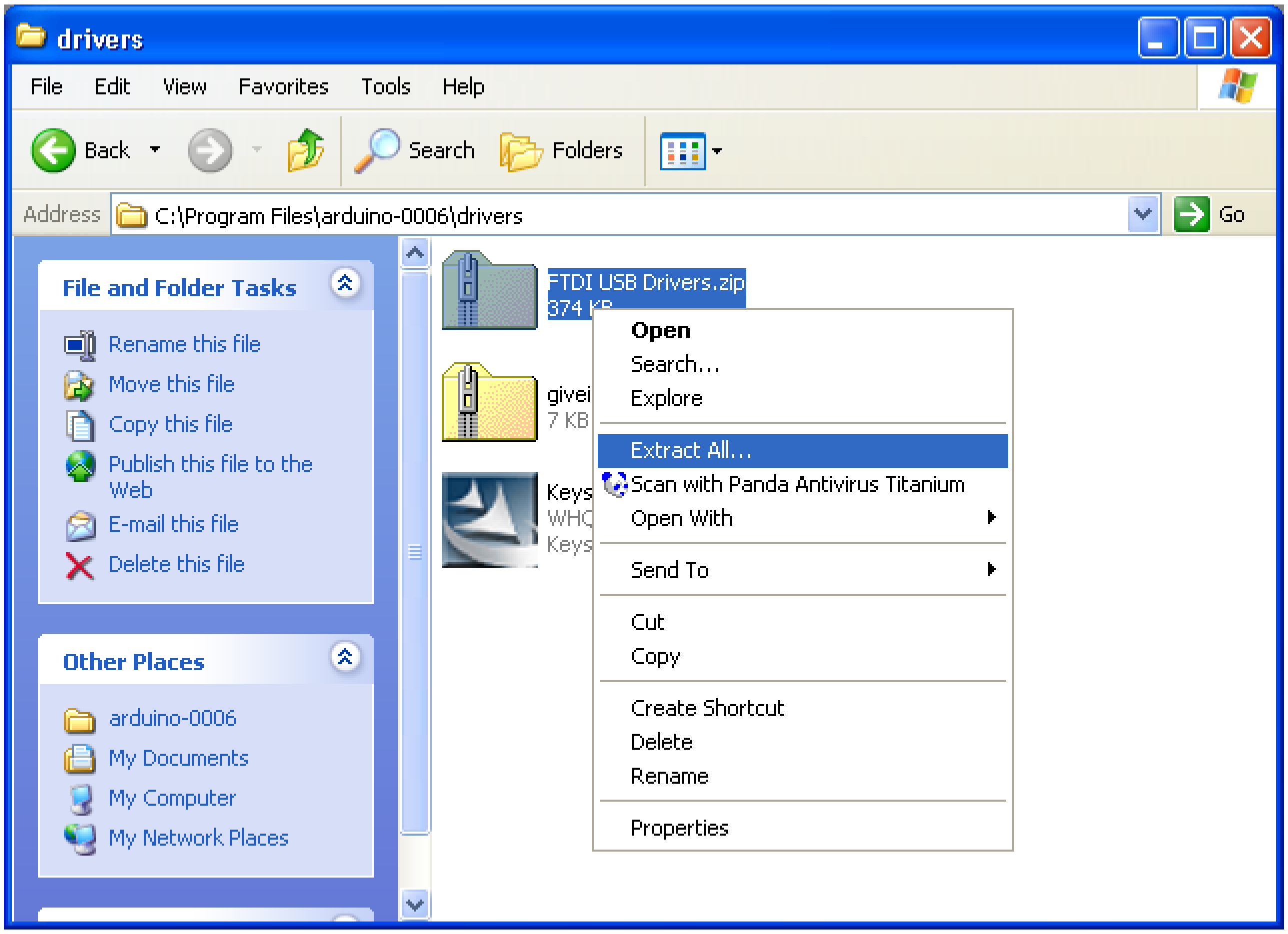
**Download Arduino**: From the software page.

For more information, see the guide to the Arduino environment.

## 3 | Install the USB drivers

If you are using a USB Arduino, you will need to install the drivers for the FTDI chip on the board. These can be found in the drivers directory of the Arduino distribution.

On Windows, you will need to unzip `FTDI USB Drivers.zip`. Then, when you plug in the Arduino board, point the Windows *Add Hardware* wizard to the `FTDI USB Drivers` directory.



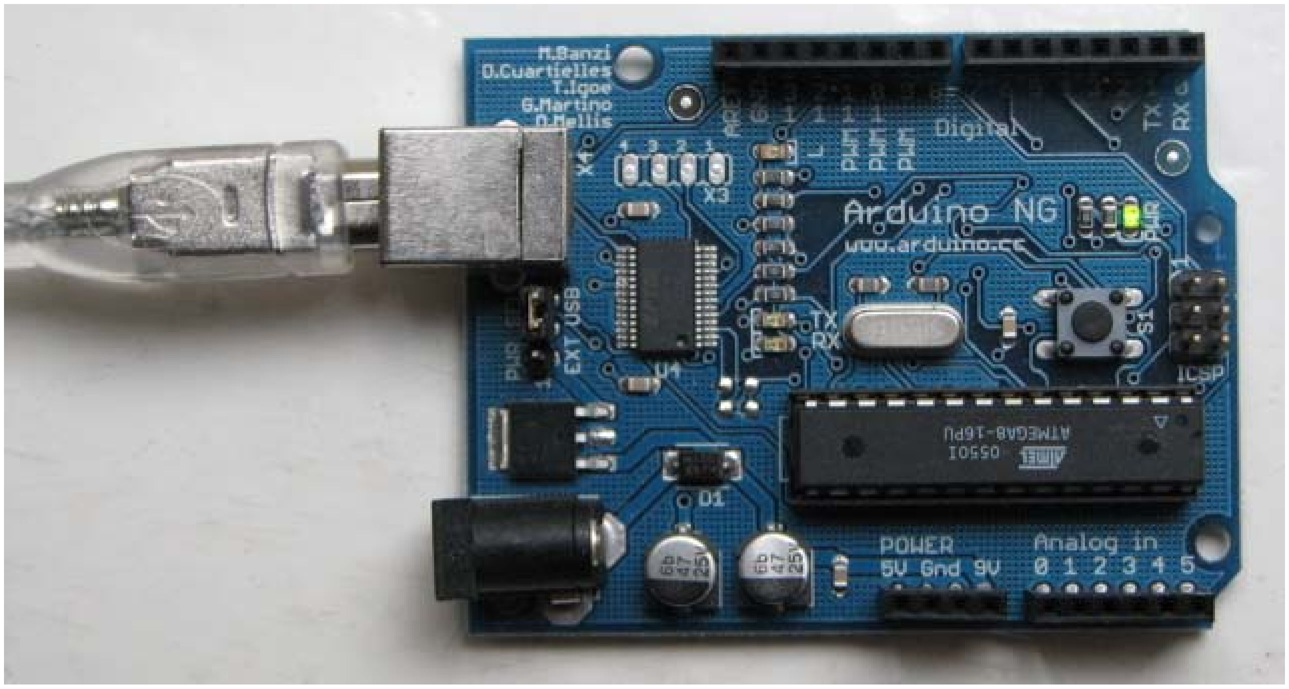
The latest version of the drivers can be found on the FTDI website.

## 4 | Connect the board

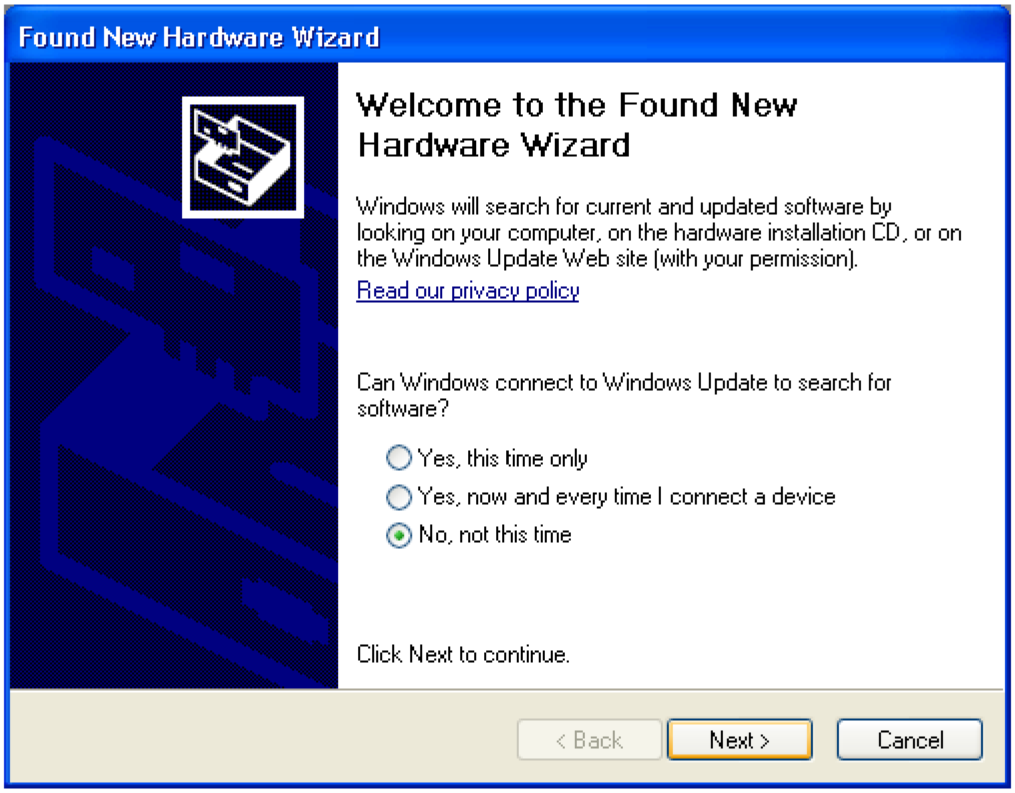
If you're using a serial board, power the board with an external power supply (6 to 25 volts DC, with the core of the connector positive). Connect the board to a serial port on your computer.

On the USB boards, the power source is selected by the jumper between the USB and power plugs. To power the board from the USB port (good for controlling low power devices like LEDs), place the jumper on the two pins closest to the USB plug. To power the board from an external power supply (needed for motors and other high current devices), place the jumper on the two pins closest to the power plug. Either way, connect the board to a USB port on your computer.

The power LED should go on.



On Windows, the Add New Hardware wizard will open. Tell it not to connect to Windows update and click next.



Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

Read our privacy policy

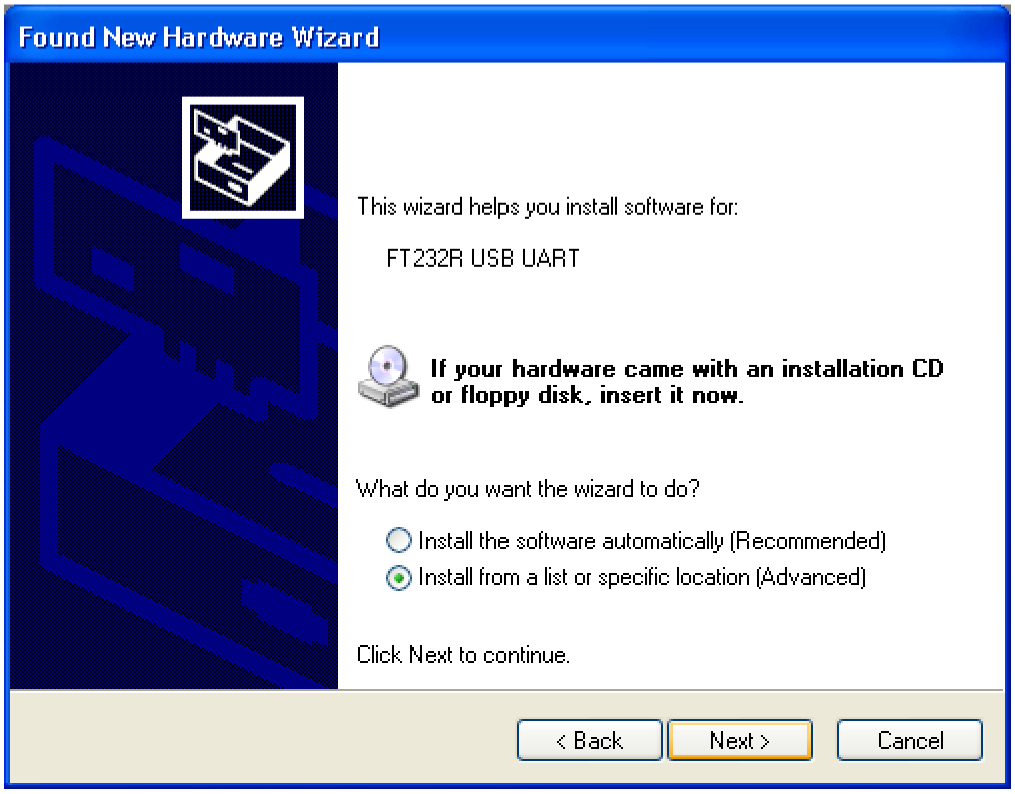
Can Windows connect to Windows Update to search for software?

○ Yes, this time only
○ Yes, now and every time I connect a device
[✓ No, not this time]

Click Next to continue.

< Back | Next > | Cancel

Then select "Install from a list or specified location (Advanced)" and click next.


Found New Hardware Wizard

This wizard helps you install software for:

FT232R USB UART

If your hardware came with an installation CD or floppy disk, insert it now.

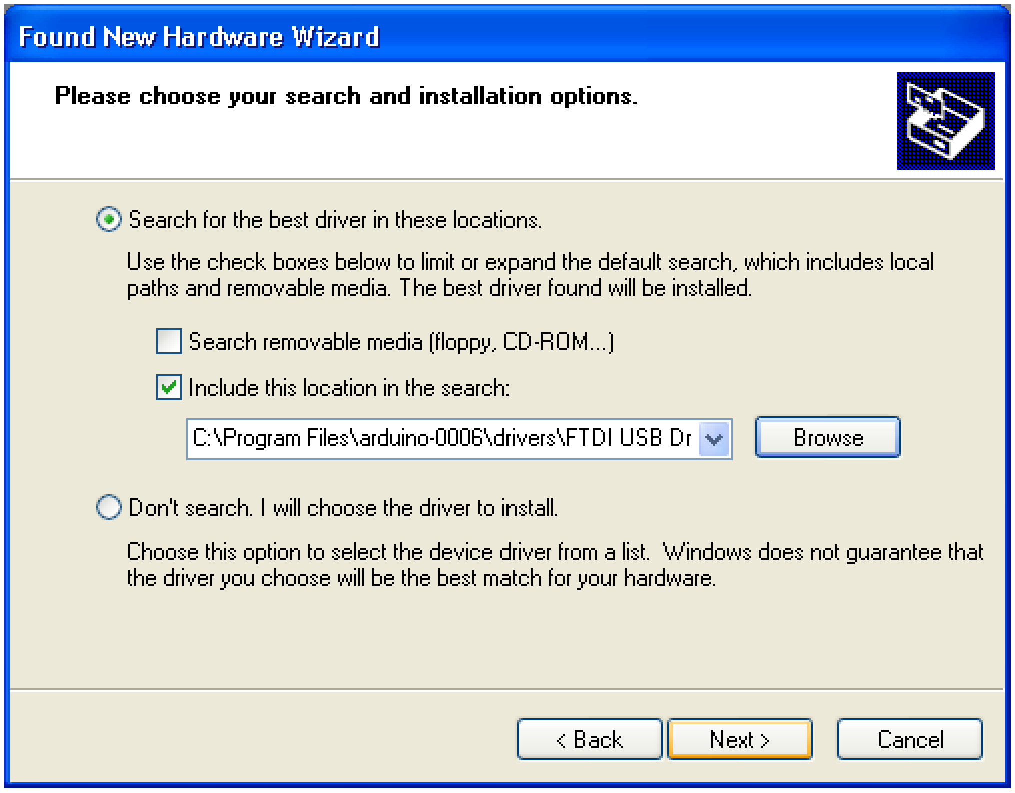
What do you want the wizard to do?

- ○ Install the software automatically (Recommended)
- [✓ Install from a list or specific location (Advanced)]

Click Next to continue.

< Back | Next > | Cancel

Make sure that "Search for the best driver in these locations is checked"; uncheck "Search removable media"; check "Include this location in the search" and browse to the location you unzipped the USB drivers to in the previous step. Click next.


Found New Hardware Wizard

Please choose your search and installation options.

[✓ Search for the best driver in these locations.]

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

- ☐ Search removable media (floppy, CD-ROM...)
- ☑ Include this location in the search:

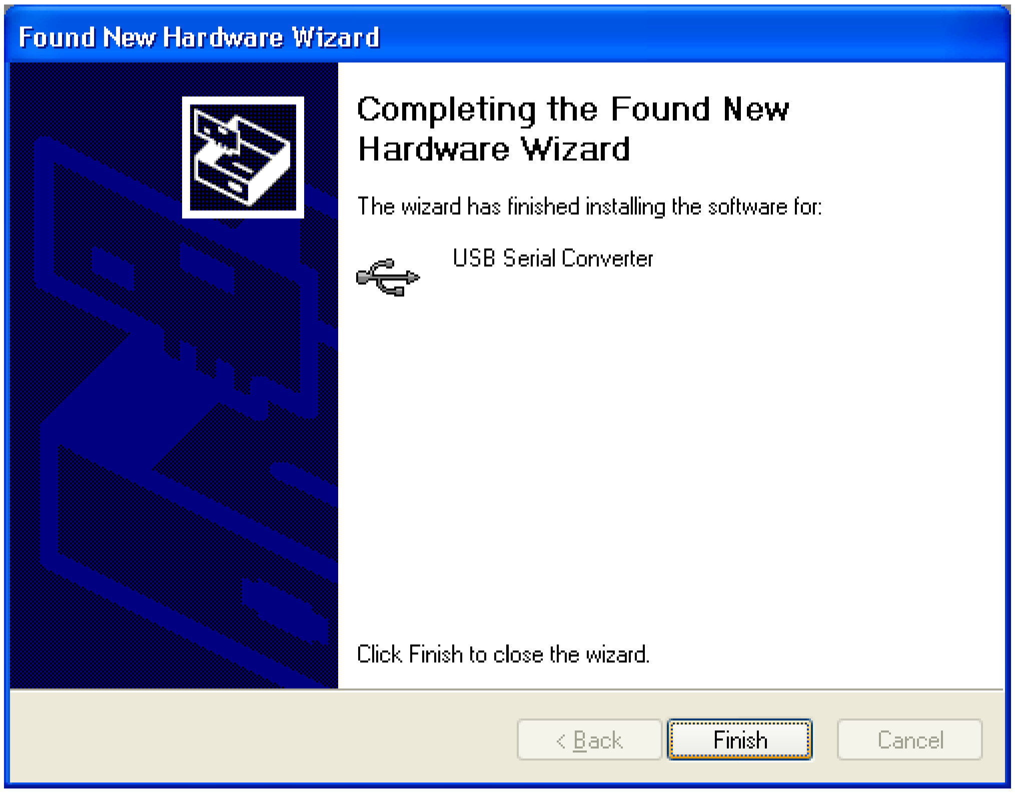
C:\Program Files\arduino-0006\drivers\FTDI USB Dr | Browse

○ Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

< Back | Next > | Cancel

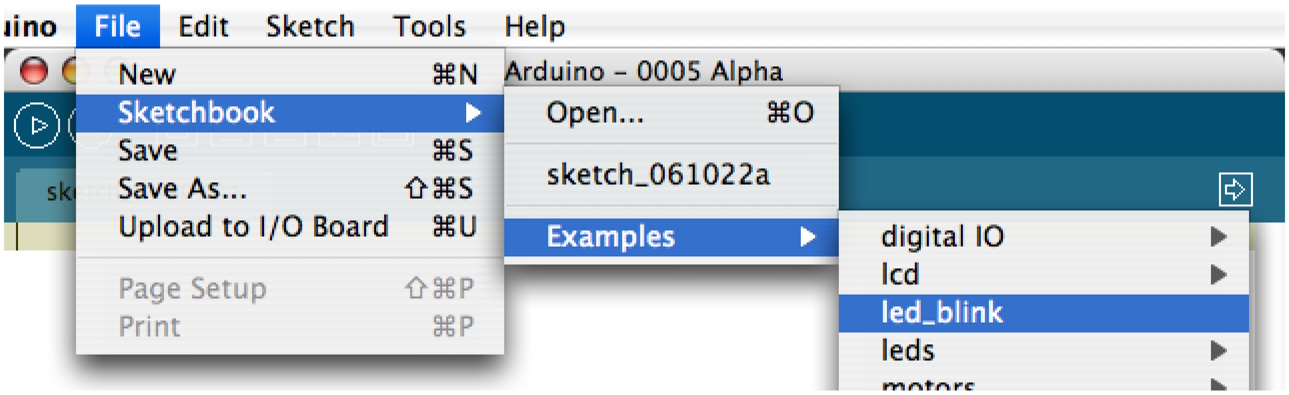
The wizard will search for the driver and then tell you that a "USB Serial Converter" was found. Click finish.


Found New Hardware Wizard
Completing the Found New Hardware Wizard
The wizard has finished installing the software for:
USB Serial Converter
Click Finish to close the wizard.
< Back
Finish
Cancel

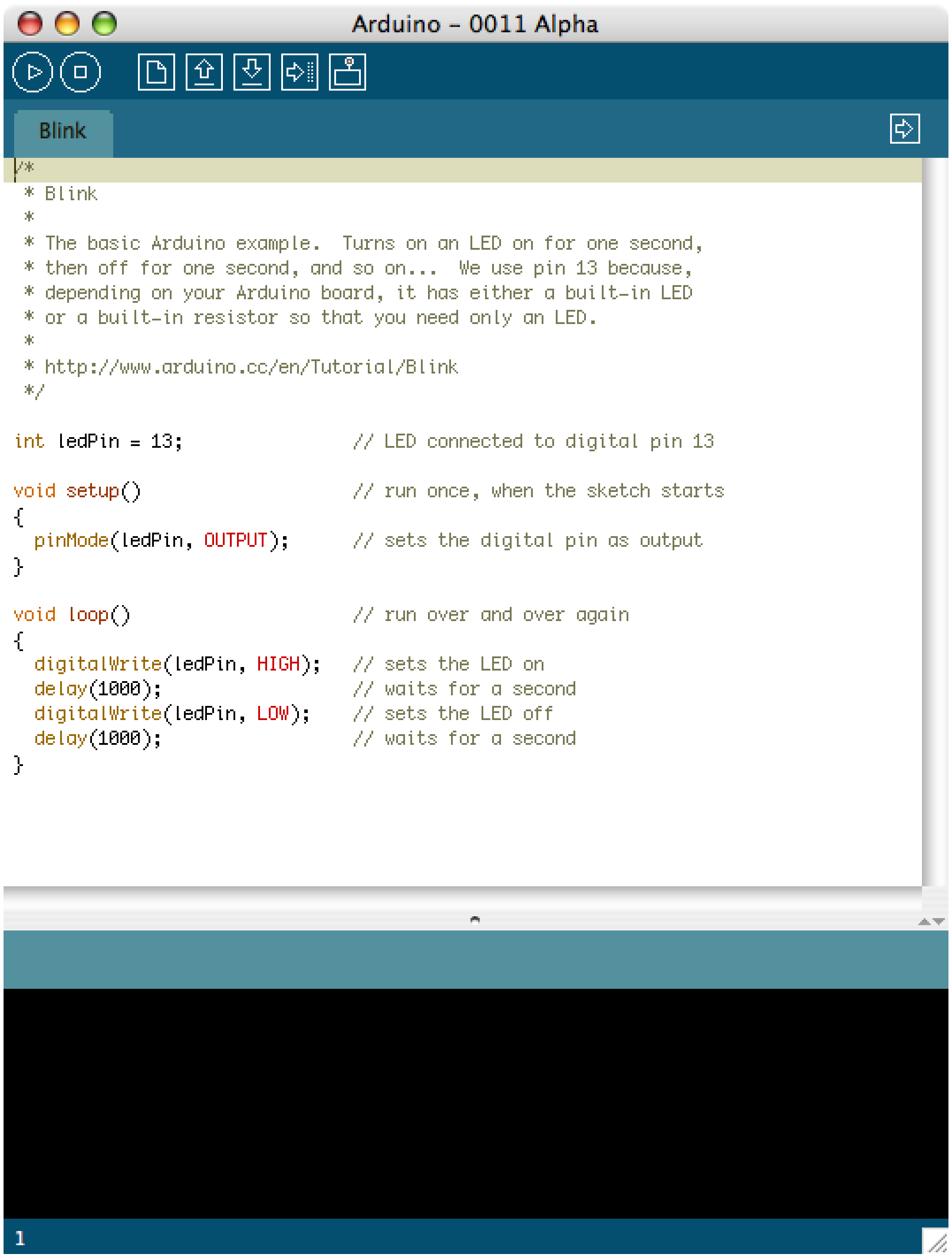
The new hardware wizard will appear again. Go through the same steps. This time, a "USB Serial Port" will be found.

**5 | Upload a program**

Open the LED blink example sketch: File > Sketchbook > Examples > led_blink.

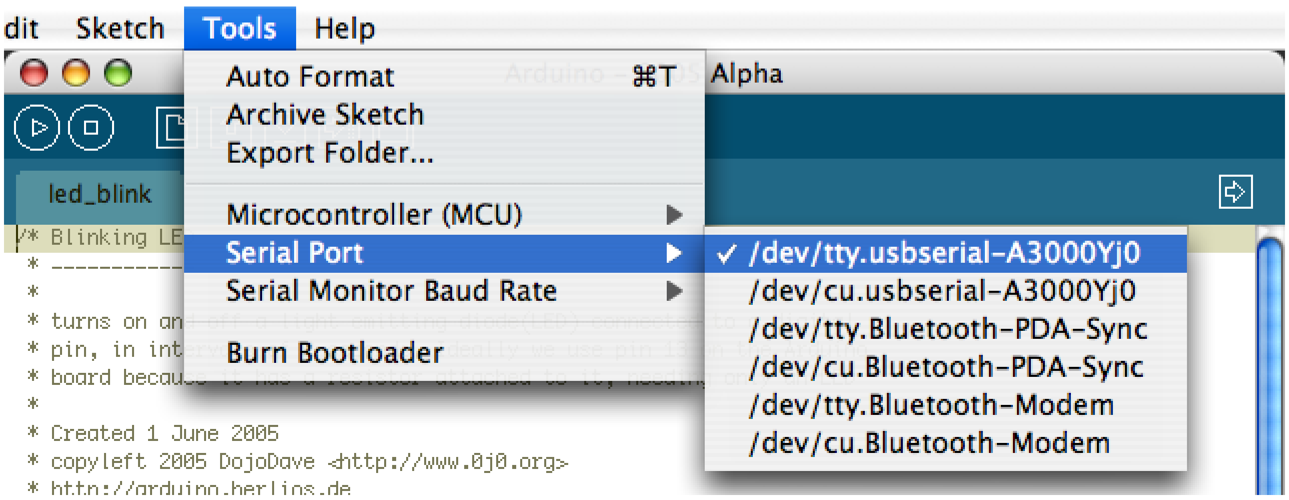


Here's what the code for the LED blink example looks like.



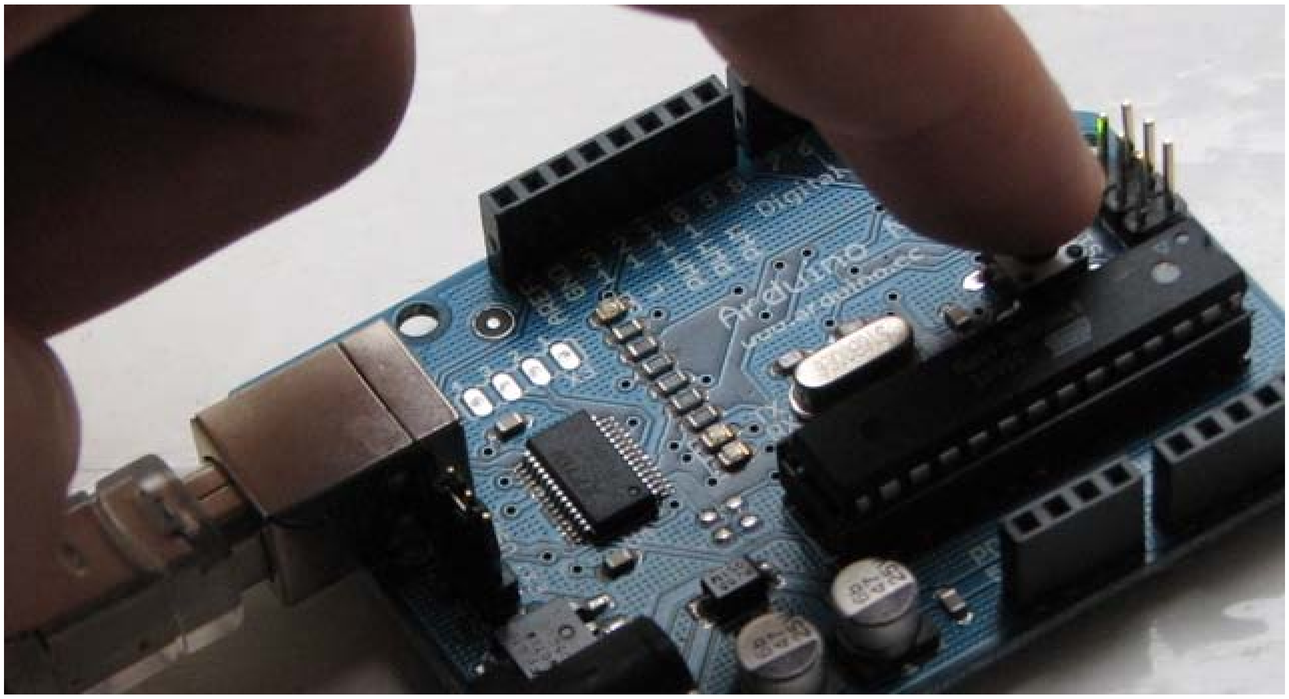
```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

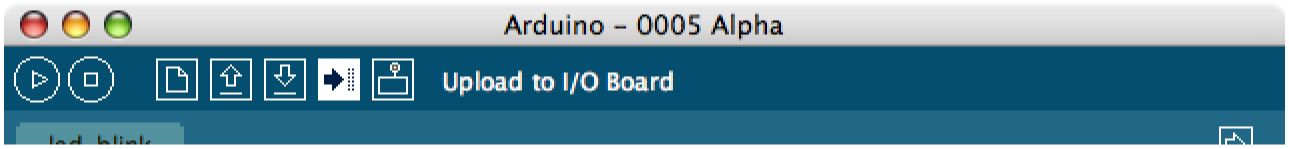
Select the serial device of the Arduino board from the Tools | Serial Port menu. On Windows, this should be COM1 or COM2 for a serial Arduino board, or COM3, COM4, or COM5 for a USB board.



```
/* Blinking LE
 * ----------
 *
 * turns on an
 * pin, in int
 * board becau
 *
 * Created 1 June 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
```

Push the reset button on the board then click the *Upload* button in the IDE. Wait a few seconds. If successful, the message "Done uploading." will appear in the status bar.





If the Arduino board doesn't show up in the Tools | Serial Port menu, or you get an error while uploading, please see the troubleshooting suggestions.

A few seconds after the upload finishes, you should see the amber (yellow) LED on the board start to blink.

## Learn More

- Read about the Arduino Environment
- Learn about the parts of the Arduino board
- See the tutorials for some example programs. (There are also some examples available in the examples directory inside the arduino directory.)
- Look up specific Arduino functions and syntax in the reference
- The Arduino programming language is compatible with the Wiring language allowing porting applications from the Wiring board to Arduino. Please note the differences between the Wiring and Processing languages.
- If you're having problems, check the FAQ.
- If you don't find a solution there, try posting in the forums.

# How To Get Arduino Running on Windows

*This document explains how to connect your Arduino board to the computer and upload your first sketch.*

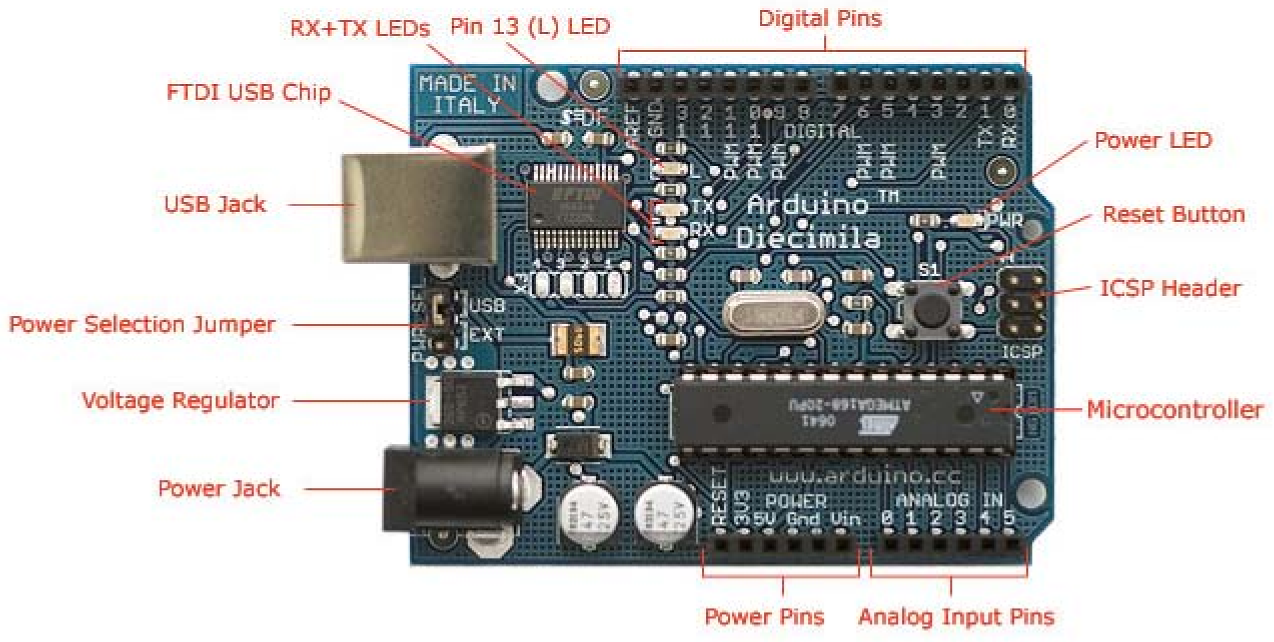
These are the steps that we'll go through:

1. Get an Arduino board and cable
2. Download the Arduino environment
3. Install the USB drivers
4. Connect the board
5. Connect an LED
6. Run the Arduino environment
7. Upload a program
8. Look for the blinking LED
9. Learn to use Arduino

## 1 | Get an Arduino board and cable

In this tutorial, we assume you're using an Arduino Diecimila. If you have another board, read the corresponding page in this getting started guide.

The Arduino Diecimila is a simple board that contains everything you need to start working with electronics and microcontroller programming. This diagram illustrates the major components of the board.



Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.

You also need a standard USB cable (A plug to B plug): the kind you would connect to a USB printer, for example.

## 2 | Download the Arduino environment

To program the Arduino board you need the Arduino environment.

**Download**: the latest version from the download page.

When the download 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.

## 3 | Locate the USB drivers

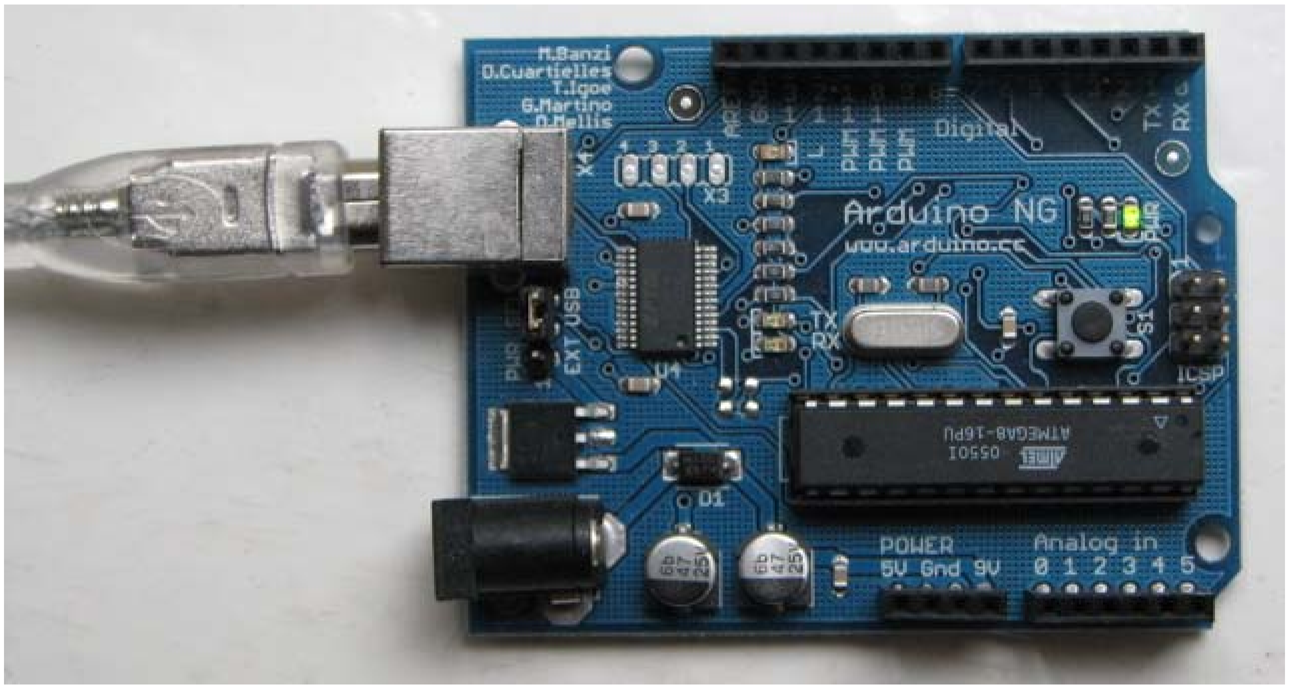
If you are using a USB Arduino, you will need to install the drivers for the FTDI chip on the board. These can be found in the `drivers/FTDI USB Drivers` directory of the Arduino distribution. In the next step ("Connect the board"), you will point Window's Add New Hardware wizard to these drivers.

The latest version of the drivers can be found on the FTDI website.

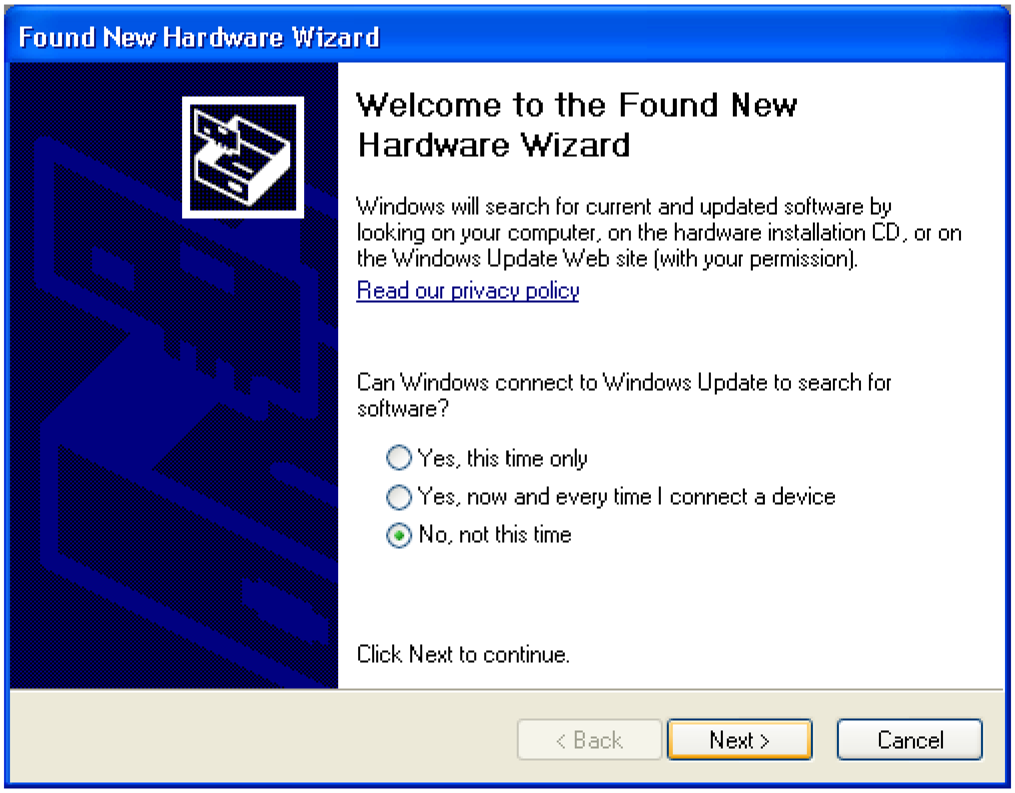
## 4 | Connect the board

The power source is selected by the jumper between the USB and power plugs. To power the board from the USB port (good for controlling low power devices like LEDs), place the jumper on the two pins closest to the USB plug. To power the board from an external power supply (6-12V), place the jumper on the two pins closest to the power plug. Either way, connect the board to a USB port on your computer.

The power LED should go on.



The Add New Hardware wizard will open. Tell it not to connect to Windows update and click next.


Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

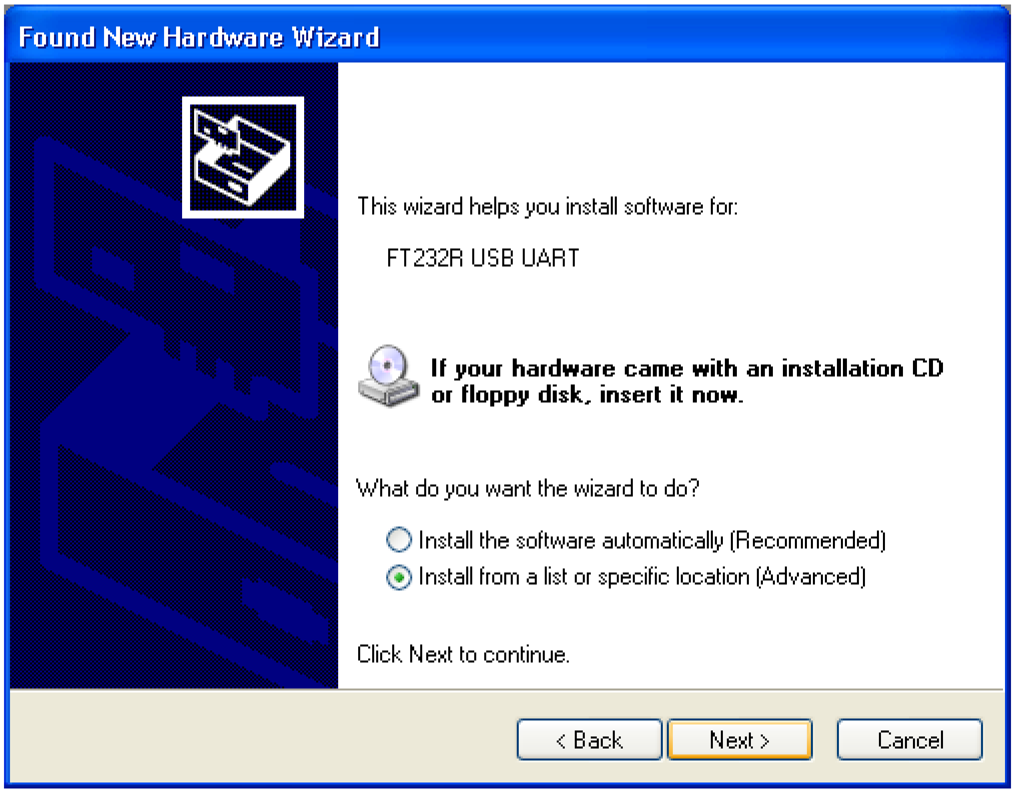
Read our privacy policy

Can Windows connect to Windows Update to search for software?

- Yes, this time only
- Yes, now and every time I connect a device
- [✓ No, not this time]

Click Next to continue.

Then select "Install from a list or specified location (Advanced)" and click next.


Found New Hardware Wizard

This wizard helps you install software for:

FT232R USB UART

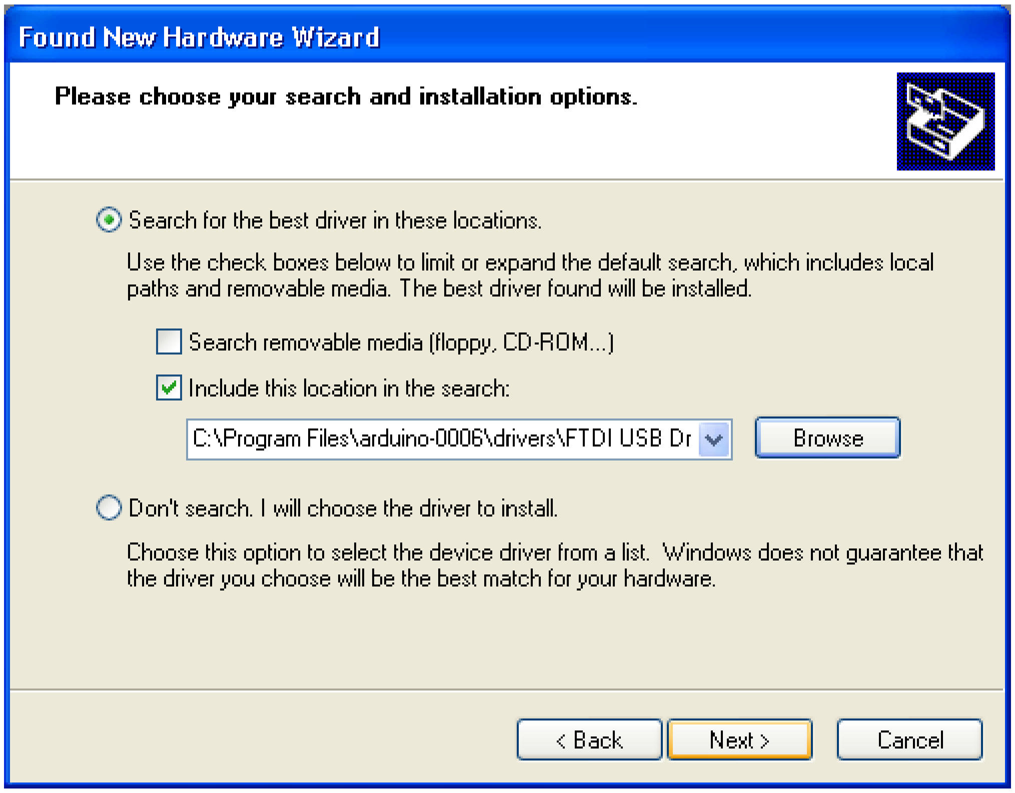
**If your hardware came with an installation CD or floppy disk, insert it now.**

What do you want the wizard to do?

- Install the software automatically (Recommended)
- [✓ Install from a list or specific location (Advanced)]

Click Next to continue.

Make sure that "Search for the best driver in these locations is checked"; uncheck "Search removable media"; check "Include this location in the search" and browse to the location you unzipped the USB drivers to in the previous step. Click next.


Found New Hardware Wizard

Please choose your search and installation options.

(●) Search for the best driver in these locations.

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

[ ] Search removable media (floppy, CD-ROM...)

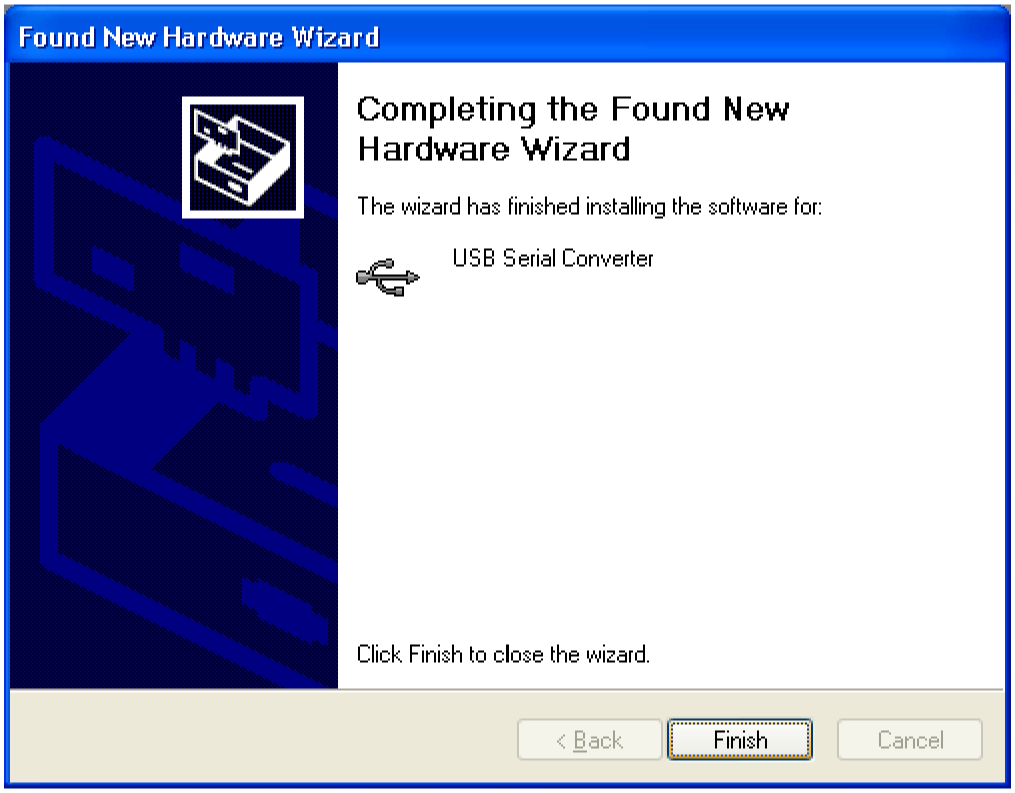
[✓] Include this location in the search:

C:\Program Files\arduino-0006\drivers\FTDI USB Dr

( ) Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

The wizard will search for the driver and then tell you that a "USB Serial Converter" was found. Click finish.


Found New Hardware Wizard

Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

USB Serial Converter

Click Finish to close the wizard.

The new hardware wizard will appear again. Go through the same steps. This time, a "USB Serial Port" will be found.

## 5 | Connect an LED (if you're using an older board)

The first sketch you will upload to the Arduino board blinks an LED. The Arduino Diecimila (and the original Arduino NG) has a built-in resistor on pin 13. On Arduino NG Rev. C and pre-NG Arduino boards, however, pin 13 does not have a built-in LED. On these boards, you'll need to connect the positive (longer) leg of an LED to pin 13 and the negative (shorter) leg to ground (marked "GND"). The LED will typically be flat on the side with the negative leg. Normally, you also need to use a resistor with the LED, but these boards have a resistor built-in on pin 13.

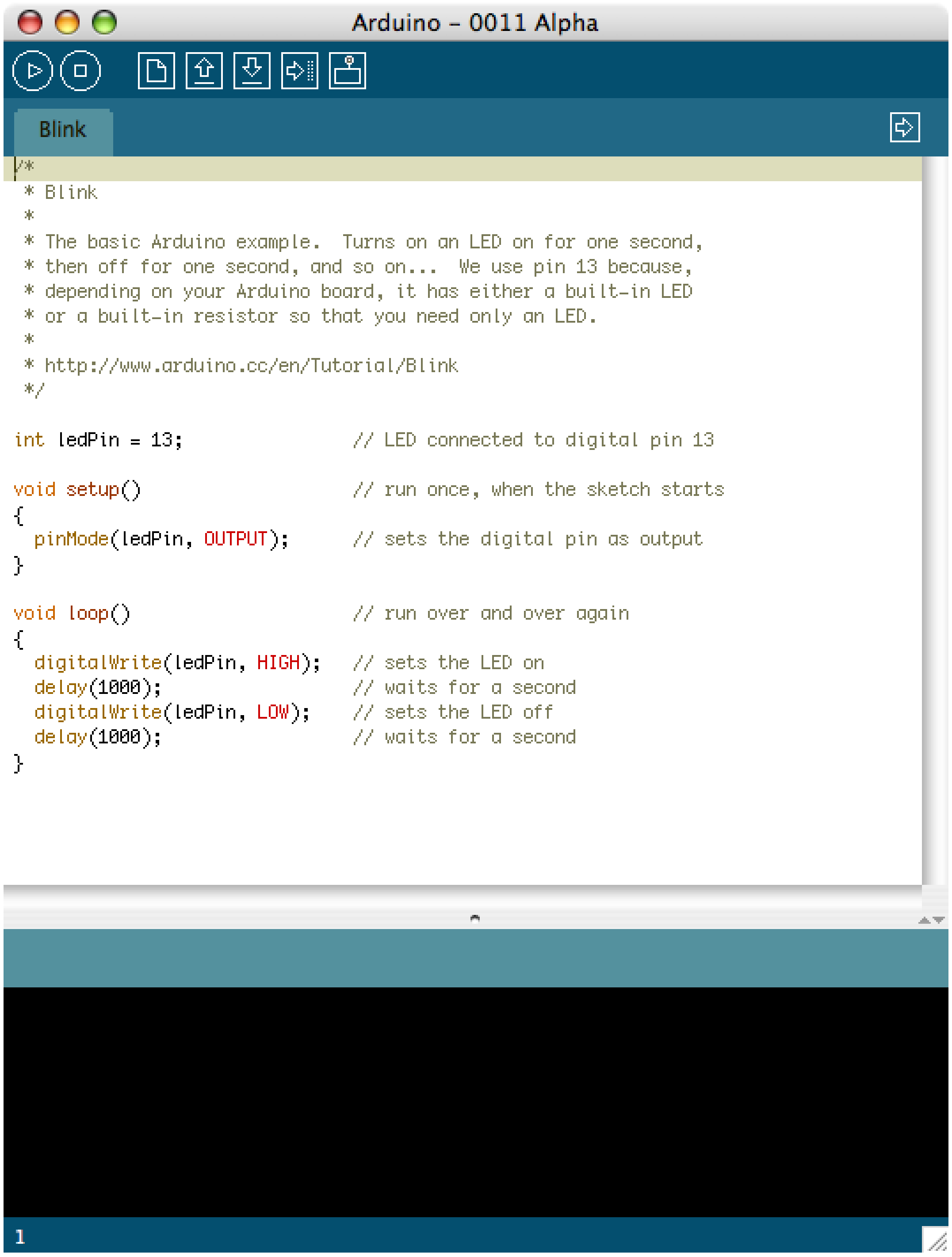
## 6 | Run the Arduino environment

Open the Arduino folder and double-click the Arduino application.

## 7 | Upload a program

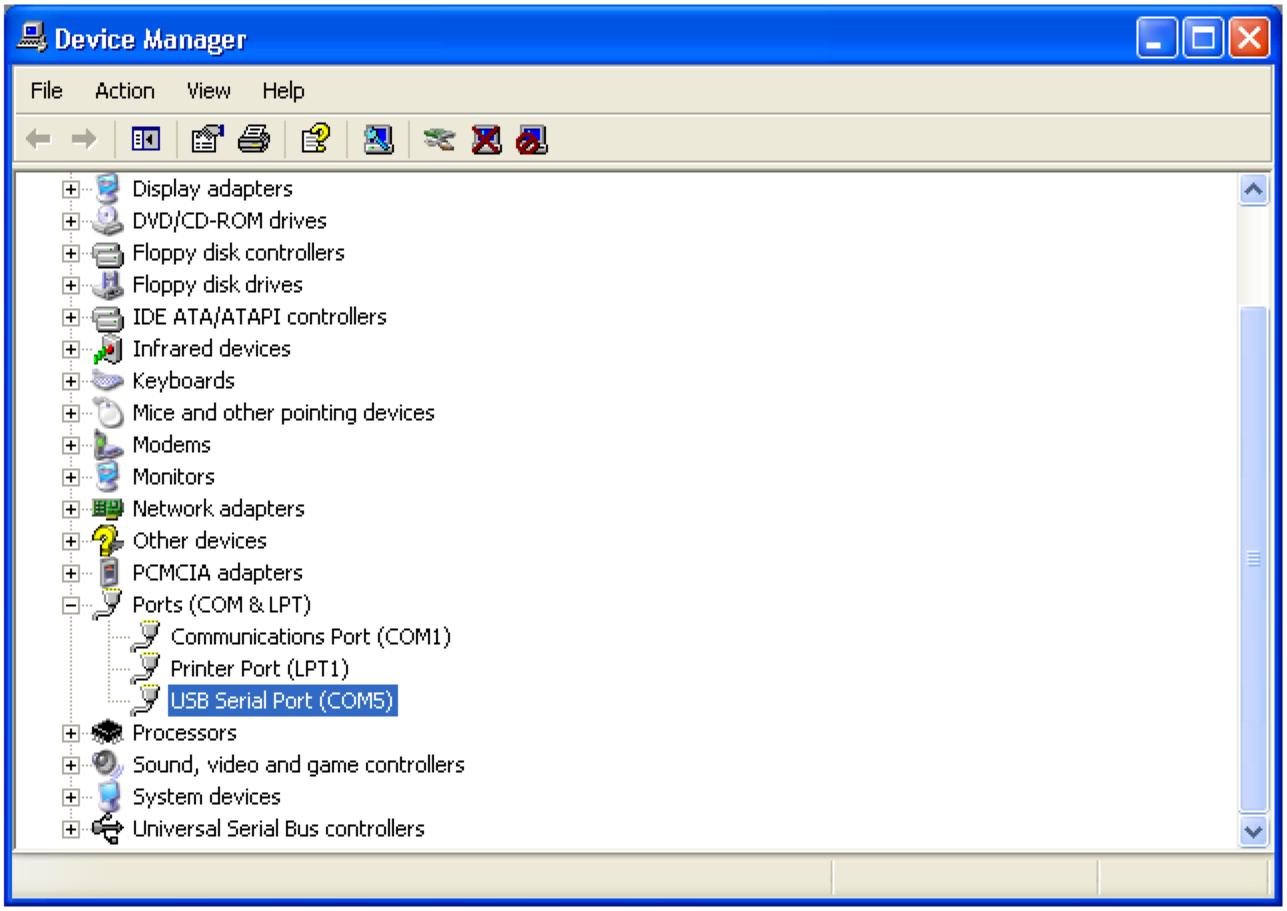
Open the LED blink example sketch: **File > Sketchbook > Examples > Digital > Blink**.

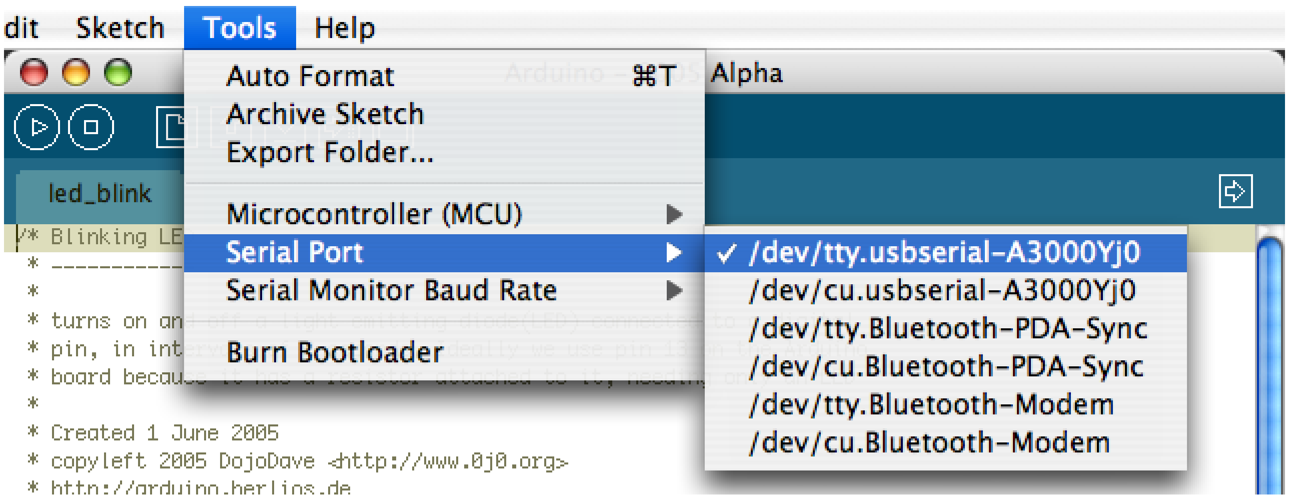
Here's what the code for the LED blink example looks like.



```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

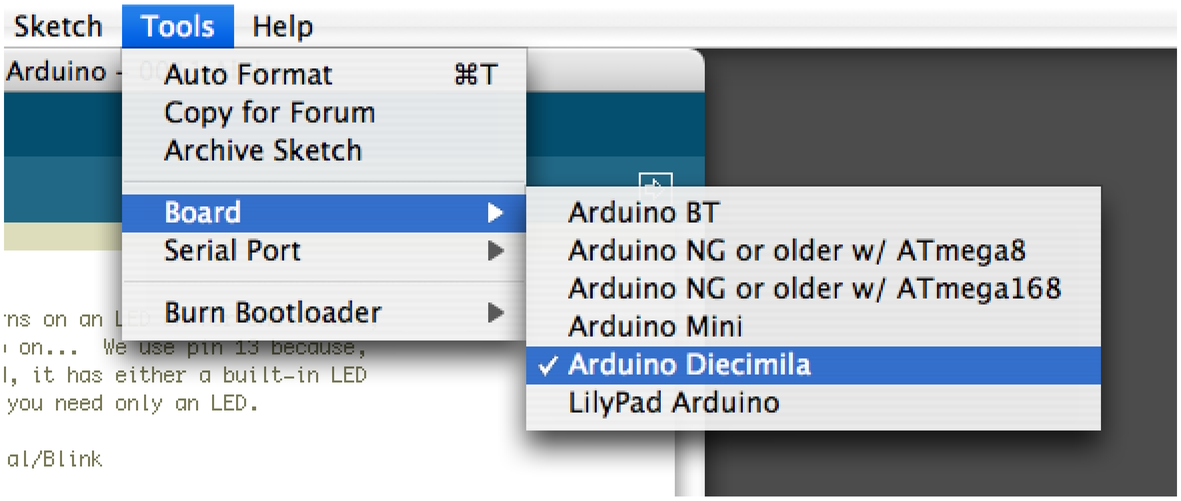
Select the serial device of the Arduino board from the Tools | Serial Port menu. On Windows, this should be COM1 or COM2 for a serial Arduino board, or COM3, COM4, or COM5 for a USB board. To find out, open the Windows Device Mananger (in the Hardware tab of System control panel). Look for a "USB Serial Port" in the Ports section; that's the Arduino board.



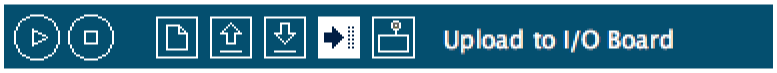


```
/* Blinking LE
 * -----------
 *
 * turns on an
 * pin, in int
 * board becau
 *
 * Created 1 June 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
```

Make sure that "Arduino Diecimila" is selected in the **Tools > Board** menu.



Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar. (*Note:* If you have an Arduino Mini, NG, or other board, you'll need to physically present the reset button on the board immediately before pressing the upload button.)



## 8 | Look for the blinking LED

A few seconds after the upload finishes, you should see the amber (yellow) LED on the board start to blink. If it does, congratulations! You've gotten Arduino up-and-running.

If you have problems, please see the troubleshooting suggestions.

## 9 | Learn to use Arduino

- Tutorials: try these example programs.
- Reference: read the reference for the Arduino language.

# Arduino

search

## Login to Arduino

| | |
|---|---|
| Username: | |
| Password: | |
| Keep me logged in: | ✓ |

Login

# Guide.MacOSX History

Hide minor edits - Show changes to markup

June 01, 2008, at 06:28 PM by David A. Mellis -
Added lines 3-4:

(:include HowtoIntro:)

Restore
June 01, 2008, at 06:18 PM by David A. Mellis -
Changed lines 7-13 from:

### 2 | Download the Arduino environment

To program the Arduino board you need the Arduino environment.

**Download**: the latest version of the software from the download page

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder with the Arduino software in side. Double-click the folder to open it.

to:

(:include HowtoDownload:)

Restore
June 01, 2008, at 06:14 PM by David A. Mellis -
Changed lines 8-13 from:

To program the Arduino board you need the Arduino environment. If you have an older Mac like a Powerbook, iBook, G4 or G5, you need the Arduino for PPC. If you have a newer Mac like an MacBook, MacBook Pro, or Mac Pro, you need the Intel version.

**Download**: Arduino 0011

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder called **arduino-0011**. Double-click the folder to open it.

to:

To program the Arduino board you need the Arduino environment.

**Download**: the latest version of the software from the download page

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder with the Arduino software in side. Double-click the folder to open it.

Changed lines 17-18 from:

On the Mac, mount the `FTDIUSBSerialDriver_v2_1_9.dmg` (on PPC machines) or the `FTDIUSBSerialDriver_v2_2_9_Intel.dmg` (on Intel machines) disk image and run the included `FTDIUSBSerialDriver.pkg`.

to:

If you have an older Mac like a Powerbook, iBook, G4 or G5, you should use the the PPC drivers: `FTDIUSBSerialDriver_v2_1_9.dmg`. If you have a newer Mac like an MacBook, MacBook Pro, or Mac Pro, you need the Intel drivers: `FTDIUSBSerialDriver_v2_2_9_Intel.dmg`. Double-click to mount the disk image and run the included `FTDIUSBSerialDriver.pkg`.

Restore

March 28, 2008, at 05:54 PM by David A. Mellis -
Changed lines 10-13 from:

**Download**: Arduino 0010

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder called **arduino-0009**. Double-click the folder to open it.

to:

**Download**: Arduino 0011

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder called **arduino-0011**. Double-click the folder to open it.

Changed lines 17-18 from:

On the Mac, mount the `FTDIUSBSerialDriver_v2_1_6.dmg` (on PPC machines) or the `FTDIUSBSerialDriver_v2_2_6_Intel.dmg` (on Intel machines) disk image and run the included `FTDIUSBSerialDriver.pkg`.

to:

On the Mac, mount the `FTDIUSBSerialDriver_v2_1_9.dmg` (on PPC machines) or the `FTDIUSBSerialDriver_v2_2_9_Intel.dmg` (on Intel machines) disk image and run the included `FTDIUSBSerialDriver.pkg`.

Restore
March 27, 2008, at 05:38 PM by David A. Mellis -
Changed lines 10-11 from:

**Download**: Arduino 0009 for PPC or Arduino 0009 for Intel

to:

**Download**: Arduino 0010

Restore
August 06, 2007, at 08:45 PM by David A. Mellis -
Changed lines 12-13 from:

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder called **arduino-0008**. Double-click the folder to open it.

to:

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder called **arduino-0009**. Double-click the folder to open it.

Restore
August 06, 2007, at 08:45 PM by David A. Mellis - updating software version to 0009.
Changed lines 10-11 from:

**Download**: Arduino 0008 for PPC or Arduino 0008 for Intel

to:

**Download**: Arduino 0009 for PPC or Arduino 0009 for Intel

Restore
June 09, 2007, at 06:55 PM by David A. Mellis -
Changed lines 12-13 from:

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder called **arduino-0007**. Double-click the folder to open it.

to:

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder called **arduino-0008**. Double-click the folder to open it.

Restore
June 09, 2007, at 06:54 PM by David A. Mellis -
Changed lines 10-11 from:

**Download**: Arduino 0007 for PPC or Arduino 0007 for Intel

to:

**Download**: Arduino 0008 for PPC or Arduino 0008 for Intel

Restore
March 03, 2007, at 02:11 PM by David A. Mellis -
Added lines 25-26:

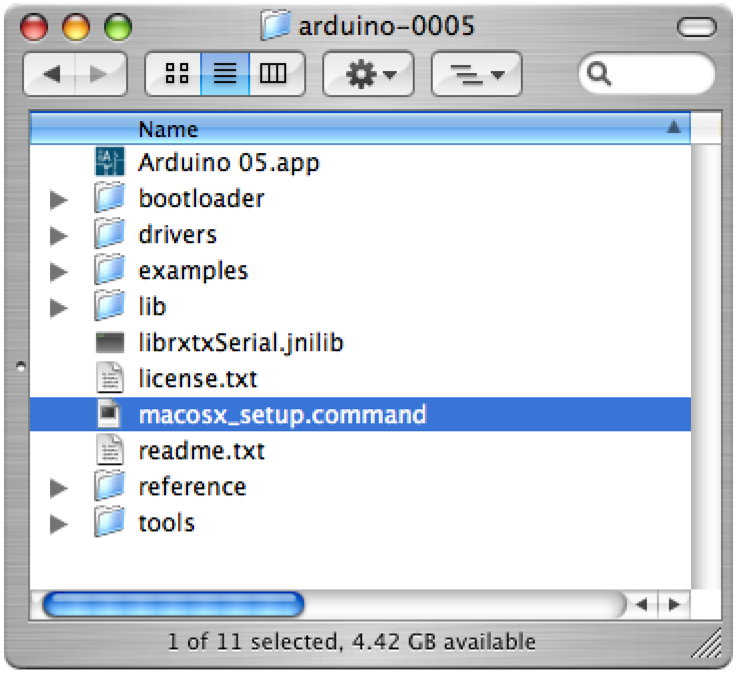
(:include HowtoLED:)

Restore
December 25, 2006, at 06:28 PM by David A. Mellis - removing reference to macosx_setup.command and updating links to point to 0007.
Changed lines 10-17 from:

**Download**: Arduino 0006 for PPC or Arduino 0006 for Intel

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder called **arduino-0006**. Double-click the folder to open it.

Then, double-click **macosx_setup.command**. This will open a Terminal and ask you if you want to continue. Type **y** and press return. When prompted, type your password and press return.



to:

**Download**: Arduino 0007 for PPC or Arduino 0007 for Intel

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder called **arduino-0007**. Double-click the folder to open it.

Restore
December 04, 2006, at 04:35 PM by David A. Mellis -
Changed lines 12-13 from:

After downloading the IDE, run the `macosx_setup.command`. It corrects permission on a few files for use with the serial port and will prompt you for your password. You may need to reboot after running this script.

to:

When the download finishes, unzip the downloaded file by double-clicking it. This should create a folder called **arduino-0006**. Double-click the folder to open it.

Then, double-click **macosx_setup.command**. This will open a Terminal and ask you if you want to continue. Type **y** and press return. When prompted, type your password and press return.

Changed lines 19-20 from:

If you are using a USB Arduino, you will need to install the drivers for the FTDI chip on the board. These can be found in the `drivers` directory of the Arduino distribution.

to:

If you are using a USB Arduino, you will need to install the drivers for the FTDI chip on the board. These can be found in the **drivers** directory of the Arduino distribution.

Changed lines 25-26 from:

The latest version of the drivers can be found on the FTDI website.

to:

(The latest version of the drivers can be found on the FTDI website.)

Restore
December 04, 2006, at 04:31 PM by David A. Mellis - linking directly to the Arduino downloads.
Changed lines 7-8 from:

(:include HowtoDownload:)

to:

### 2 | Download the Arduino environment

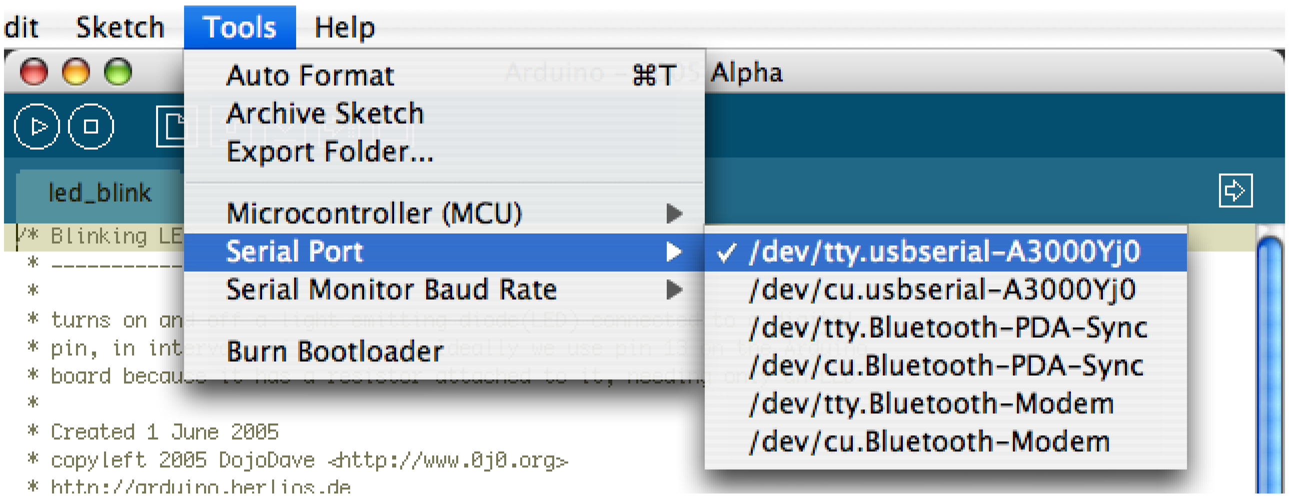
To program the Arduino board you need the Arduino environment. If you have an older Mac like a Powerbook, iBook, G4 or G5, you need the Arduino for PPC. If you have a newer Mac like an MacBook, MacBook Pro, or Mac Pro, you need the Intel version.

**Download**: Arduino 0006 for PPC or Arduino 0006 for Intel

Restore
November 16, 2006, at 04:48 PM by David A. Mellis -
Added lines 24-29:

(:include HowtoExample:)

Select the serial device of the Arduino board from the Tools | Serial Port menu. On the Mac, this should be something with `/dev/tty.usbserial` in it.



Restore
November 04, 2006, at 07:01 AM by David A. Mellis -
Deleted lines 24-25:

(:include HowtoReferences:)

Restore
November 04, 2006, at 06:53 AM by David A. Mellis -
Changed lines 1-31 from:

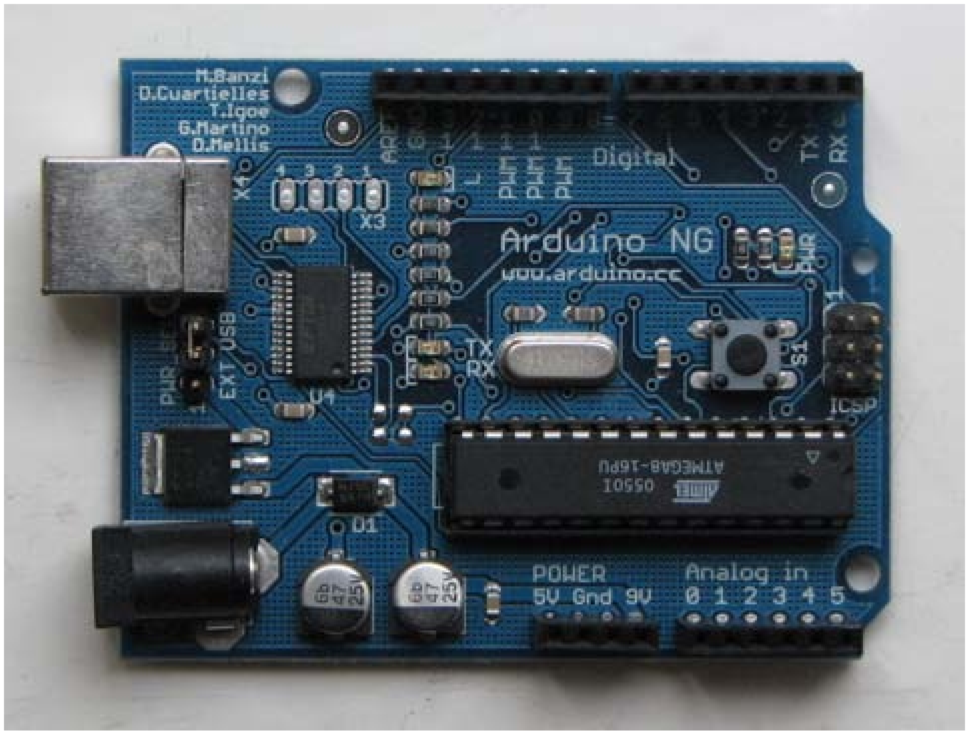
## Arduino Howto

These are the steps you need to follow in order to be up and running:

1. Get an Arduino board
2. Download the Arduino environment
3. Install the USB drivers
4. Connect the board
5. Upload a program

### 1 | Get an Arduino board

The Arduino i/o board is a simple circuit featuring the ATmega8 processor from Atmel. The board is composed of a printed circuit board (PCB) and electronic parts.



There are a few ways to get an Arduino board:

- **buy a ready made board**. See how you can buy a board or just the PCB.
  - European distributor
  - US distributor
- **build your own board**. If you want you can build your own PCB just by downloading the CAD files from the Hardware page. Extract the .brd file and send it to a PCB manufacturer. Be aware that manufacturing a single pcb will be very expensive. It's better to get together with other people and make 20 or 30 at a time. Since you get the full CAD files you can make your own customised version of Arduino. if you make modifications or fix bugs please send us your changes!
  - **purchase parts**. purchase the parts from any electronics store. The Serial version in particular has been designed to use the most basic parts that can be found anywhere in the world. The USB version on the other hand requires some advanced soldering skills because of the FTDI chip that is an smd part. Here is a list? of parts for the serial board.
  - **assemble the board**. We put together a step by step guide on how to build an arduino board. *Newbies: never soldered before? afraid of trashing thousands of boards before getting one properly soldered? fear not :) learn to master the art of soldering.*
  - **program the bootloader**. In order for the development environment to be able to program the chip, this has to be programmed with a piece of code called *bootloader*. See the bootloader page on how to program it on your chip.

### 2 | Download the Arduino environment

To program the Arduino board you need the Arduino environment.

**Download Arduino**: From the software page.

to:

## How To Get Arduino Running on Mac OS X (10.3.9 or later)

(:include HowtoSteps:)

(:include HowtoGet:)

(:include HowtoDownload:)

Deleted lines 12-13:

For more information, see the guide to the Arduino environment.

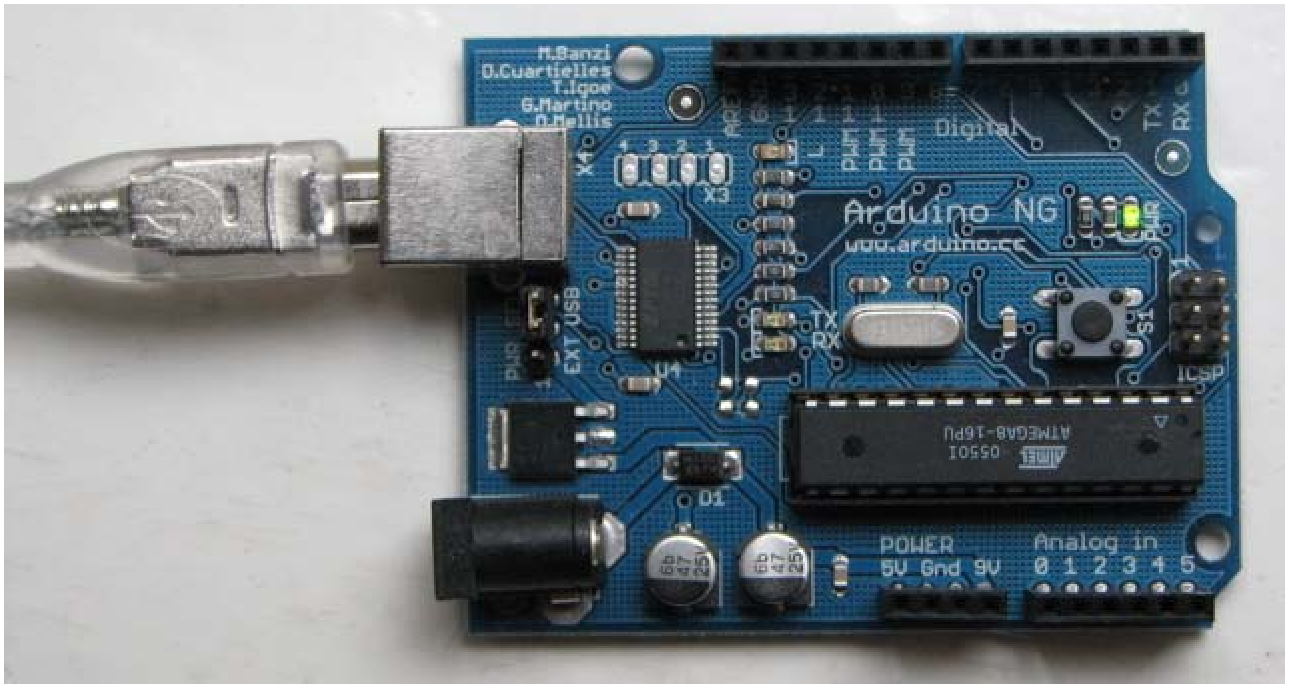
Changed lines 22-66 from:

### 4 | Connect the board

If you're using a serial board, power the board with an external power supply (6 to 25 volts DC, with the core of the connector positive). Connect the board to a serial port on your computer.

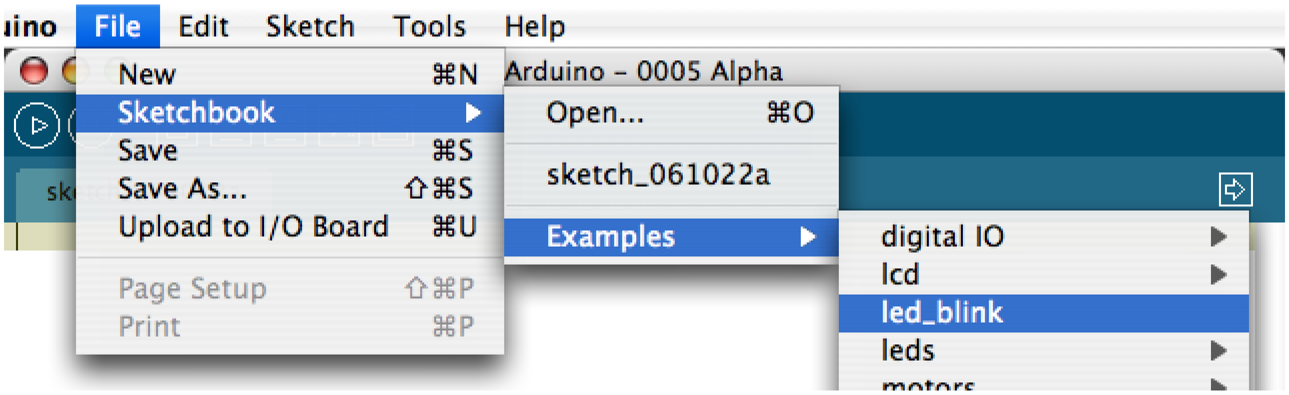
On the USB boards, the power source is selected by the jumper between the USB and power plugs. To power the board from the USB port (good for controlling low power devices like LEDs), place the jumper on the two pins closest to the USB plug. To power the board from an external power supply (needed for motors and other high current devices), place the jumper on the two pins closest to the power plug. Either way, connect the board to a USB port on your computer.

The power LED should go on.

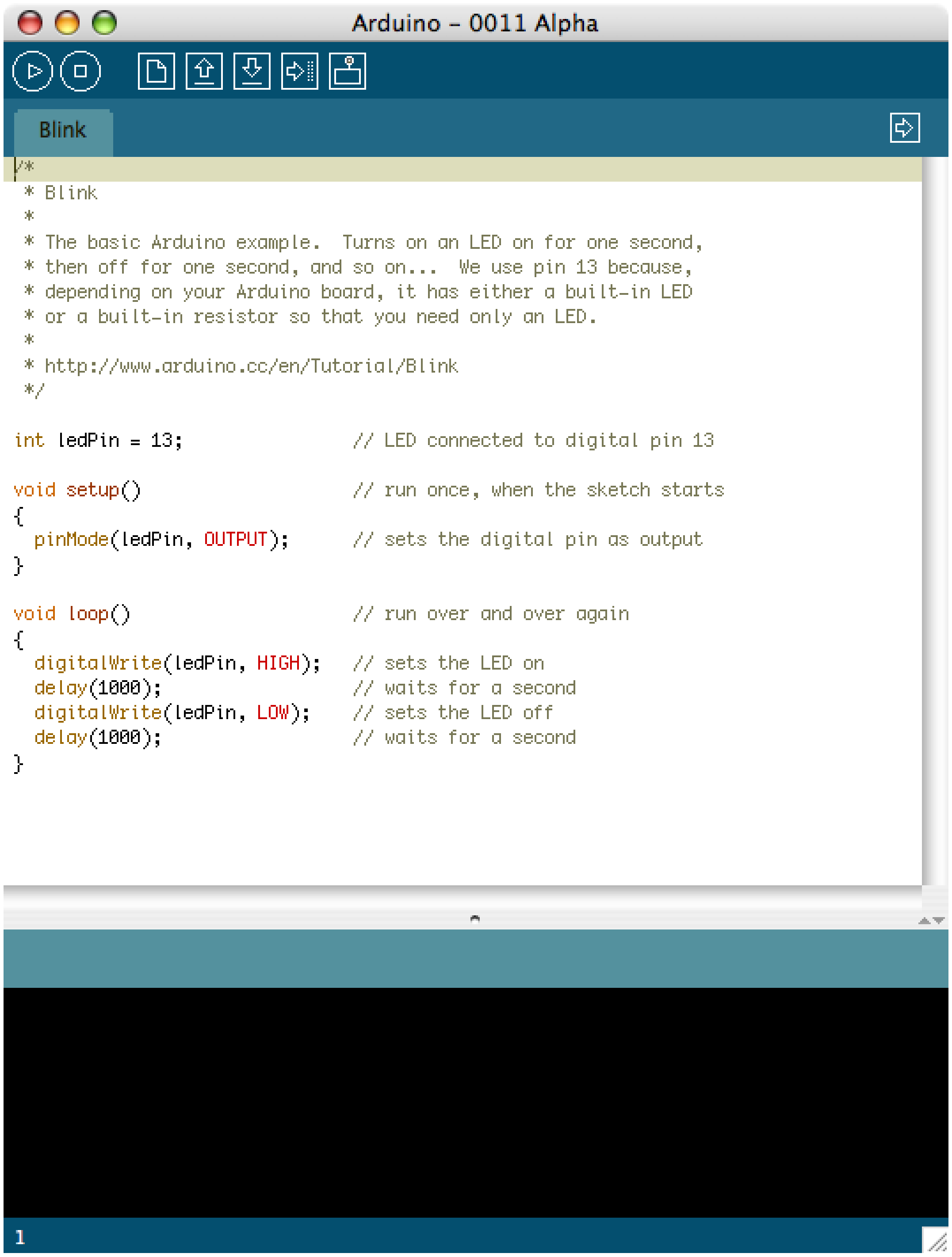


### 5 | Upload a program

Open the LED blink example sketch: File > Sketchbook > Examples > led_blink.

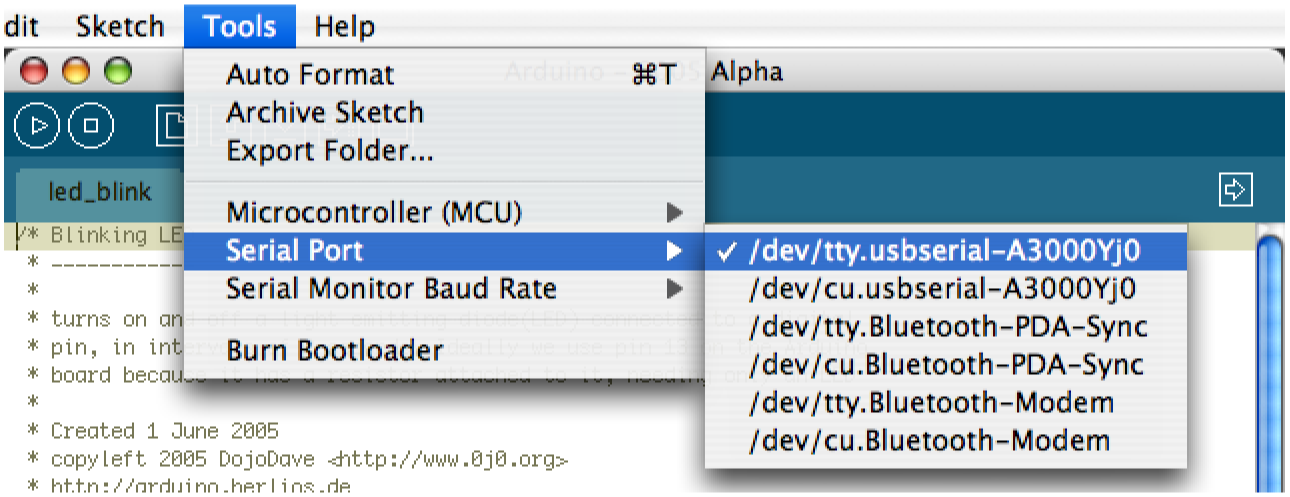


Here's what the code for the LED blink example looks like.



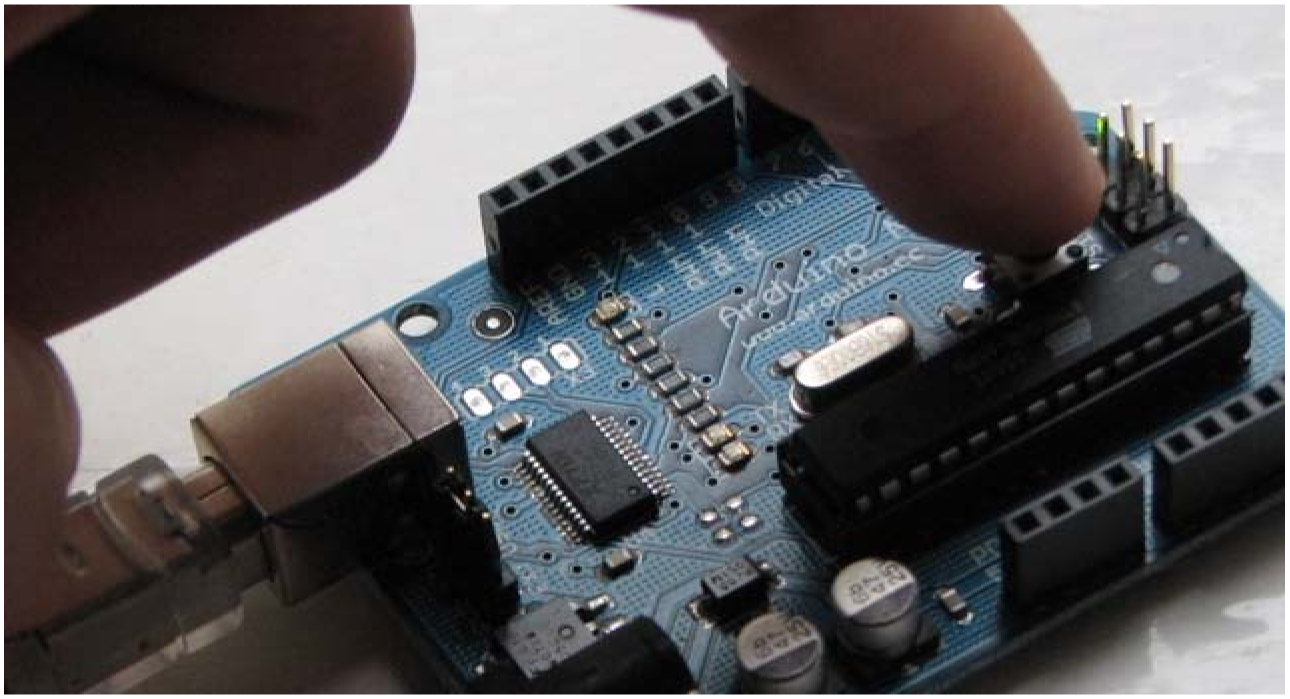
```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

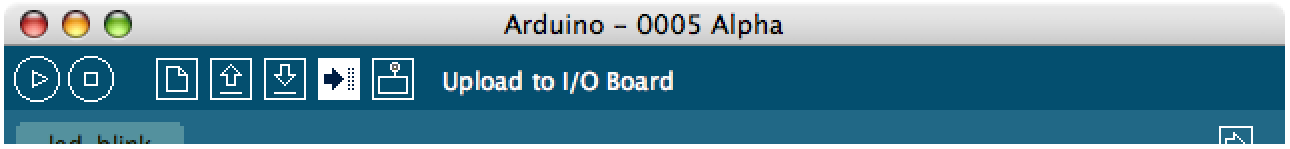
Select the serial device of the Arduino board from the Tools | Serial Port menu. On the Mac, this should be something like /dev/cu.usbserial-1B1 for a USB board, or something like /dev/cu.USA19QW1b1P1.1 if using a Keyspan adapter with a serial board (other USB-to-serial adapters use different names).



```
/* Blinking LE
 * ----------
 *
 * turns on an
 * pin, in int
 * board becau
 *
 * Created 1 June 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
```

Push the reset button on the board then click the *Upload* button in the IDE. Wait a few seconds. If successful, the message "Done uploading." will appear in the status bar.





If the Arduino board doesn't show up in the Tools | Serial Port menu, or you get an error while uploading, please see the troubleshooting suggestions.

A few seconds after the upload finishes, you should see the amber (yellow) LED on the board start to blink.

## Learn More

- Read about the Arduino Environment
- Learn about the parts of the Arduino board
- See the tutorials for some example programs. (There are also some examples available in the examples directory inside the arduino directory.)
- Look up specific Arduino functions and syntax in the reference
- The Arduino programming language is compatible with the Wiring language allowing porting applications from the Wiring board to Arduino. Please note the differences between the Wiring and Processing languages.
- If you're having problems, check the FAQ.
- If you don't find a solution there, try posting in the forums.

to:

(:include HowtoConnect:)

(:include HowtoUpload:)

(:include HowtoReferences:)

Restore
November 04, 2006, at 06:42 AM by David A. Mellis -
Added lines 1-91:

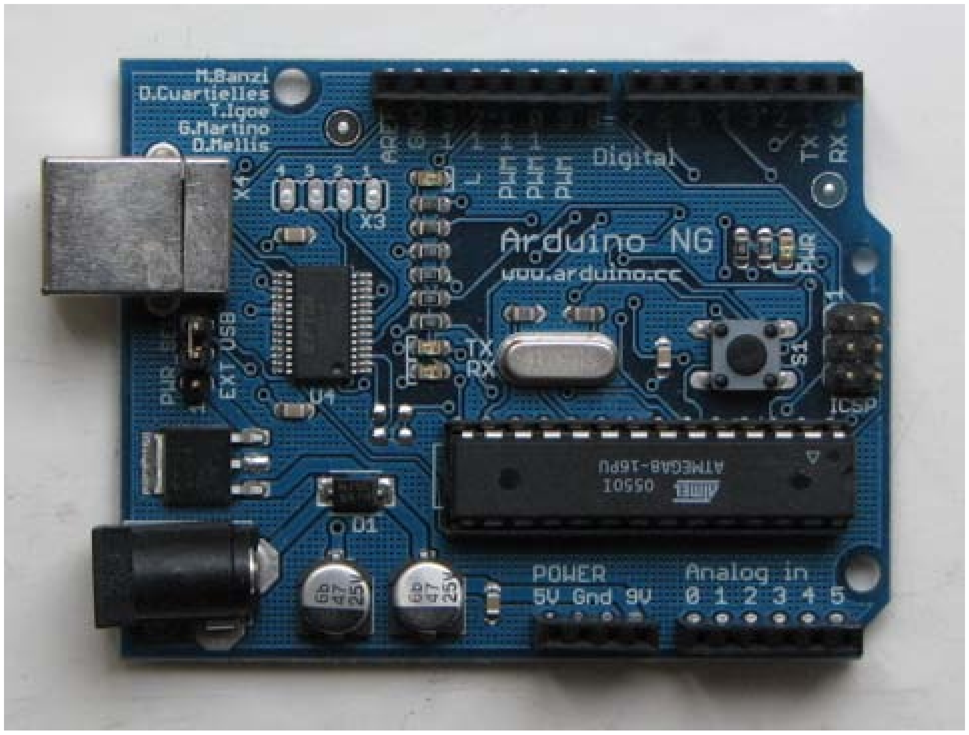
## Arduino Howto

These are the steps you need to follow in order to be up and running:

1. Get an Arduino board
2. Download the Arduino environment
3. Install the USB drivers
4. Connect the board
5. Upload a program

### 1 | Get an Arduino board

The Arduino i/o board is a simple circuit featuring the ATmega8 processor from Atmel. The board is composed of a printed circuit board (PCB) and electronic parts.



There are a few ways to get an Arduino board:

- **buy a ready made board**. See how you can buy a board or just the PCB.
    - European distributor
    - US distributor
- **build your own board**. If you want you can build your own PCB just by downloading the CAD files from the Hardware page. Extract the .brd file and send it to a PCB manufacturer. Be aware that manufacturing a single pcb will be very expensive. It's better to get together with other people and make 20 or 30 at a time. Since you get the full CAD files you can make your own customised version of Arduino. if you make modifications or fix bugs please send us your changes!
    - **purchase parts**. purchase the parts from any electronics store. The Serial version in particular has been designed to use the most basic parts that can be found anywhere in the world. The USB version on the other hand requires some advanced soldering skills because of the FTDI chip that is an smd part. Here is a list? of parts for the serial board.
    - **assemble the board**. We put together a step by step guide on how to build an arduino board. *Newbies: never soldered before? afraid of trashing thousands of boards before getting one properly soldered? fear not :)*

*learn to master the art of soldering.*

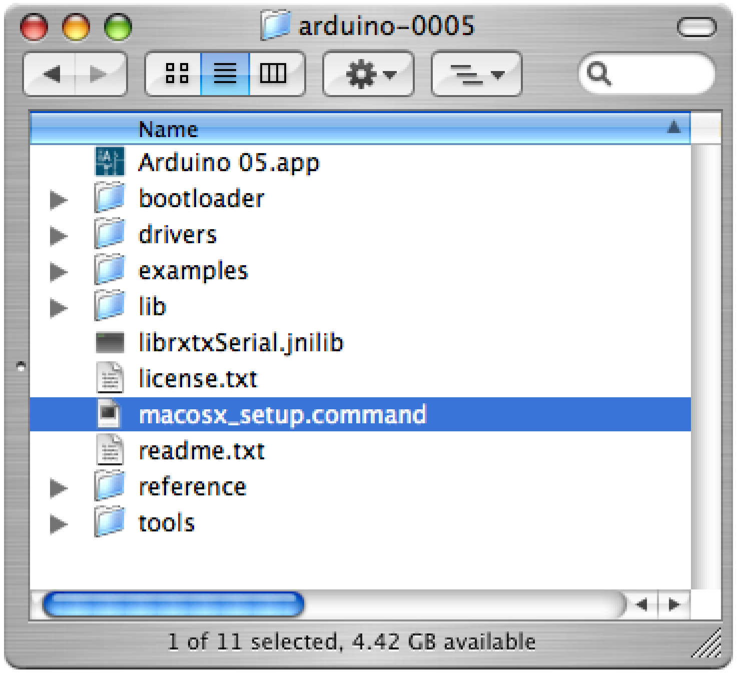
- **program the bootloader**. In order for the development environment to be able to program the chip, this has to be programmed with a piece of code called *bootloader*. See the bootloader page on how to program it on your chip.

## 2 | Download the Arduino environment

To program the Arduino board you need the Arduino environment.

**Download Arduino**: From the software page.

After downloading the IDE, run the `macosx_setup.command`. It corrects permission on a few files for use with the serial port and will prompt you for your password. You may need to reboot after running this script.



arduino-0005

Name

- Arduino 05.app
- bootloader
- drivers
- examples
- lib
- librxtxSerial.jnilib
- license.txt
- macosx_setup.command
- readme.txt
- reference
- tools

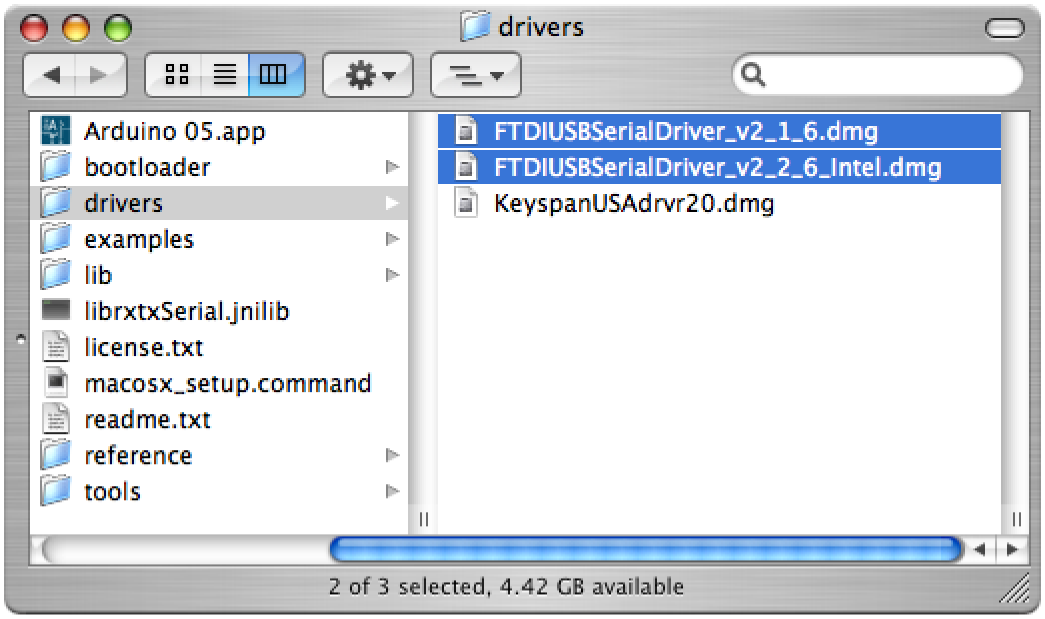
1 of 11 selected, 4.42 GB available

For more information, see the guide to the Arduino environment.

## 3 | Install the USB drivers

If you are using a USB Arduino, you will need to install the drivers for the FTDI chip on the board. These can be found in the drivers directory of the Arduino distribution.

On the Mac, mount the `FTDIUSBSerialDriver_v2_1_6.dmg` (on PPC machines) or the `FTDIUSBSerialDriver_v2_2_6_Intel.dmg` (on Intel machines) disk image and run the included `FTDIUSBSerialDriver.pkg`.



drivers

- Arduino 05.app
- bootloader
- drivers
- examples
- lib
- librxtxSerial.jnilib
- license.txt
- macosx_setup.command
- readme.txt
- reference
- tools

- FTDIUSBSerialDriver_v2_1_6.dmg
- FTDIUSBSerialDriver_v2_2_6_Intel.dmg
- KeyspanUSAdrvr20.dmg

2 of 3 selected, 4.42 GB available

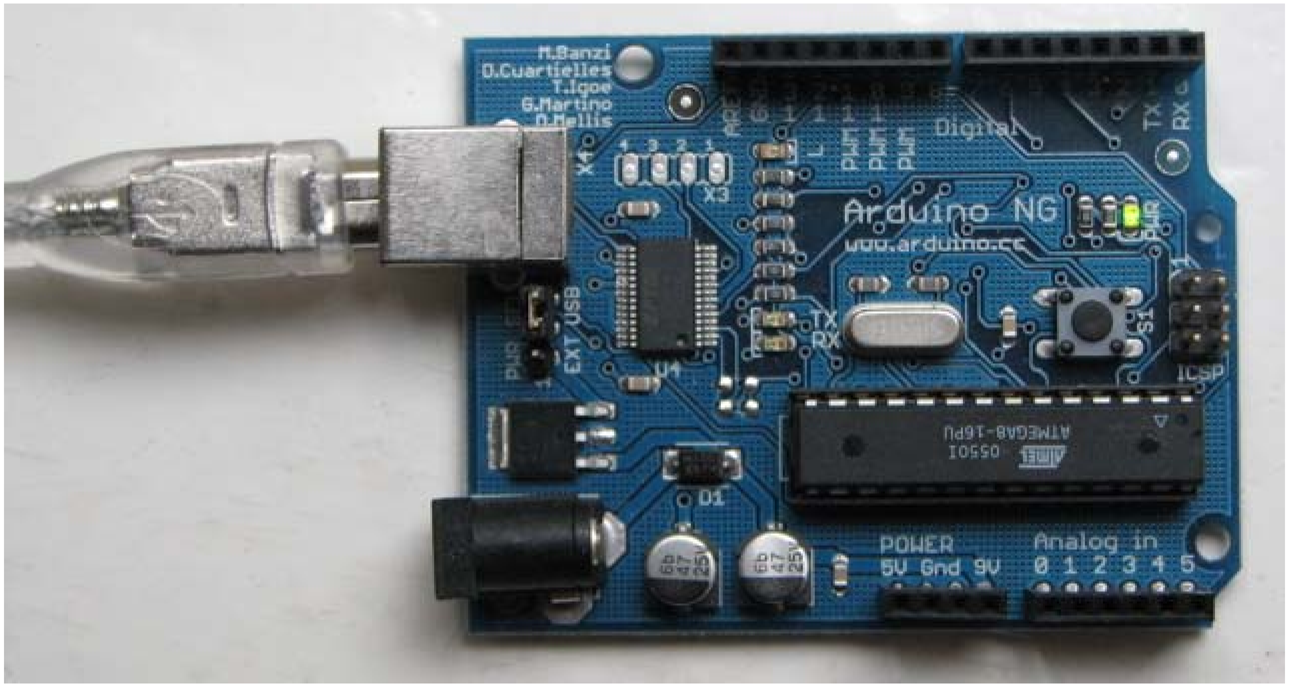
The latest version of the drivers can be found on the FTDI website.

### 4 | Connect the board

If you're using a serial board, power the board with an external power supply (6 to 25 volts DC, with the core of the connector positive). Connect the board to a serial port on your computer.

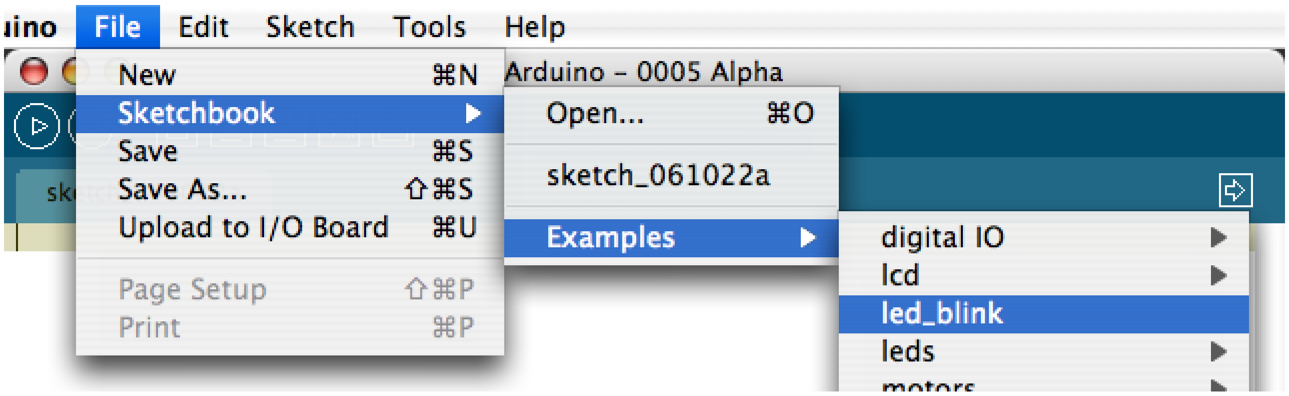
On the USB boards, the power source is selected by the jumper between the USB and power plugs. To power the board from the USB port (good for controlling low power devices like LEDs), place the jumper on the two pins closest to the USB plug. To power the board from an external power supply (needed for motors and other high current devices), place the jumper on the two pins closest to the power plug. Either way, connect the board to a USB port on your computer.

The power LED should go on.

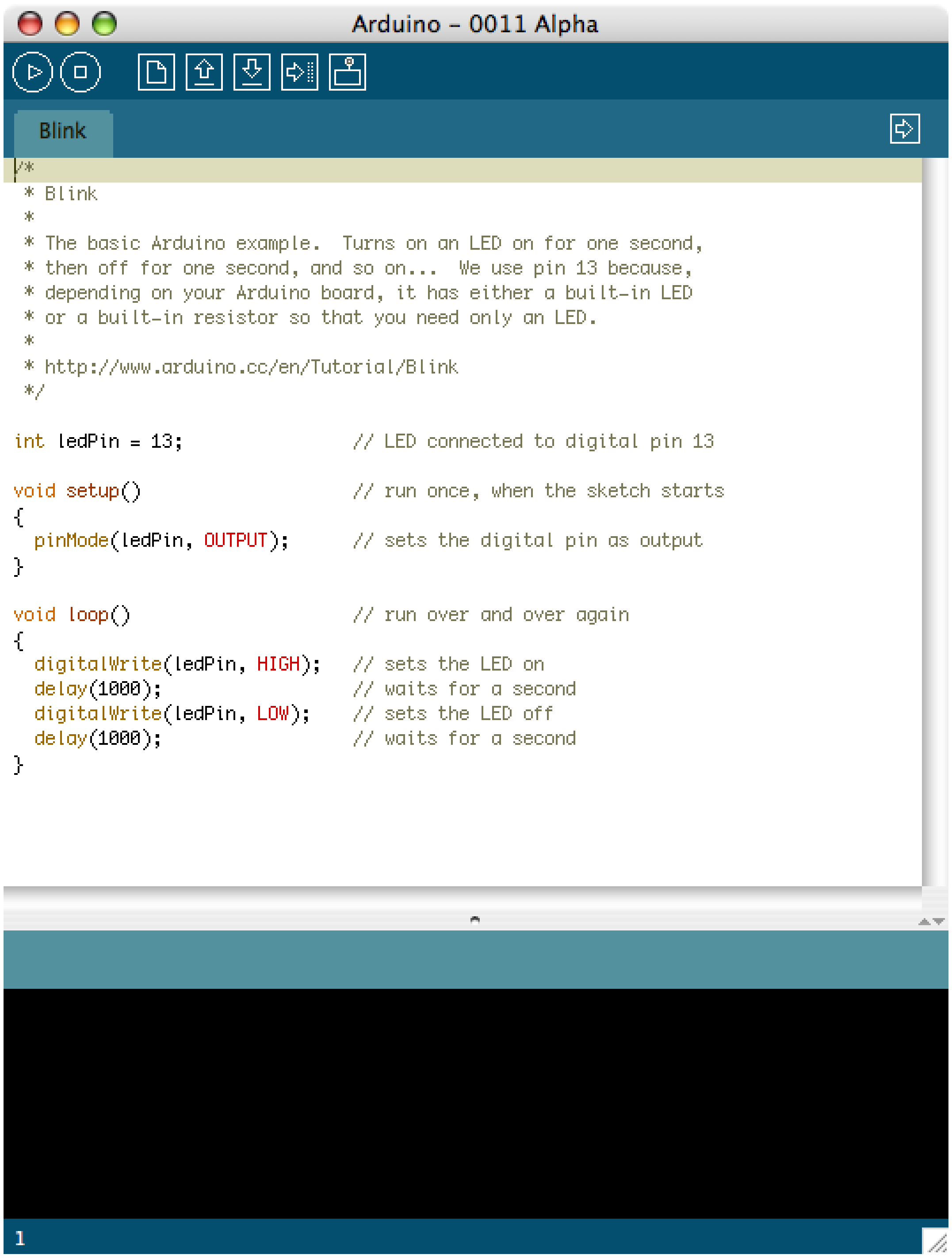


### 5 | Upload a program

Open the LED blink example sketch: File > Sketchbook > Examples > led_blink.

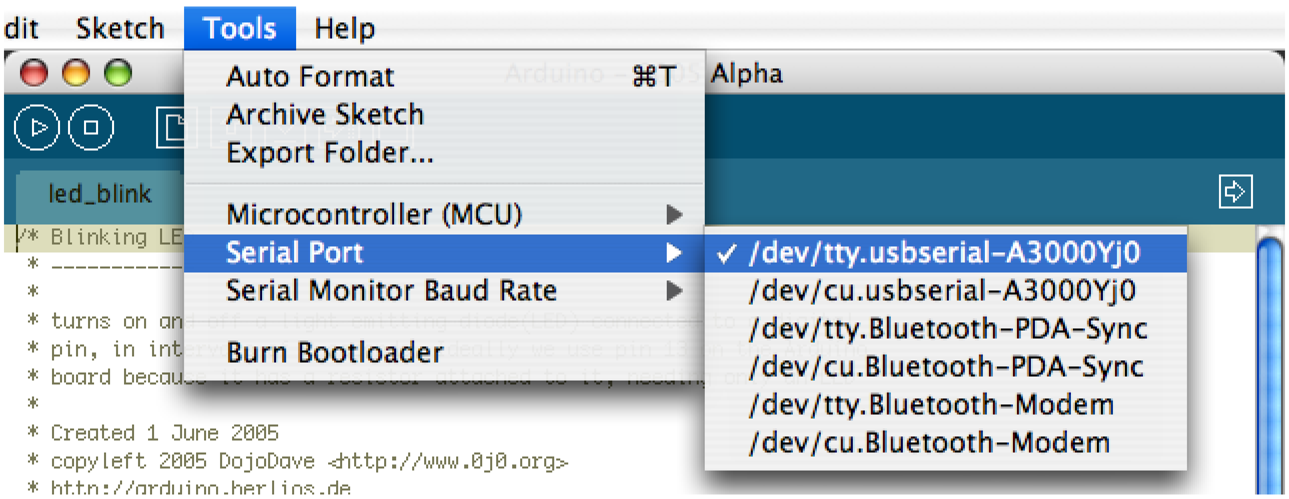


Here's what the code for the LED blink example looks like.



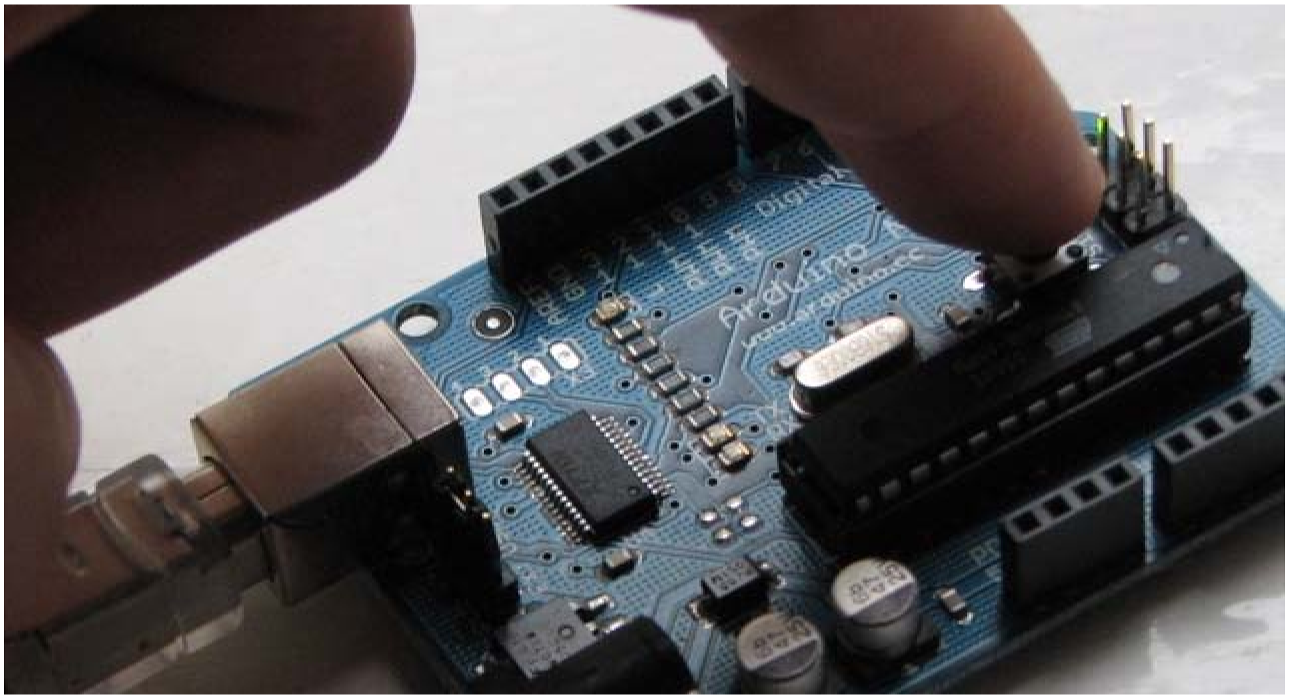
```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

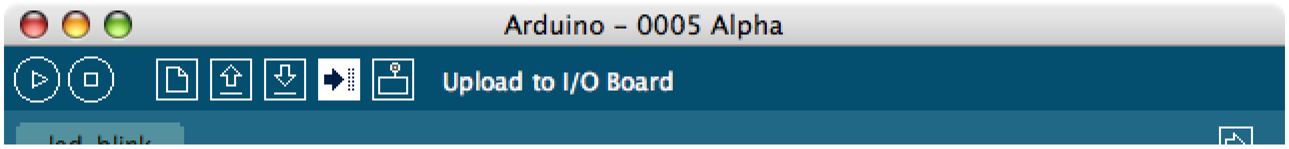
Select the serial device of the Arduino board from the Tools | Serial Port menu. On the Mac, this should be something like /dev/cu.usbserial-1B1 for a USB board, or something like /dev/cu.USA19QW1b1P1.1 if using a Keyspan adapter with a serial board (other USB-to-serial adapters use different names).



```
/* Blinking LE
 * ------------
 *
 * turns on an
 * pin, in int
 * board becau
 *
 * Created 1 June 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
```

Push the reset button on the board then click the *Upload* button in the IDE. Wait a few seconds. If successful, the message "Done uploading." will appear in the status bar.





If the Arduino board doesn't show up in the Tools | Serial Port menu, or you get an error while uploading, please see the troubleshooting suggestions.

A few seconds after the upload finishes, you should see the amber (yellow) LED on the board start to blink.

## Learn More

- Read about the Arduino Environment
- Learn about the parts of the Arduino board
- See the tutorials for some example programs. (There are also some examples available in the examples directory inside the arduino directory.)
- Look up specific Arduino functions and syntax in the reference
- The Arduino programming language is compatible with the Wiring language allowing porting applications from the Wiring board to Arduino. Please note the differences between the Wiring and Processing languages.
- If you're having problems, check the FAQ.
- If you don't find a solution there, try posting in the forums.

# How To Get Arduino Running on Mac OS X (10.3.9 or later)

*This document explains how to connect your Arduino board to the computer and upload your first sketch.*

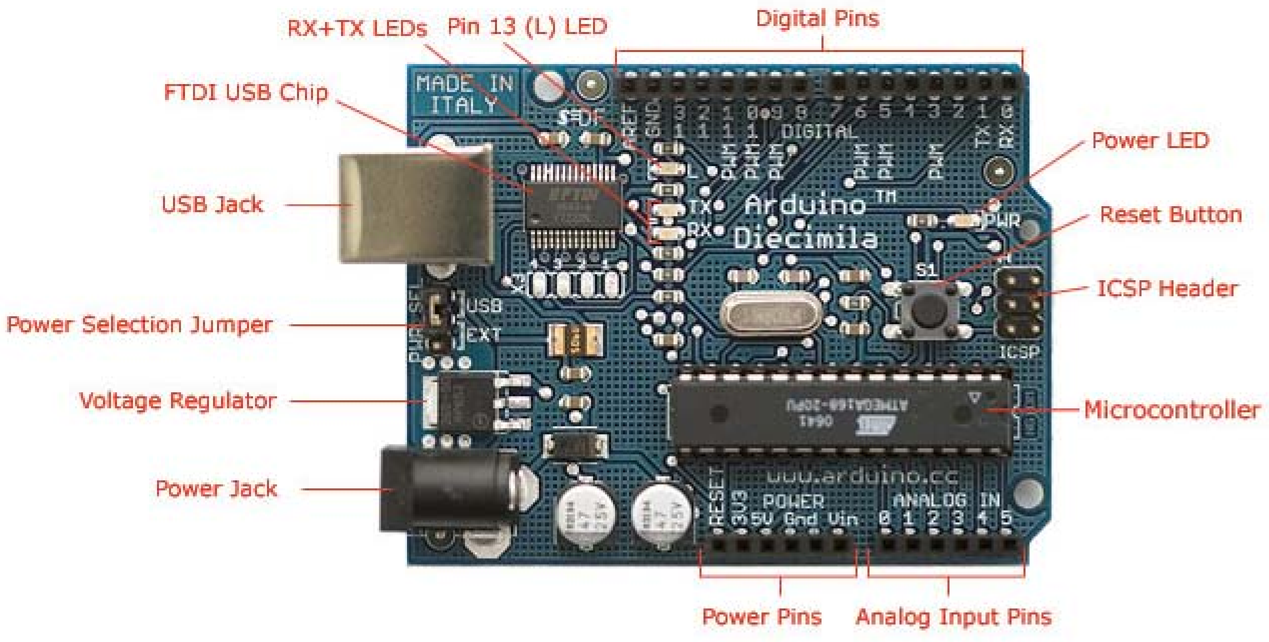
These are the steps that we'll go through:

1. Get an Arduino board and cable
2. Download the Arduino environment
3. Install the USB drivers
4. Connect the board
5. Connect an LED
6. Run the Arduino environment
7. Upload a program
8. Look for the blinking LED
9. Learn to use Arduino

## 1 | Get an Arduino board and cable

In this tutorial, we assume you're using an Arduino Diecimila. If you have another board, read the corresponding page in this getting started guide.

The Arduino Diecimila is a simple board that contains everything you need to start working with electronics and microcontroller programming. This diagram illustrates the major components of the board.



*Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.*

You also need a standard USB cable (A plug to B plug): the kind you would connect to a USB printer, for example.

## 2 | Download the Arduino environment

To program the Arduino board you need the Arduino environment.

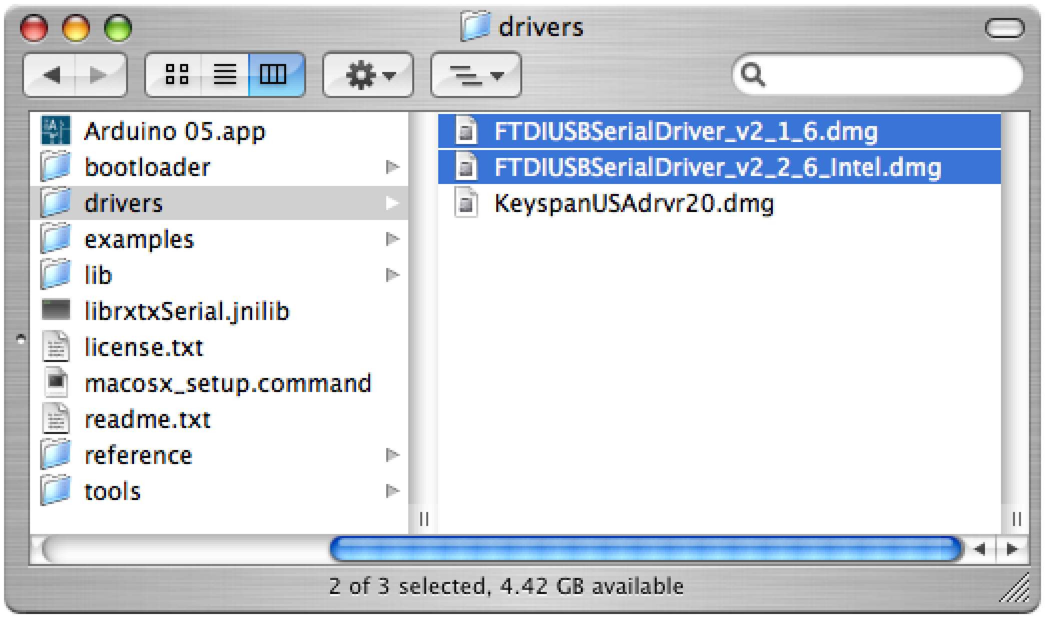
**Download**: the latest version from the download page.

When the download 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.

## 3 | Install the USB drivers

If you are using a USB Arduino, you will need to install the drivers for the FTDI chip on the board. These can be found in the **drivers** directory of the Arduino distribution.

If you have an older Mac like a Powerbook, iBook, G4 or G5, you should use the the PPC drivers: `FTDIUSBSerialDriver_v2_1_9.dmg`. If you have a newer Mac like an MacBook, MacBook Pro, or Mac Pro, you need the Intel drivers: `FTDIUSBSerialDriver_v2_2_9_Intel.dmg`. Double-click to mount the disk image and run the included `FTDIUSBSerialDriver.pkg`.

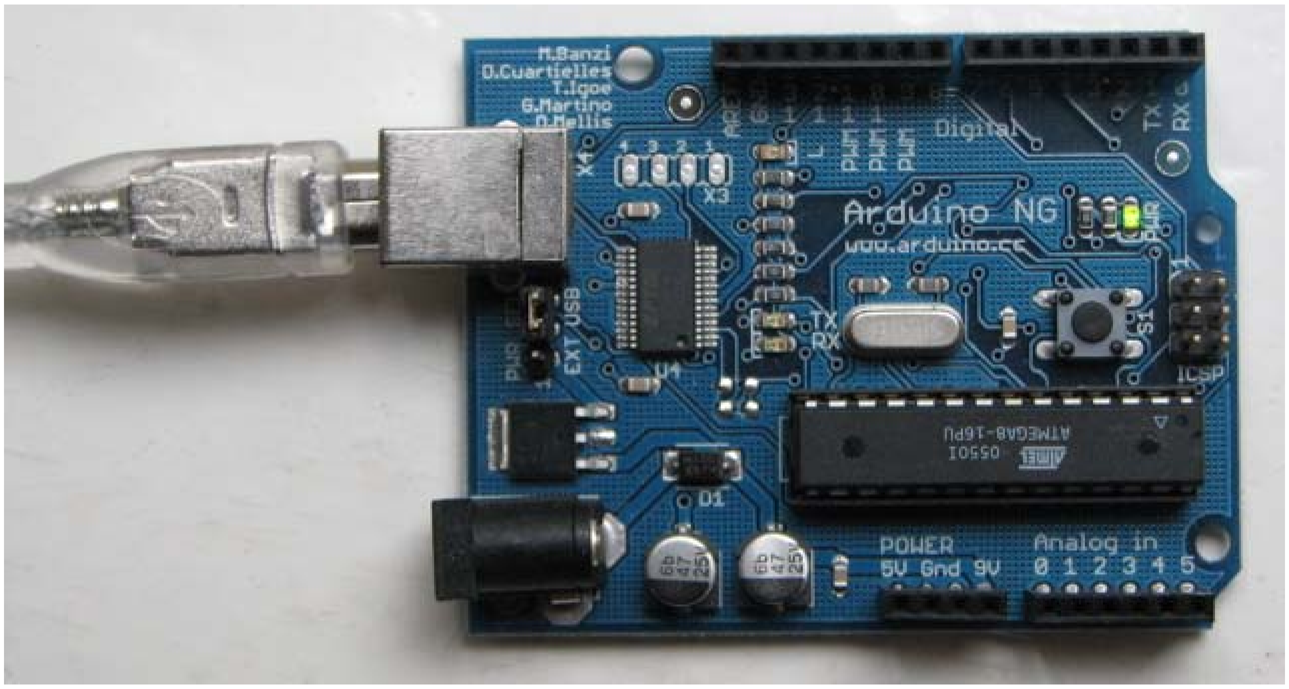


(The latest version of the drivers can be found on the FTDI website.)

## 4 | Connect the board

The power source is selected by the jumper between the USB and power plugs. To power the board from the USB port (good for controlling low power devices like LEDs), place the jumper on the two pins closest to the USB plug. To power the board from an external power supply (6-12V), place the jumper on the two pins closest to the power plug. Either way, connect the board to a USB port on your computer.

The power LED should go on.



## 5 | Connect an LED (if you're using an older board)

The first sketch you will upload to the Arduino board blinks an LED. The Arduino Diecimila (and the original Arduino NG) has a built-in resistor on pin 13. On Arduino NG Rev. C and pre-NG Arduino boards, however, pin 13 does not have a built-in LED. On these boards, you'll need to connect the positive (longer) leg of an LED to pin 13 and the negative (shorter) leg to ground (marked "GND"). The LED will typically be flat on the side with the negative leg. Normally, you also need to use a resistor with the LED, but these boards have a resistor built-in on pin 13.

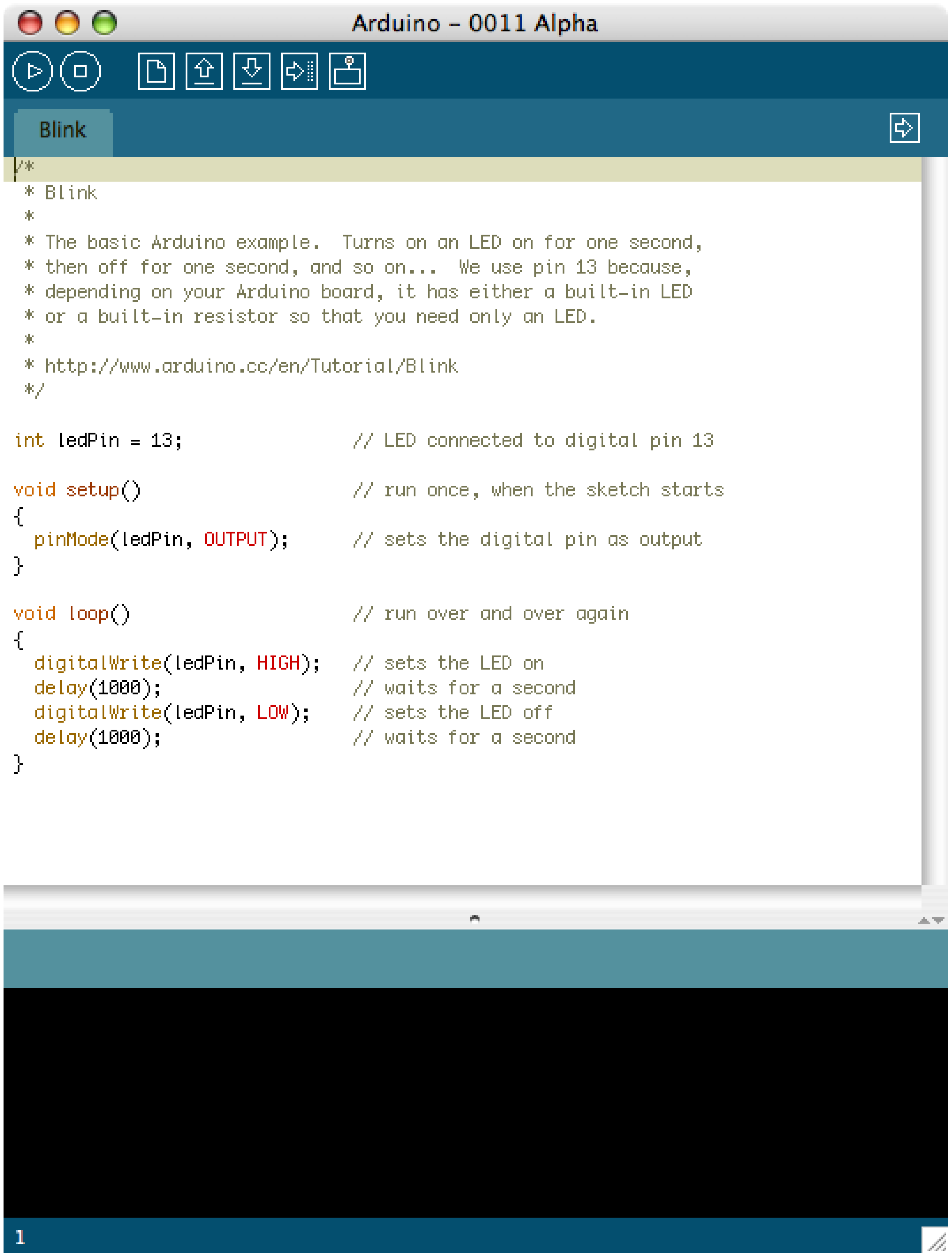
## 6 | Run the Arduino environment

Open the Arduino folder and double-click the Arduino application.

## 7 | Upload a program

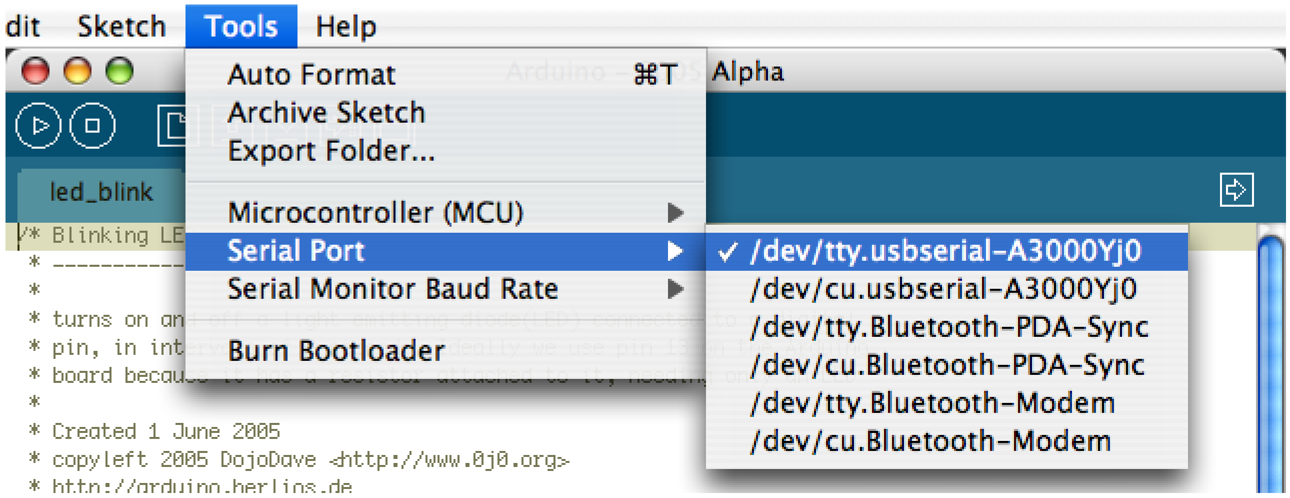
Open the LED blink example sketch: **File > Sketchbook > Examples > Digital > Blink**.

Here's what the code for the LED blink example looks like.



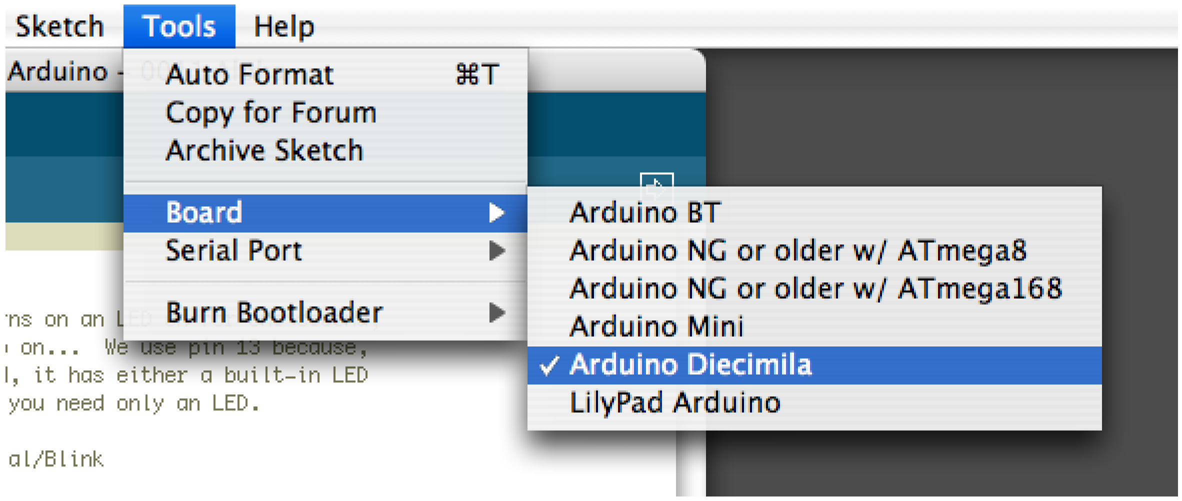
```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

Select the serial device of the Arduino board from the Tools | Serial Port menu. On the Mac, this should be something with `/dev/tty.usbserial` in it.



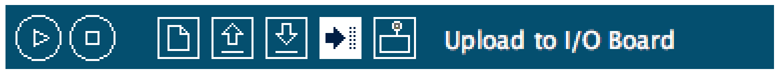
```
/* Blinking LE
 * -----------
 *
 * turns on an
 * pin, in int
 * board becau
 *
 * Created 1 June 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
```

Make sure that "Arduino Diecimila" is selected in the **Tools > Board** menu.



```
ns on an L
 on...  We use pin 13 because,
l, it has either a built-in LED
you need only an LED.

al/Blink
```

Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar. (*Note:* If you have an Arduino Mini, NG, or other board, you'll need to physically present the reset button on the board immediately before pressing the upload button.)



## 8 | Look for the blinking LED

A few seconds after the upload finishes, you should see the amber (yellow) LED on the board start to blink. If it does, congratulations! You've gotten Arduino up-and-running.

If you have problems, please see the troubleshooting suggestions.

## 9 | Learn to use Arduino

- Tutorials: try these example programs.
- Reference: read the reference for the Arduino language.

# Arduino

search

## Login to Arduino

| | |
|---|---|
| Username: | |
| Password: | |
| Keep me logged in: | ✓ |

Login

## Guide.ArduinoNano History

Hide minor edits - Show changes to markup

June 21, 2008, at 12:36 PM by David A. Mellis -
Added lines 15-16:

Restore
June 21, 2008, at 12:36 PM by David A. Mellis -
Added lines 9-10:

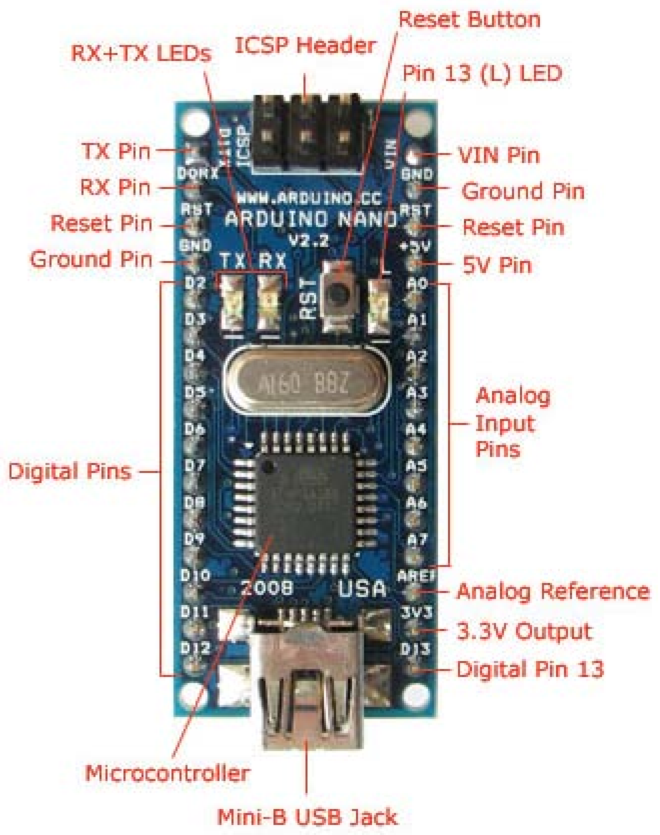
Restore
June 21, 2008, at 12:35 PM by David A. Mellis -
Added lines 7-8:

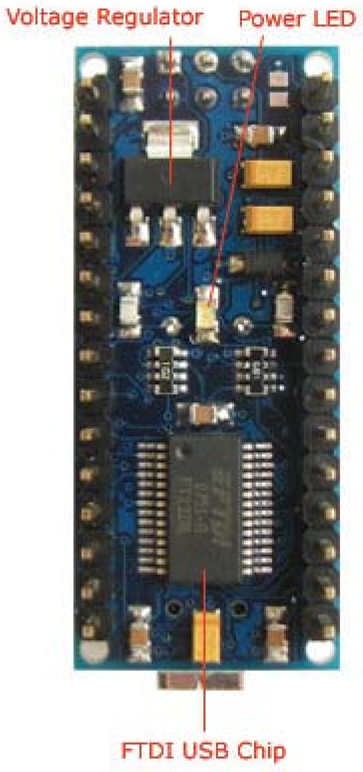
*The various components of the Arduino Nano.*

Added lines 11-12:

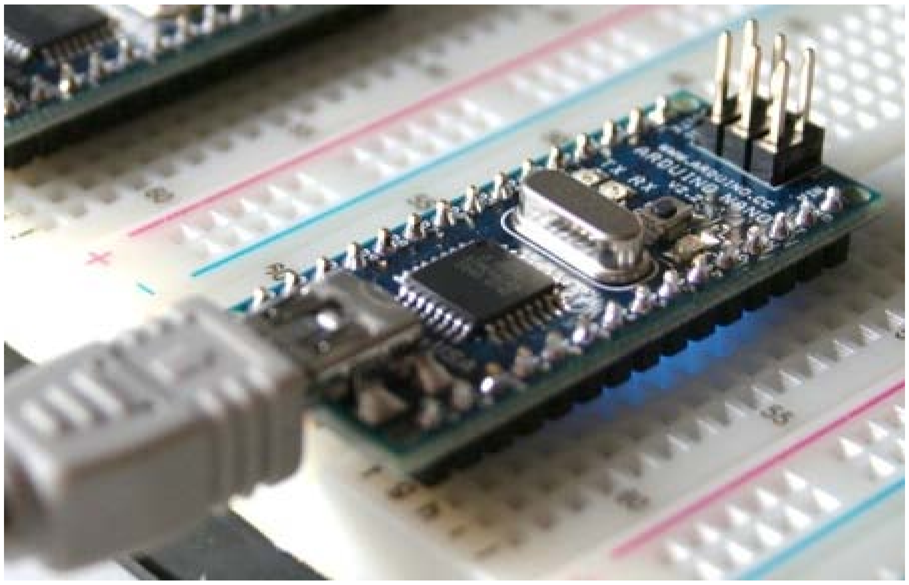
*Connecting the Arduino Nano to a computer with a Mini-B USB cable. Note the blue power LED underneath the board.*

Restore
June 21, 2008, at 12:27 PM by David A. Mellis -
Added lines 5-6:





Restore
June 18, 2008, at 09:34 AM by David A. Mellis -
Added lines 5-6:



Restore
June 18, 2008, at 09:29 AM by David A. Mellis -
Changed lines 7-9 from:

To upload a sketch to the Nano, make sure you have the **Arduino Diecimila** option selected from the **Tools > Board** menu and the correct serial port selected from the **Tools > Serial Port** menu. Then simply press the upload button in the Arduino environment. The board will automatically reset and the sketch will be uploaded, as with the Arduino Diecimila. If you have any problems, see the troubleshooting guide.

to:

To upload a sketch to the Nano, make sure you have the **Arduino Diecimila** option selected from the **Tools > Board** menu and the correct serial port selected from the **Tools > Serial Port** menu. Then simply press the upload button in the Arduino environment. The board will automatically reset and the sketch will be uploaded, as with the Arduino Diecimila. If you have any problems, see the troubleshooting guide.

For more details on the Arduino Nano, see the hardware page.

Restore
June 18, 2008, at 09:28 AM by David A. Mellis -
Added lines 1-7:

## Guide to the Arduino Nano

*This is a preliminary guide and will be updated and expanded in the next few days.*

To connect the Arduino Nano to your computer, you'll need a Mini-B USB cable. This also provides power to the board, as indicated by the blue LED on the bottom.

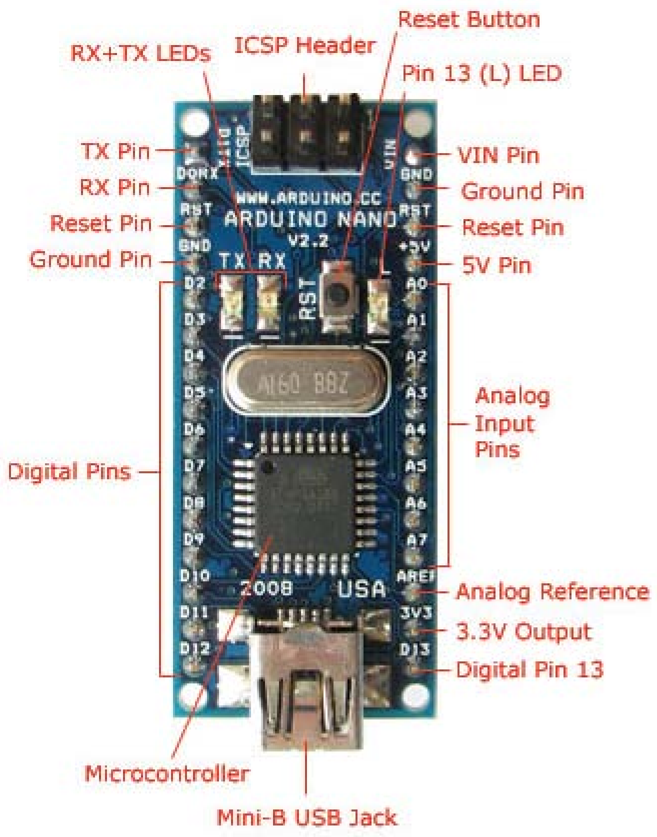
To upload a sketch to the Nano, make sure you have the **Arduino Diecimila** option selected from the **Tools > Board** menu and the correct serial port selected from the **Tools > Serial Port** menu. Then simply press the upload button in the Arduino environment. The board will automatically reset and the sketch will be uploaded, as with the Arduino Diecimila. If you have any problems, see the troubleshooting guide.

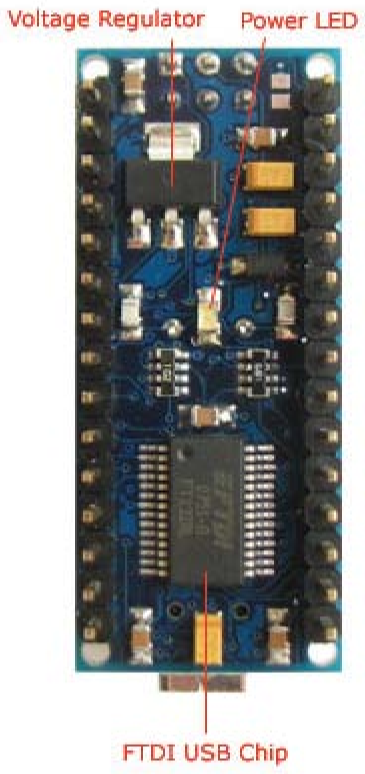
Restore

**Guide** Contents | Introduction | How To: Windows, Mac OS X, Linux; Arduino Nano, Arduino Mini, Arduino BT, LilyPad Arduino; Xbee shield | Troubleshooting | Environment

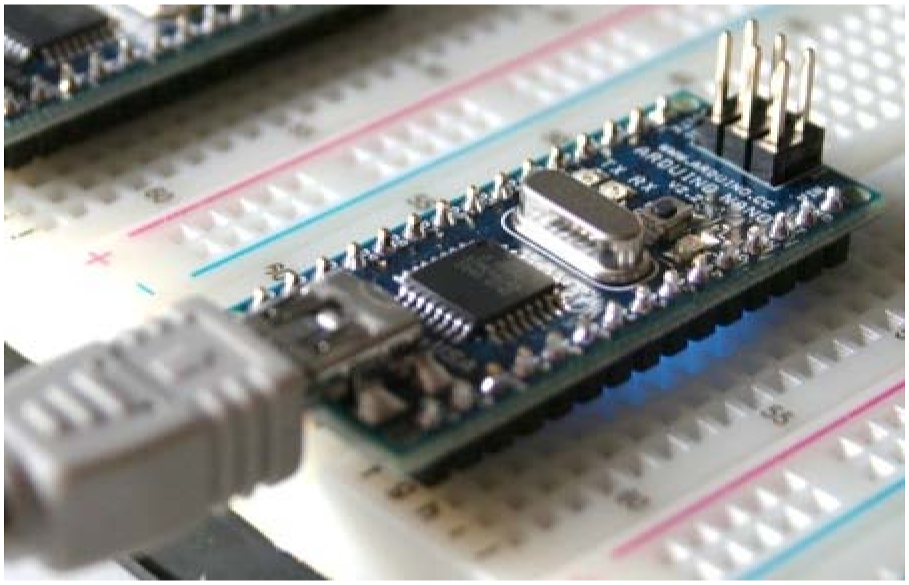
## Guide to the Arduino Nano

*This is a preliminary guide and will be updated and expanded in the next few days.*





*The various components of the Arduino Nano.*



*Connecting the Arduino Nano to a computer with a Mini-B USB cable. Note the blue power LED underneath the board.*

To connect the Arduino Nano to your computer, you'll need a Mini-B USB cable. This also provides power to the board, as indicated by the blue LED on the bottom.

To upload a sketch to the Nano, make sure you have the **Arduino Diecimila** option selected from the **Tools > Board** menu and the correct serial port selected from the **Tools > Serial Port** menu. Then simply press the upload button in the Arduino environment. The board will automatically reset and the sketch will be uploaded, as with the Arduino Diecimila. If you have any problems, see the troubleshooting guide.

For more details on the Arduino Nano, see the hardware page.

# Arduino

search

## Login to Arduino

|  |  |
|---|---|
| Username: |  |
| Password: |  |
| Keep me logged in: | ✓ |

Login

## Guide.ArduinoMini History

Hide minor edits - Show changes to markup

January 25, 2008, at 05:05 PM by David A. Mellis -
Changed lines 3-4 from:

To get started with the Arduino Mini, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux), with the following modifications:

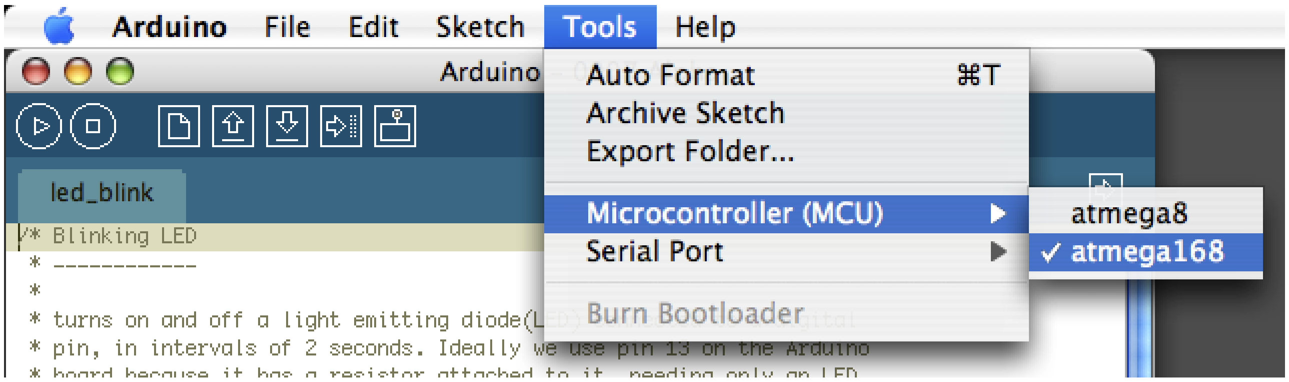
to:

To get started with the Arduino Mini, follow the directions for the regular Arduino on your operating system (Windows, Mac OS X, Linux), with the following modifications:

Deleted lines 65-70:

### Selecting ATmega168 in the Arduino Environment

To tell the Arduino environment that you're using an Arduino Mini, which has a different chip than the regular Arduino, you need to select **ATmega168** from the **Tools | Microcontroller** menu.



```
/* Blinking LED
 * ------------
 *
 * turns on and off a light emitting diode(LED)
 * pin, in intervals of 2 seconds. Ideally we use pin 13 on the Arduino
```

Restore
January 25, 2008, at 05:04 PM by David A. Mellis -
Changed lines 7-8 from:

- You need to select **ATmega168** from the **Tools | Microcontroller** menu of the Arduino environment.

to:

- You need to select **Arduino Mini** from the **Tools | Board** menu of the Arduino environment.

Restore
June 15, 2007, at 04:52 PM by David A. Mellis -
Changed lines 11-20 from:

The Arduino Mini has a more powerful chip than the regular Arduino board (an ATmega168 instead of an ATmega8), meaning:

- Your sketches can be twice as big (14 KB instead of 7 KB).
- There are three extra PWM outputs (on pins 3, 5, and 6) in addition to the three on regular Arduino boards (pins 9, 10, and 11).
- There are two extra analog inputs (8 total). Four of these, however, are not connected to the legs that come on the Arduino Mini, requiring you to solder wires to their holes to use them. Two of these unconnected pins are also used by the Wire library (I2C), meaning that its use will require soldering as well.

The Arduino Mini is more **fragile and easy to break** than a regular Arduino board.

to:

The microcontroller (an ATmega168) on the Arduino Mini is a physically smaller version of the chip on the USB Arduino boards, with the following small difference:

- There are two extra analog inputs on the Mini (8 total). Four of these, however, are not connected to the legs that come on the Arduino Mini, requiring you to solder wires to their holes to use them. Two of these unconnected pins are also used by the Wire library (I2C), meaning that its use will require soldering as well.

Also, the Arduino Mini is more **fragile and easy to break** than a regular Arduino board.

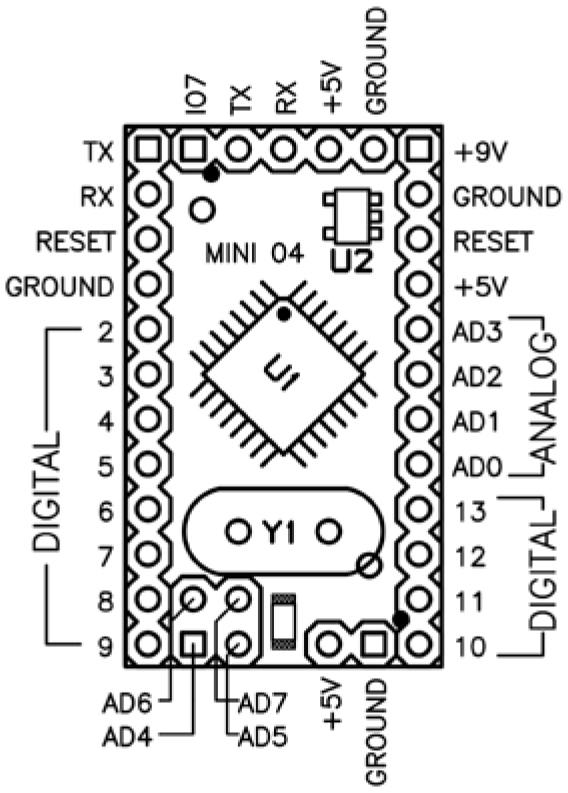
Restore
June 15, 2007, at 03:07 PM by David A. Mellis -
Added lines 30-32:

(:table width=95% border=0 cellpadding=5 cellspacing=0:) (:cell width=50%:)

Added lines 35-44:

*Mini 03 pinout* (compatible with earlier revisions)

(:cell width=50%:)

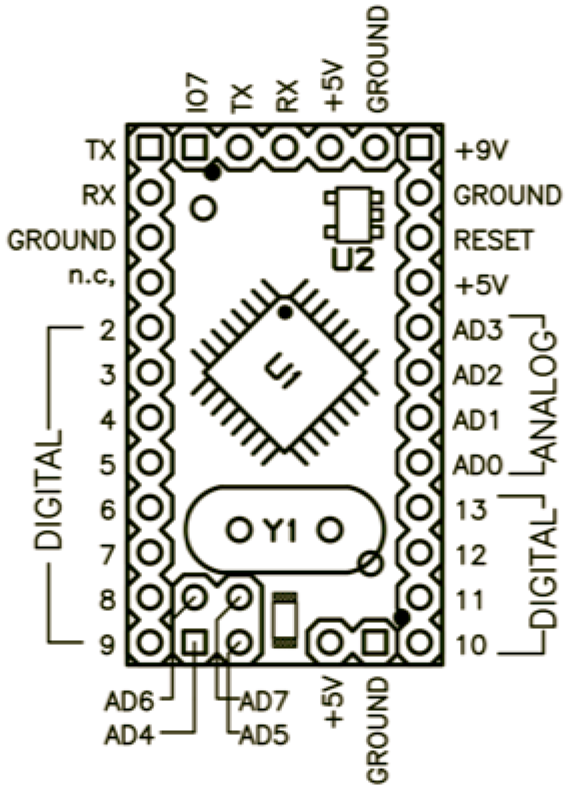


*Mini 04 pinout* (the ground on the left has moved down one pin)

(:tableend:)

Restore
April 22, 2007, at 01:06 PM by David A. Mellis -
Added lines 28-31:

Here's a diagram of the pin layout of the Arduino Mini:



Restore
January 07, 2007, at 07:01 AM by David A. Mellis -
Changed lines 23-24 from:

- You can't remove the ATmega168, so if you kill it, you need a new Arduino Mini.

to:

- You can't remove the ATmega168, so if you kill it, you need a new Mini.

Restore
January 07, 2007, at 07:00 AM by David A. Mellis -
Changed lines 5-6 from:

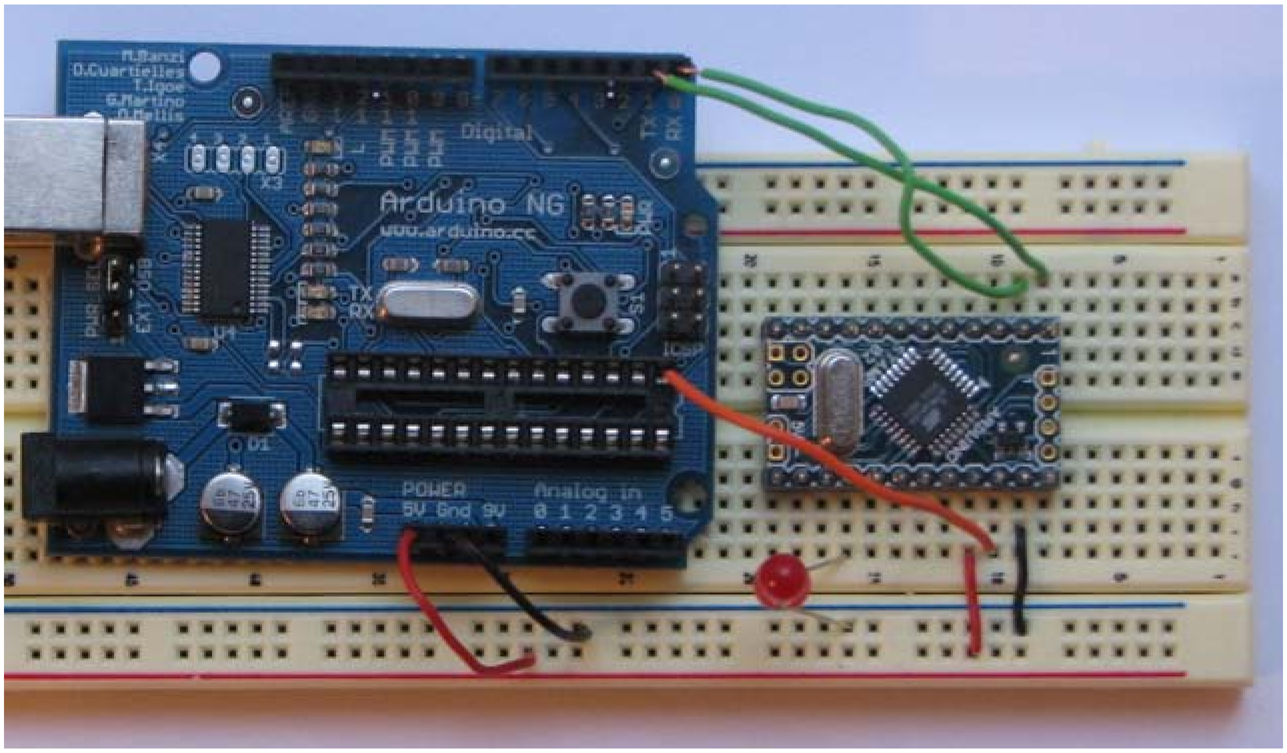
- Connecting the Arduino Mini is a bit more complicated than a regular Arduino board (see below for instructions and photos).

to:

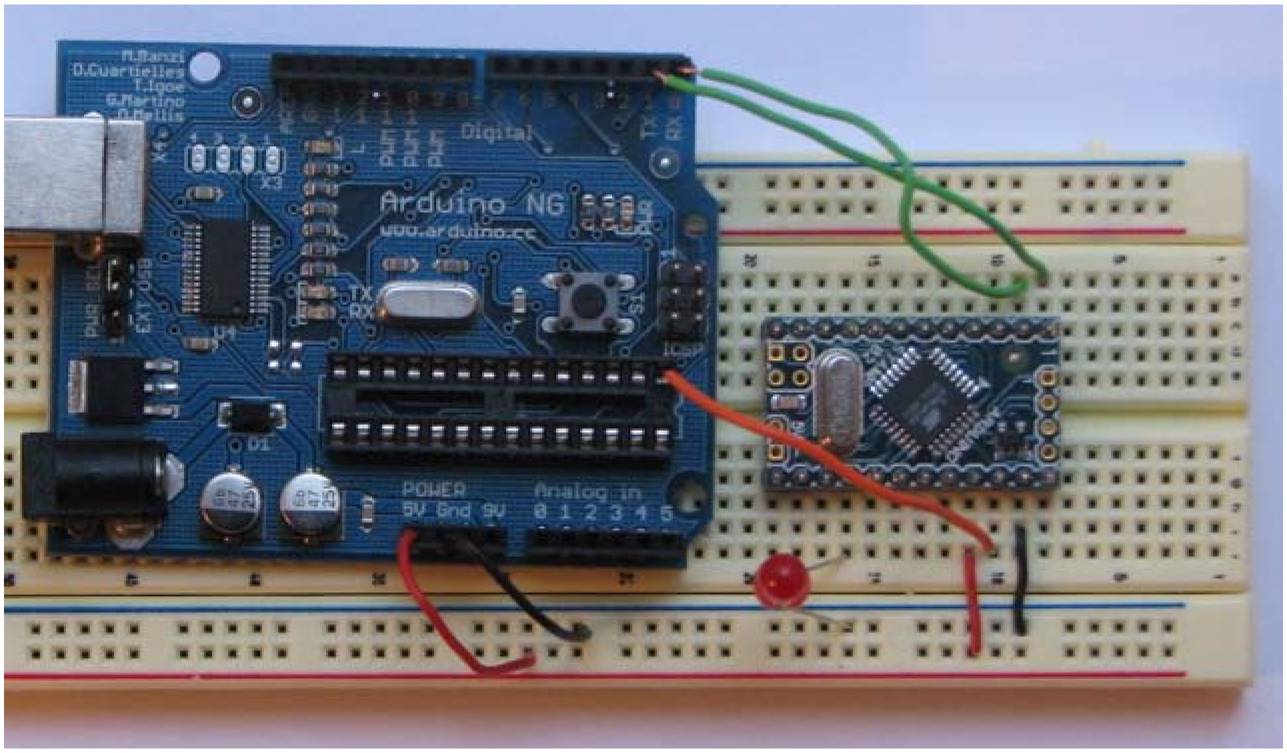
- Connecting the Arduino Mini is a bit more complicated than a regular Arduino board (see below for instructions and photos).

Added line 25:

Restore
January 07, 2007, at 06:57 AM by David A. Mellis -
Changed lines 51-57 from:

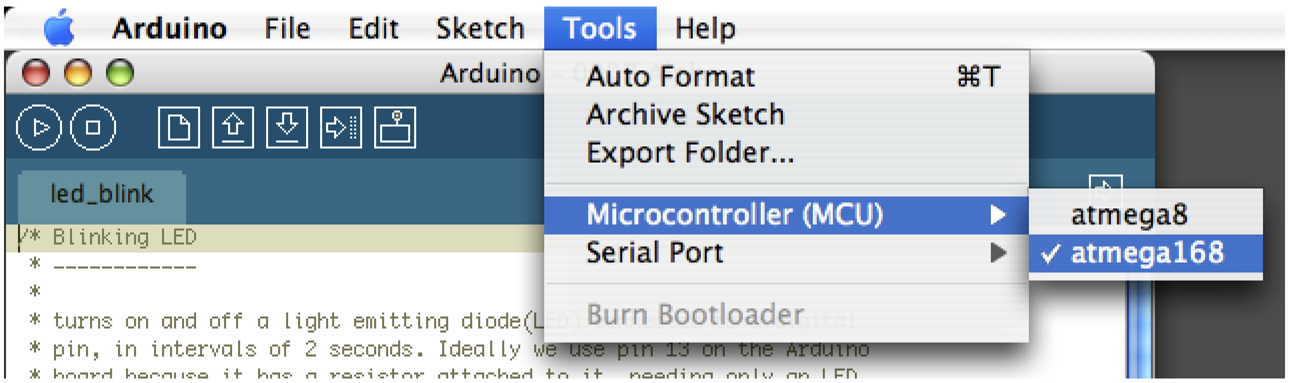


to:



### Selecting ATmega168 in the Arduino Environment

To tell the Arduino environment that you're using an Arduino Mini, which has a different chip than the regular Arduino, you need to select **ATmega168** from the **Tools | Microcontroller** menu.



```
/* Blinking LED
 * ------------
 *
 * turns on and off a light emitting diode(LED) connected to a digital
 * pin, in intervals of 2 seconds. Ideally we use pin 13 on the Arduino
 * board because it has a resistor attached to it, needing only an LED
```

Restore
January 07, 2007, at 06:53 AM by David A. Mellis -
Changed lines 11-24 from:

- The Arduino Mini has a more powerful chip than the regular Arduino board (an ATmega168 instead of an ATmega8), meaning:
  - Your sketches can be twice as big (14 KB instead of 7 KB).
  - There are three extra PWM outputs (on pins 3, 5, and 6) in addition to the three on regular Arduino boards (pins 9, 10, and 11).
  - There are two extra analog inputs (8 total). Four of these, however, are not connected to the legs that come on the Arduino Mini, requiring you to solder wires to their holes to use them. Two of these unconnected pins are also used by the Wire library (I2C), meaning that its use will require soldering as well.
- The Arduino Mini is more **fragile and easy to break** than a regular Arduino board.
  - Don't connect more than 9 volts to the +9V pin or reverse the power and ground pins of your power supply, or you might kill the ATmega168 on the Arduino Mini.
  - You can't remove the ATmega168, so if you kill it, you need a new Arduino Mini.

to:

The Arduino Mini has a more powerful chip than the regular Arduino board (an ATmega168 instead of an ATmega8), meaning:

- Your sketches can be twice as big (14 KB instead of 7 KB).
- There are three extra PWM outputs (on pins 3, 5, and 6) in addition to the three on regular Arduino boards (pins 9, 10, and 11).
- There are two extra analog inputs (8 total). Four of these, however, are not connected to the legs that come on the Arduino Mini, requiring you to solder wires to their holes to use them. Two of these unconnected pins are also used by the Wire library (I2C), meaning that its use will require soldering as well.

The Arduino Mini is more **fragile and easy to break** than a regular Arduino board.

- Don't connect more than 9 volts to the +9V pin or reverse the power and ground pins of your power supply, or you might kill the ATmega168 on the Arduino Mini.
- You can't remove the ATmega168, so if you kill it, you need a new Arduino Mini.

Restore
January 07, 2007, at 06:53 AM by David A. Mellis -
Added lines 19-24:

- The Arduino Mini is more **fragile and easy to break** than a regular Arduino board.
  - Don't connect more than 9 volts to the +9V pin or reverse the power and ground pins of your power supply, or you might kill the ATmega168 on the Arduino Mini.
  - You can't remove the ATmega168, so if you kill it, you need a new Arduino Mini.

Restore
January 07, 2007, at 06:51 AM by David A. Mellis -
Changed lines 5-6 from:

- Connecting the Arduino Mini is a bit more complicated than a regular Arduino board.

to:

- Connecting the Arduino Mini is a bit more complicated than a regular Arduino board (see below for instructions and photos).

Changed lines 11-12 from:

- The Arduino Mini has a more powerful chip than the regular Arduino board (an ATmega168 instead of an ATmega8), meaning that your sketches can be twice as big.

to:

- The Arduino Mini has a more powerful chip than the regular Arduino board (an ATmega168 instead of an ATmega8), meaning:
  - Your sketches can be twice as big (14 KB instead of 7 KB).
  - There are three extra PWM outputs (on pins 3, 5, and 6) in addition to the three on regular Arduino boards (pins 9, 10, and 11).
  - There are two extra analog inputs (8 total). Four of these, however, are not connected to the legs that come on the Arduino Mini, requiring you to solder wires to their holes to use them. Two of these unconnected pins are also used by the Wire library (I2C), meaning that its use will require soldering as well.

Restore
January 07, 2007, at 06:41 AM by David A. Mellis -
Added lines 7-10:

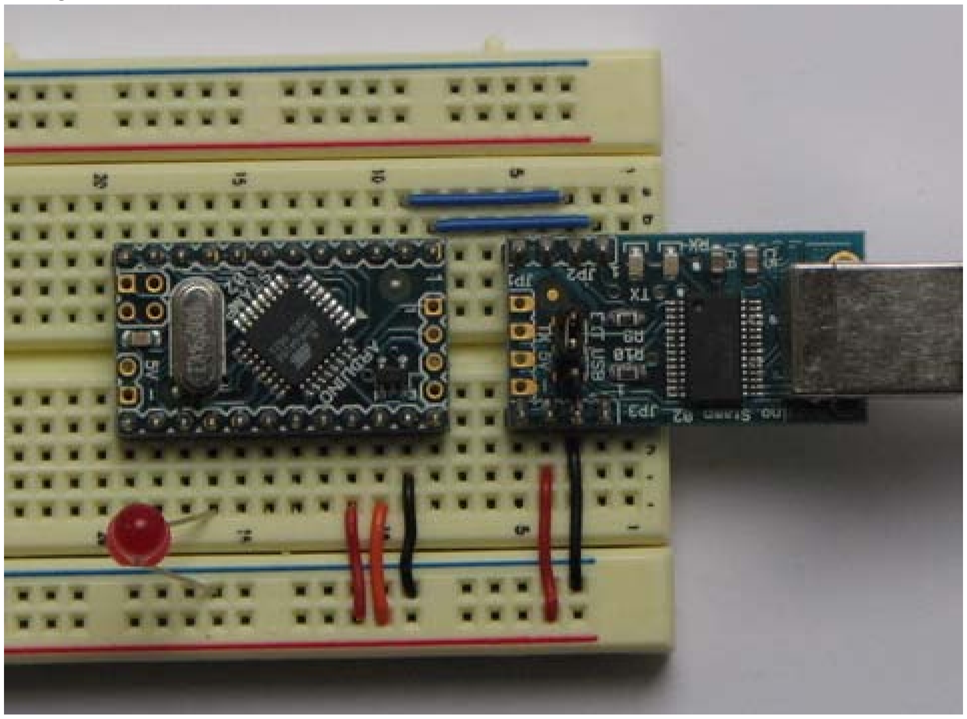
- You need to select **ATmega168** from the **Tools | Microcontroller** menu of the Arduino environment.

### Information about the Arduino Mini

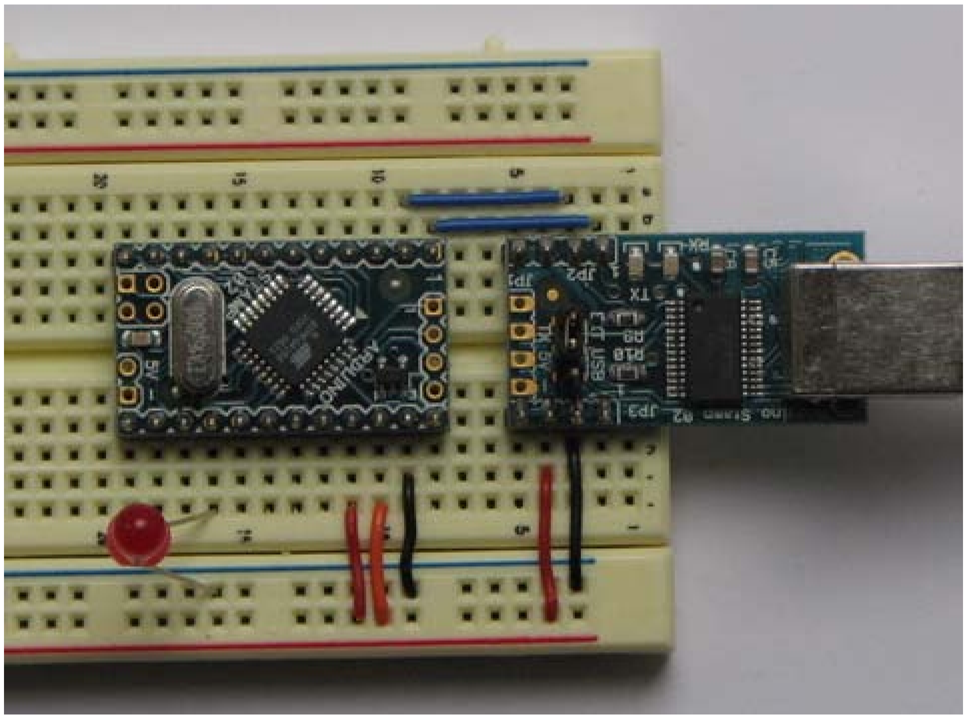
Deleted lines 12-13:

- However, you need to select **ATmega168** from the **Tools | Microcontroller** menu of the Arduino environment.

Restore
January 07, 2007, at 06:36 AM by David A. Mellis -
Changed lines 31-37 from:

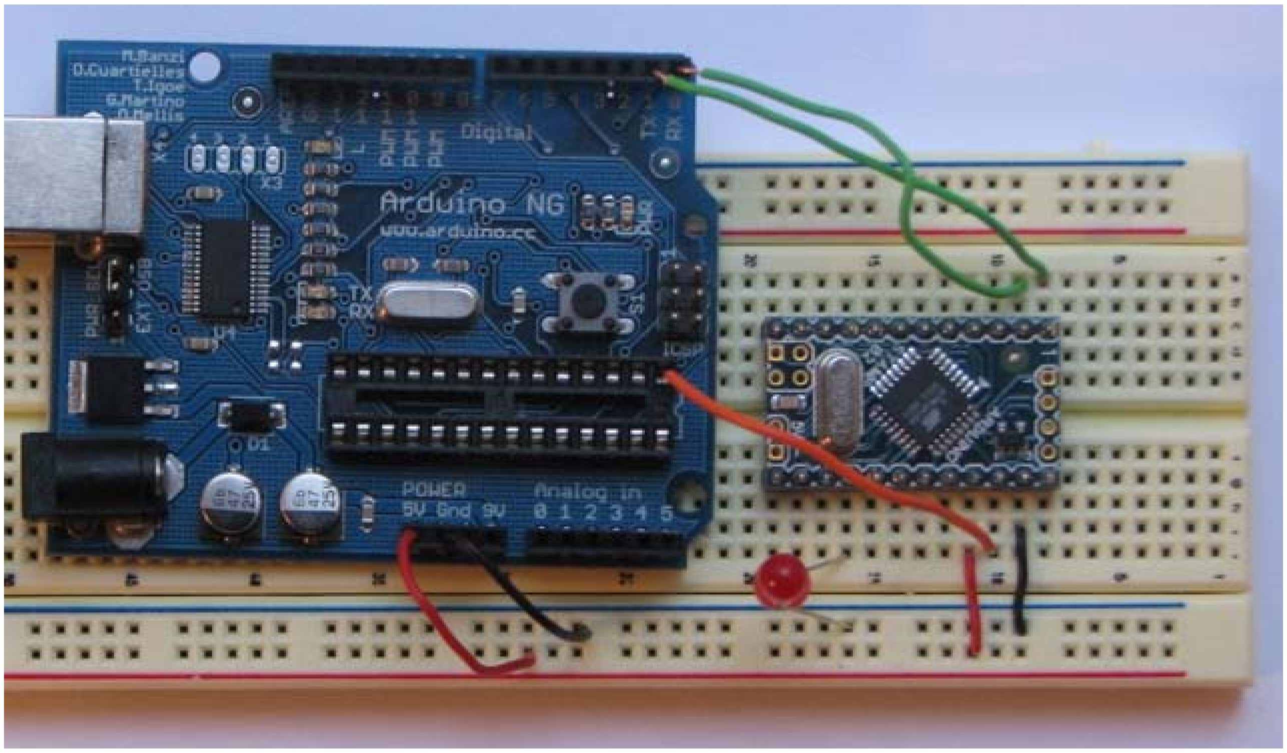


to:



**Connecting the Arduino Mini and a regular Arduino**

Here's a photo of the Arduino Mini connected to an Arduino NG. The NG has its ATmega8 removed and is being used for its USB connection, power source, and reset button. Thus, you can reset the Arduino Mini just by pressing the button on the NG.



Restore
January 07, 2007, at 05:40 AM by David A. Mellis -
Changed lines 29-30 from:

Here is a photo showing the Arduino Mini connected to the Mini USB adapter. Notice that the reset pin is connected directly to +5V, without a pushbutton. Thus, to reset the Arduino Mini, you will need to unplug and reconnect the USB cable to the Mini USB Adapter, or manually move the orange wire connected to the reset pin from +5V to ground and back.

to:

Here is a photo showing the Arduino Mini connected to the Mini USB adapter. Notice that the reset pin is connected directly to +5V (the orange wire), without a pushbutton. Thus, to reset the Arduino Mini, you will need to unplug and reconnect the

USB cable to the Mini USB Adapter, or manually move the orange wire connected to the reset pin from +5V to ground and back.

Restore
January 07, 2007, at 05:30 AM by David A. Mellis -
Changed lines 23-24 from:

- An LED. While not technically necessary, connecting an LED to the Arduino Mini makes it easier to check if it's working. Pin 13 has a 1 KB resistor on it, so you can connect an LED to it directly without an external resistor.

to:

- An LED. While not technically necessary, connecting an LED to the Arduino Mini makes it easier to check if it's working. Pin 13 has a 1 KB resistor on it, so you can connect an LED to it directly between it and ground. When using another pin, you will need an external resistor.

Added lines 29-30:

Here is a photo showing the Arduino Mini connected to the Mini USB adapter. Notice that the reset pin is connected directly to +5V, without a pushbutton. Thus, to reset the Arduino Mini, you will need to unplug and reconnect the USB cable to the Mini USB Adapter, or manually move the orange wire connected to the reset pin from +5V to ground and back.

Restore
January 07, 2007, at 05:25 AM by David A. Mellis -
Restore
January 07, 2007, at 05:13 AM by David A. Mellis -
Changed lines 7-8 from:

- You need to select **ATmega168** from the **Tools | Microcontroller** menu of the Arduino environment.

to:

- The Arduino Mini has a more powerful chip than the regular Arduino board (an ATmega168 instead of an ATmega8), meaning that your sketches can be twice as big.

- However, you need to select **ATmega168** from the **Tools | Microcontroller** menu of the Arduino environment.

Restore
January 07, 2007, at 05:11 AM by David A. Mellis -
Changed lines 3-4 from:

To get started with the Arduino Mini, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux), with the following modifications.

to:

To get started with the Arduino Mini, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux), with the following modifications:

- Connecting the Arduino Mini is a bit more complicated than a regular Arduino board.

- You need to select **ATmega168** from the **Tools | Microcontroller** menu of the Arduino environment.

Restore
January 07, 2007, at 05:09 AM by David A. Mellis -
Changed lines 3-4 from:

To get the Arduino Mini, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux), with the following modifications.

to:

To get started with the Arduino Mini, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux), with the following modifications.

Changed lines 15-16 from:

- Reset.

to:

- Reset. Whenever this pin is connected to ground, the Arduino Mini resets. You can wire it to a pushbutton, or connect it to +5V to prevent the Arduino Mini from resetting (except when it loses power). If you leave the reset pin unconnected, the Arduino Mini will reset randomly.

- An LED. While not technically necessary, connecting an LED to the Arduino Mini makes it easier to check if it's working. Pin 13 has a 1 KB resistor on it, so you can connect an LED to it directly without an external resistor.

You have a few options for connecting the board: the Mini USB Adapter, a regular Arduino board, or your own power supply and USB/Serial adapter.

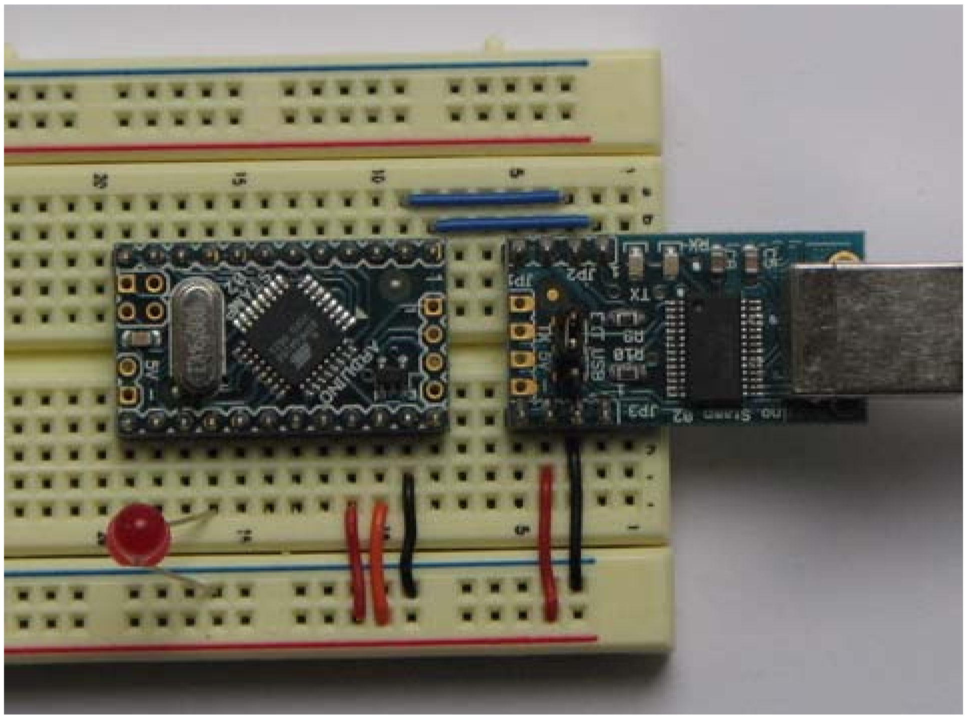
**Connecting the Arduino Mini and Mini USB Adapter**

Restore
January 07, 2007, at 05:00 AM by David A. Mellis -
Added lines 7-16:

To use the Arduino Mini, you need to connect:

- Power. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or an Arduino NG) connected to the +5V pin of the Arduino Mini. Or, a +9V power source (e.g. a 9 volt battery) connected to the +9V pin of the Arduino Mini.
- Ground. One of the ground pins on the Arduino Mini must be connected to ground of the power source.
- TX/RX. These pins are used both for uploading new sketches to the board and communicating with a computer or other device.
- Reset.

Restore
January 07, 2007, at 04:37 AM by David A. Mellis -
Added line 7:



Restore
January 07, 2007, at 04:36 AM by David A. Mellis -
Added lines 5-6:

### Connecting the Arduino Mini

Restore
January 07, 2007, at 04:35 AM by David A. Mellis -
Added lines 1-4:

## Guide to the Arduino Mini

To get the Arduino Mini, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux), with the following modifications.

# Guide to the Arduino Mini

To get started with the Arduino Mini, follow the directions for the regular Arduino on your operating system (Windows, Mac OS X, Linux), with the following modifications:

- Connecting the Arduino Mini is a bit more complicated than a regular Arduino board (see below for instructions and photos).

- You need to select **Arduino Mini** from the **Tools | Board** menu of the Arduino environment.

## Information about the Arduino Mini

The microcontroller (an ATmega168) on the Arduino Mini is a physically smaller version of the chip on the USB Arduino boards, with the following small difference:

- There are two extra analog inputs on the Mini (8 total). Four of these, however, are not connected to the legs that come on the Arduino Mini, requiring you to solder wires to their holes to use them. Two of these unconnected pins are also used by the Wire library (I2C), meaning that its use will require soldering as well.

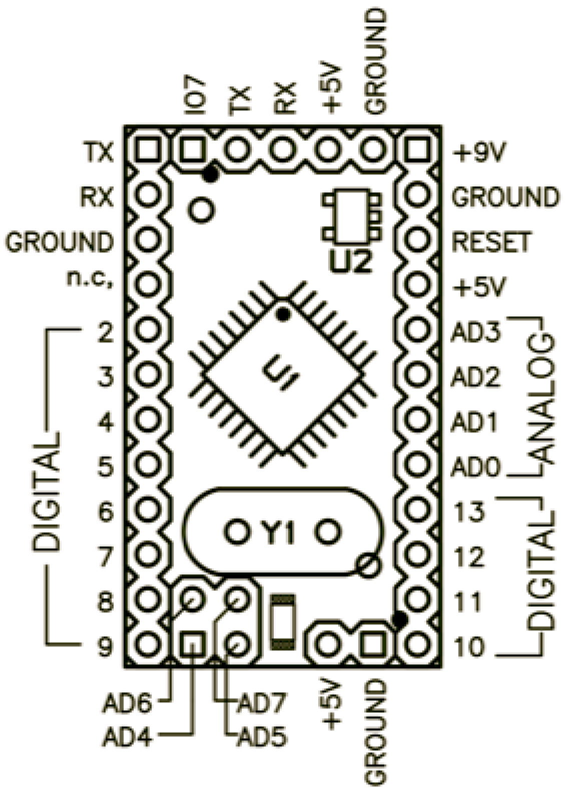
Also, the Arduino Mini is more **fragile and easy to break** than a regular Arduino board.

- Don't connect more than 9 volts to the +9V pin or reverse the power and ground pins of your power supply, or you might kill the ATmega168 on the Arduino Mini.

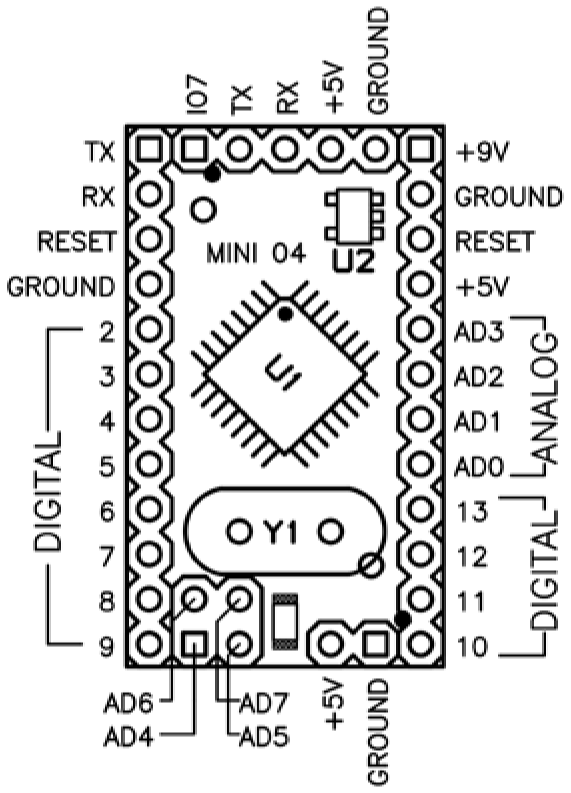
- You can't remove the ATmega168, so if you kill it, you need a new Mini.

## Connecting the Arduino Mini

Here's a diagram of the pin layout of the Arduino Mini:



*Mini 03 pinout* (compatible with earlier revisions)



*Mini 04 pinout* (the ground on the left has moved down one pin)

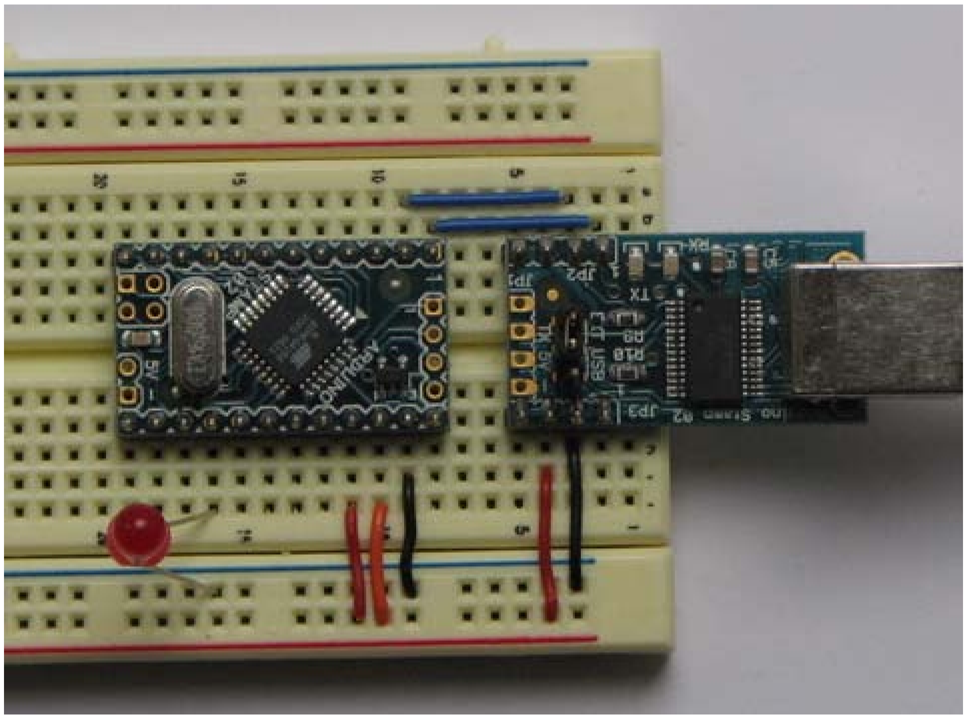
To use the Arduino Mini, you need to connect:

- Power. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or an Arduino NG) connected to the +5V pin of the Arduino Mini. Or, a +9V power source (e.g. a 9 volt battery) connected to the +9V pin of the Arduino Mini.
- Ground. One of the ground pins on the Arduino Mini must be connected to ground of the power source.
- TX/RX. These pins are used both for uploading new sketches to the board and communicating with a computer or other device.
- Reset. Whenever this pin is connected to ground, the Arduino Mini resets. You can wire it to a pushbutton, or connect it to +5V to prevent the Arduino Mini from resetting (except when it loses power). If you leave the reset pin unconnected, the Arduino Mini will reset randomly.
- An LED. While not technically necessary, connecting an LED to the Arduino Mini makes it easier to check if it's working. Pin 13 has a 1 KB resistor on it, so you can connect an LED to it directly between it and ground. When using another pin, you will need an external resistor.

You have a few options for connecting the board: the Mini USB Adapter, a regular Arduino board, or your own power supply and USB/Serial adapter.

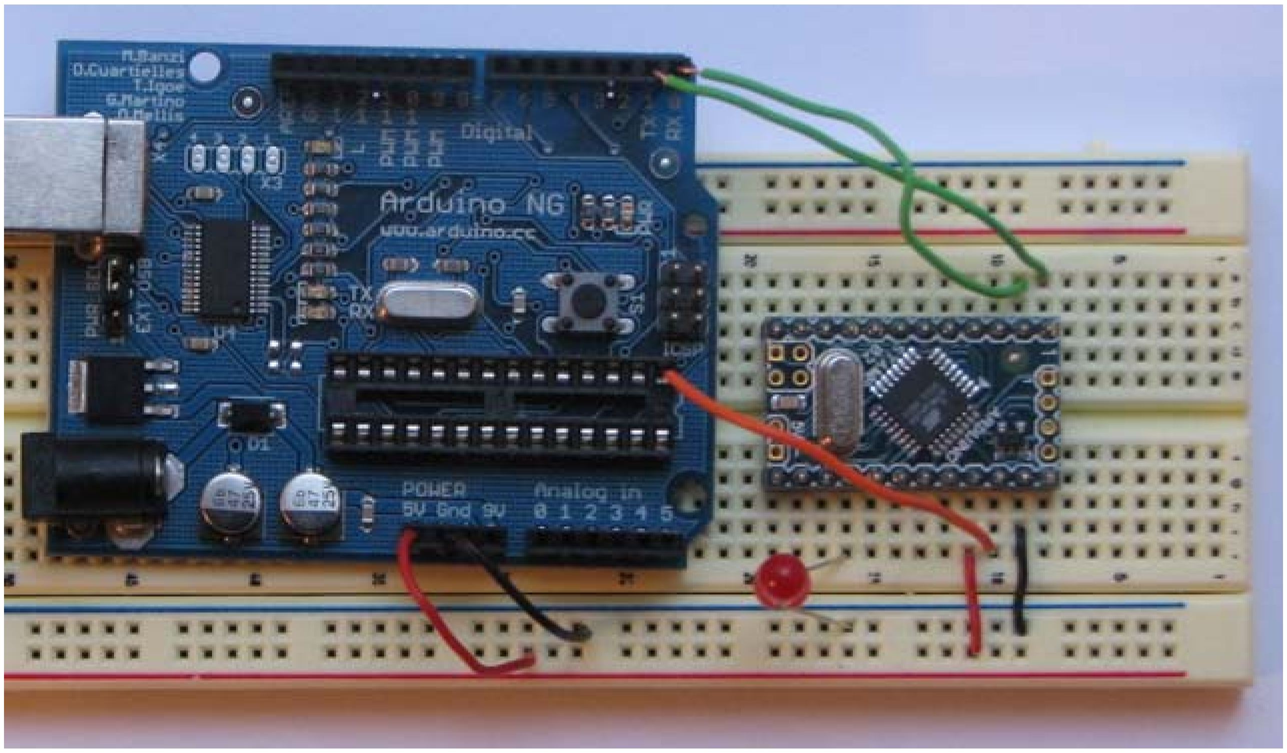
### Connecting the Arduino Mini and Mini USB Adapter

Here is a photo showing the Arduino Mini connected to the Mini USB adapter. Notice that the reset pin is connected directly to +5V (the orange wire), without a pushbutton. Thus, to reset the Arduino Mini, you will need to unplug and reconnect the USB cable to the Mini USB Adapter, or manually move the orange wire connected to the reset pin from +5V to ground and back.



## Connecting the Arduino Mini and a regular Arduino

Here's a photo of the Arduino Mini connected to an Arduino NG. The NG has its ATmega8 removed and is being used for its USB connection, power source, and reset button. Thus, you can reset the Arduino Mini just by pressing the button on the NG.



## Login to Arduino

|  |  |
|---|---|
| Username: |  |
| Password: |  |
| Keep me logged in: | ✓ |

Login

## Guide.ArduinoBT History

Hide minor edits - Show changes to markup

March 08, 2008, at 04:18 PM by David A. Mellis -
Added lines 11-12:

In most respects, the Arduino BT is similar to the Arduino Diecimila. Here are the main differences of BT board (besides the fact that it communicates over bluetooth instead of USB):

Restore
March 08, 2008, at 04:16 PM by David A. Mellis -
Changed lines 11-14 from:

- The Arduino BT is more **fragile and easy to break** than a regular Arduino board.
- Don't power the board with more than 5.5 volts to the or reverse the polarity (power and ground pins) of your power supply, or you might kill the ATmega168 on the Arduino BT. The Arduino BT can, however, run with a minimum of 1.2 volts, making it easier to power with batteries.

to:

- The Arduino BT is more fragile and easy to break than a regular Arduino board.
- **Don't power the board with more than 5.5 volts to the or reverse the polarity (power and ground pins) of your power supply, or you might kill the ATmega168 on the Arduino BT.** The Arduino BT can, however, run with a minimum of 1.2 volts, making it easier to power with batteries.

Restore
March 08, 2008, at 04:16 PM by David A. Mellis -
Changed lines 11-12 from:

The Arduino BT is more **fragile and easy to break** than a regular Arduino board.

to:

- The Arduino BT is more **fragile and easy to break** than a regular Arduino board.

Added lines 19-20:

- Pin 7 is connected to the reset pin of the bluetooth module; **don't use it for anything** (except resetting the module).

Changed lines 25-27 from:

Only communicate at 115200 baud using the serial commands (e.g. `Serial.begin()`, etc.). This is the speed for which the bluetooth is configured.

Pin 7 is connected to the reset pin of the bluetooth module; **don't use it for anything** (except resetting the module).

to:

The on-board serial communication between the bluetooth module and the Arduino sketch (running on the ATmega168) needs to be at 115200 baud (i.e. call Serial.begin(115200) in your setup() function). Communication between the bluetooth module and the computer can be at any baud rate.

Communication between the BT module and the computer can be temperamental. You might want to open the serial monitor a couple of seconds after resetting the board.

Restore
March 08, 2008, at 04:12 PM by David A. Mellis -
Deleted lines 10-13:

The microcontroller (an ATmega168) on the Arduino BT is a physically smaller version of the chip on the USB Arduino boards, with the following small difference:

- There are two extra analog inputs on the Arduino BT (8 total). Two of these, however, are not connected to the pin headers on the board; you'll need to solder something to the pads next to the numbers "6" and "7".

Changed lines 15-16 from:

- You can't remove the ATmega168, so if you kill it, you need a new Arduino BT.

to:

- The microcontroller (an ATmega168) on the Arduino BT is a physically smaller version of the chip on the USB Arduino boards. You can't remove it, so if you kill it, you need a new Arduino BT.

- There are two extra analog inputs on the Arduino BT (8 total). Two of these, however, are not connected to the pin headers on the board; you'll need to solder something to the pads next to the numbers "6" and "7".

Restore
January 25, 2008, at 05:06 PM by David A. Mellis -
Changed lines 7-8 from:

- Select **ATmega168** from the **Tools | Microcontroller (MCU)** menu of the Arduino environment.

to:

- Select **Arduino BT** from the **Tools | Board** menu of the Arduino environment.

Restore
June 15, 2007, at 05:01 PM by David A. Mellis -
Changed lines 11-18 from:

The Arduino BT has a more powerful chip than the regular Arduino board (an ATmega168 instead of an ATmega8), meaning:

- Your sketches can be twice as big (14 KB instead of 7 KB).

- There are three extra PWM outputs (on pins 3, 5, and 6) in addition to the three on regular Arduino boards (pins 9, 10, and 11).

- There are two extra analog inputs (8 total).

to:

The microcontroller (an ATmega168) on the Arduino BT is a physically smaller version of the chip on the USB Arduino boards, with the following small difference:

- There are two extra analog inputs on the Arduino BT (8 total). Two of these, however, are not connected to the pin headers on the board; you'll need to solder something to the pads next to the numbers "6" and "7".

Restore
February 21, 2007, at 05:45 AM by David A. Mellis -
Changed lines 5-6 from:

- First, pair the Arduino BT with your computer and create a virtual serial port for it.

to:

- First, pair the Arduino BT with your computer and create a virtual serial port for it. Look for a bluetooth device called **ARDUINOBT** and the pass code is **12345**.

Restore
January 27, 2007, at 08:30 AM by David A. Mellis -
Changed lines 29-31 from:

Only communicate at 115200 baud using the serial commands (e.g. `Serial.begin()`, etc.). This is the speed for which the bluetooth is configured.

to:

Only communicate at 115200 baud using the serial commands (e.g. `Serial.begin()`, etc.). This is the speed for which the bluetooth is configured.

Pin 7 is connected to the reset pin of the bluetooth module; **don't use it for anything** (except resetting the module).

Restore
January 27, 2007, at 08:29 AM by David A. Mellis -
Changed lines 25-29 from:

For more details, see the Arduino BT hardware page.

to:

For more details, see the Arduino BT hardware page.

## Using the Arduino BT

Only communicate at 115200 baud using the serial commands (e.g. `Serial.begin()`, etc.). This is the speed for which the bluetooth is configured.

Restore
January 27, 2007, at 08:18 AM by David A. Mellis -
Changed lines 19-20 from:

The Arduino Mini is more **fragile and easy to break** than a regular Arduino board.

to:

The Arduino BT is more **fragile and easy to break** than a regular Arduino board.

Restore
January 27, 2007, at 08:18 AM by David A. Mellis -
Changed lines 3-15 from:

The Arduino BT is an Arduino board with built-in bluetooth module, allowing for wireless communication. This page explains how to get started using your Arduino BT. For more information on the board, see the Arduino BT hardware page.

## Using the Arduino BT

To use the ArduinoBT, you'll need bluetooth connectivity for your computer. Many computers come with bluetooth connectivity, if yours doesn't, you'll need a bluetooth dongle.

First, you need to power the Arduino BT. The board takes a minimum of 1.2 volts and a maximum of 5.5 volts. **If you apply more voltage or reverse the polarity of the power supply, you'll kill the board.**

Second, you'll have to pair the Arduino BT with your computer. Search for bluetooth devices using your computer's bluetooth software. The Arduino BT will be called **ARDUINOBT**. The default pass code is **12345**. You'll also need to create a virtual serial port (also called a virtual com port) for the Arduino BT. The steps required depend on your computer's bluetooth hardware and software.

Third, you'll need to set a couple of options within the Arduino environment. Select the virtual serial port corresponding to your Arduino BT from the **Tools | Serial Port** menu. Select **ATmega168** from the **Tools | Microcontroller (MCU)** menu.

Now you should be able to upload a sketch to the Arduino BT. Open the led_blink example, reset the board, and press the upload button in the environment.

to:

The Arduino BT is an Arduino board with built-in bluetooth module, allowing for wireless communication. To get started with the Arduino BT, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux), with the following modifications:

- First, pair the Arduino BT with your computer and create a virtual serial port for it.
- Select **ATmega168** from the **Tools | Microcontroller (MCU)** menu of the Arduino environment.

## Information about the Arduino BT

The Arduino BT has a more powerful chip than the regular Arduino board (an ATmega168 instead of an ATmega8), meaning:

- Your sketches can be twice as big (14 KB instead of 7 KB).
- There are three extra PWM outputs (on pins 3, 5, and 6) in addition to the three on regular Arduino boards (pins 9, 10, and 11).
- There are two extra analog inputs (8 total).

The Arduino Mini is more **fragile and easy to break** than a regular Arduino board.

- Don't power the board with more than 5.5 volts to the or reverse the polarity (power and ground pins) of your power supply, or you might kill the ATmega168 on the Arduino BT. The Arduino BT can, however, run with a minimum of 1.2 volts, making it easier to power with batteries.
- You can't remove the ATmega168, so if you kill it, you need a new Arduino BT.

For more details, see the Arduino BT hardware page.

Restore
January 27, 2007, at 08:12 AM by David A. Mellis -
Changed lines 9-15 from:

First, you need to power the Arduino BT. The board takes a minimum of 1.2 volts and a maximum of 5.5 volts. **If you apply more voltage or reverse the polarity of the power supply, you'll kill the board.**

to:

First, you need to power the Arduino BT. The board takes a minimum of 1.2 volts and a maximum of 5.5 volts. **If you apply more voltage or reverse the polarity of the power supply, you'll kill the board.**

Second, you'll have to pair the Arduino BT with your computer. Search for bluetooth devices using your computer's bluetooth software. The Arduino BT will be called **ARDUINOBT**. The default pass code is **12345**. You'll also need to create a virtual serial port (also called a virtual com port) for the Arduino BT. The steps required depend on your computer's bluetooth hardware and software.

Third, you'll need to set a couple of options within the Arduino environment. Select the virtual serial port corresponding to your Arduino BT from the **Tools | Serial Port** menu. Select **ATmega168** from the **Tools | Microcontroller (MCU)** menu.

Now you should be able to upload a sketch to the Arduino BT. Open the led_blink example, reset the board, and press the upload button in the environment.

Restore
January 27, 2007, at 08:04 AM by David A. Mellis -
Added lines 1-9:

## ArduinoBT

The Arduino BT is an Arduino board with built-in bluetooth module, allowing for wireless communication. This page explains how to get started using your Arduino BT. For more information on the board, see the Arduino BT hardware page.

### Using the Arduino BT

To use the ArduinoBT, you'll need bluetooth connectivity for your computer. Many computers come with bluetooth connectivity, if yours doesn't, you'll need a bluetooth dongle.

First, you need to power the Arduino BT. The board takes a minimum of 1.2 volts and a maximum of 5.5 volts. **If you apply more voltage or reverse the polarity of the power supply, you'll kill the board.**

Restore

# ArduinoBT

The Arduino BT is an Arduino board with built-in bluetooth module, allowing for wireless communication. To get started with the Arduino BT, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux), with the following modifications:

- First, pair the Arduino BT with your computer and create a virtual serial port for it. Look for a bluetooth device called **ARDUINOBT** and the pass code is **12345**.
- Select **Arduino BT** from the **Tools | Board** menu of the Arduino environment.

## Information about the Arduino BT

In most respects, the Arduino BT is similar to the Arduino Diecimila. Here are the main differences of BT board (besides the fact that it communicates over bluetooth instead of USB):

- The Arduino BT is more fragile and easy to break than a regular Arduino board.
- **Don't power the board with more than 5.5 volts to the or reverse the polarity (power and ground pins) of your power supply, or you might kill the ATmega168 on the Arduino BT.** The Arduino BT can, however, run with a minimum of 1.2 volts, making it easier to power with batteries.
- The microcontroller (an ATmega168) on the Arduino BT is a physically smaller version of the chip on the USB Arduino boards. You can't remove it, so if you kill it, you need a new Arduino BT.
- There are two extra analog inputs on the Arduino BT (8 total). Two of these, however, are not connected to the pin headers on the board; you'll need to solder something to the pads next to the numbers "6" and "7".
- Pin 7 is connected to the reset pin of the bluetooth module; **don't use it for anything** (except resetting the module).

For more details, see the Arduino BT hardware page.

## Using the Arduino BT

The on-board serial communication between the bluetooth module and the Arduino sketch (running on the ATmega168) needs to be at 115200 baud (i.e. call Serial.begin(115200) in your setup() function). Communication between the bluetooth module and the computer can be at any baud rate.

Communication between the BT module and the computer can be temperamental. You might want to open the serial monitor a couple of seconds after resetting the board.

# Arduino

search

## Login to Arduino

| | |
|---|---|
| Username: | |
| Password: | |
| Keep me logged in: | ✓ |

Login

## Guide.ArduinoLilyPad History

Hide minor edits - Show changes to markup

May 27, 2008, at 11:53 AM by Leah Buechley -
Changed line 60 from:

See the LilyPad Arduino tutorial on Leah's website for more information about building a working wearable: http://www.cs.colorado.edu/~buechley/LilyPad/lilypad.html. See http://www.sparkfun.com for more stitchable modules that you can use with your LilyPad Arduino.

to:

See the LilyPad Arduino tutorial on Leah's website for more information about building a working wearable. See SparkFun for more stitchable modules that you can use with your LilyPad Arduino.

Restore
March 17, 2008, at 06:21 PM by Leah Buechley -
Changed lines 25-26 from:

The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to the -, tx, rx, 5v labeled holes at the top of your LilyPad to make the connection. The LilyPad USB Link is available here. and right angle male headers are available here

to:

The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to the -, tx, rx, 5v labeled holes at the top of your LilyPad to make the connection. The LilyPad USB Link is available here and right angle male headers are available here.

Restore
March 17, 2008, at 06:16 PM by Leah Buechley -
Changed lines 25-26 from:

The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to the -, tx, rx, 5v labeled holes at the top of your LilyPad to make the connection. The LilyPad USB Link is available here: http://www.sparkfun.com/commerce/product_info.php?products_id=8604 and right angle male headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553

to:

The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to the -, tx, rx, 5v labeled holes at the top of your LilyPad to make the connection. The LilyPad USB Link is available here. and right angle male headers are available here

Changed lines 31-32 from:

Solder a right angle male header to the Arduino mini USB adapter and then use female-female jumper cables to connect +,-,tx, and rx on the two boards. Right angle headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553 and female-female jumper cables are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=8430 On the version 3 Arduino mini USB adapter you want to connect tx to tx and rx to rx. We're using a red jumper for +, black for -, green for TX and yellow for RX.

to:

Solder a right angle male header to the Arduino mini USB adapter and then use female-female jumper cables to connect +,-,tx, and rx on the two boards. Right angle male headers are available here and female-female jumper cables are available here. On the version 3 Arduino mini USB adapter you want to connect tx to tx and rx to rx. We're using a red jumper for +, black for -, green for TX and yellow for RX.

Restore
March 17, 2008, at 06:11 PM by Leah Buechley -
Changed lines 31-32 from:

Solder a right angle male header to the Arduino mini USB adapter and then use female-female jumper cables to connect +,-,tx, and rx on the two boards. Right angle male headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553 and female-female jumper cables are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=8430 On the version 3 Arduino mini USB adapter you want to connect tx to tx and rx to rx. We're using a red jumper for +, black for -, green for TX and yellow for RX.

to:

Solder a right angle male header to the Arduino mini USB adapter and then use female-female jumper cables to connect +,-,tx, and rx on the two boards. Right angle headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553 and female-female jumper cables are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=8430 On the version 3 Arduino mini USB adapter you want to connect tx to tx and rx to rx. We're using a red jumper for +, black for -, green for TX and yellow for RX.

Restore
March 17, 2008, at 06:10 PM by Leah Buechley -
Changed lines 8-9 from:

http://www.cs.colorado.edu/~buechley/LilyPad/lilypad.html

to:

http://www.cs.colorado.edu/~buechley/LilyPad

Restore
March 17, 2008, at 06:09 PM by Leah Buechley -
Changed lines 21-22 from:

You have a few options for connecting the board to your computer: the SparkFun LilyPad USB Link, the Mini USB Adapter, an Arduino NG board, or your own power supply and USB/Serial adapter.

to:

You have a few options for connecting the board to your computer: the SparkFun LilyPad USB Link, the Mini USB Adapter, a regular Arduino board, or your own power supply and USB/Serial adapter.

Changed lines 31-37 from:

Solder a right angle male header to the Arduino mini USB adapter and then use female-female jumper cables to connect +,-,tx, and rx on the two boards. Right angle male headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553 and female-female jumper cables are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=8430

On the version 3 Arduino mini USB adapter you want to connect tx to tx and rx to rx. We're using a red jumper for +, black for -, green for TX and yellow for RX.

to:

Solder a right angle male header to the Arduino mini USB adapter and then use female-female jumper cables to connect +,-,tx, and rx on the two boards. Right angle male headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553 and female-female jumper cables are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=8430 On the version 3 Arduino mini USB adapter you want to connect tx to tx and rx to rx. We're using a red jumper for +, black for -, green for TX and yellow for RX.

Restore
March 17, 2008, at 06:06 PM by Leah Buechley -
Added lines 35-37:

On the version 3 Arduino mini USB adapter you want to connect tx to tx and rx to rx. We're using a red jumper for +, black for -, green for TX and yellow for RX.

Changed lines 40-41 from:

On the version 3 Arduino mini USB adapter you want to connect tx to tx and rx to rx. We're using a red jumper for +, black for -, green for TX and yellow for RX. Here is a close up view of the miniusb side of the connection:

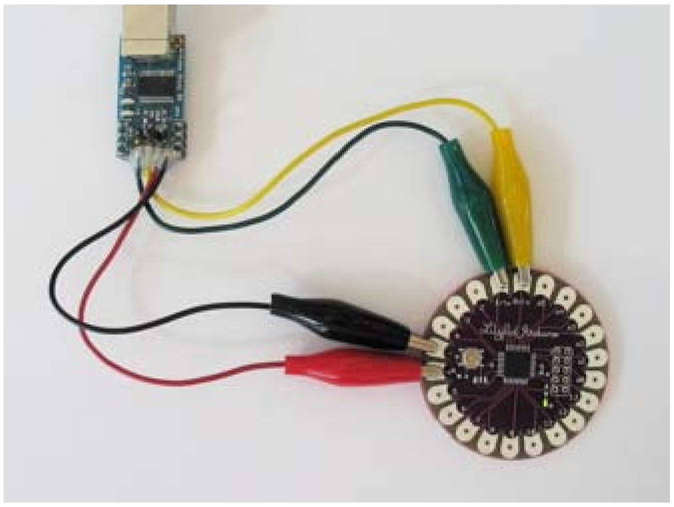
to:

Here is a close up view of the miniusb side of the connection:

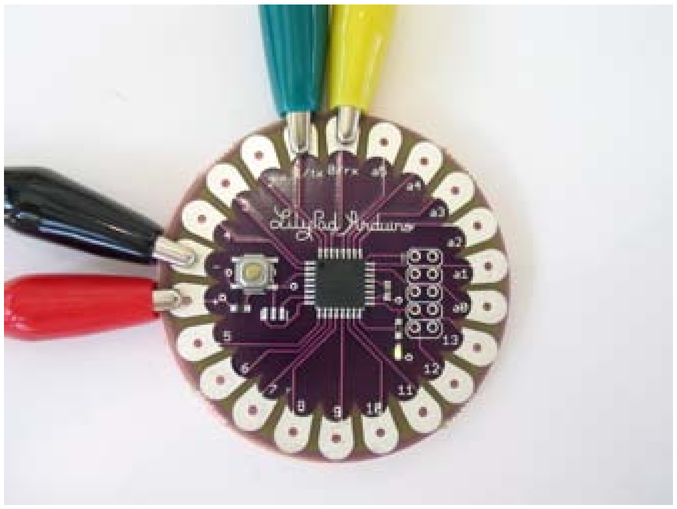
Deleted line 47:
Changed lines 50-59 from:

**Connecting the LilyPad Arduino and Mini USB Adapter**

Now you can attach the LilyPad to your computer by plugging the Mini USB into your computer and clipping the alligator clips to the TX, RX, +, and - tabs on the LilyPad. If you have a LilyPad whose TX and RX tabs are underneath a male header, you can trim the header pins to make room for your clips. You might want to cut a piece of foam or felt to put under your LilyPad before attaching the clips. This will make them less prone to slipping. Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.



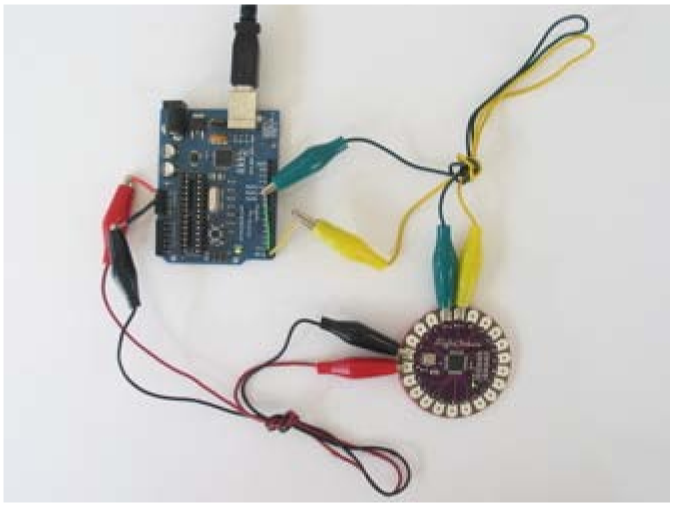
And here's a close-up showing how the alligator clips attach to the LilyPad.



to:

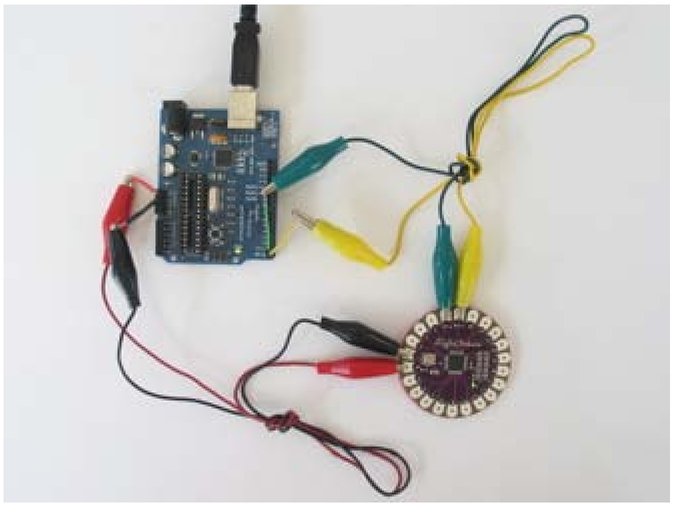
**Connecting the LilyPad Arduino and a regular Arduino**

You can also use an Arduino NG to connect the LilyPad Arduino to your computer, using a regular Arduino as a power supply and USB/Serial connection. Just remove the ATmega8 or ATmega168 from the regular Arduino and then use jumper wires and alligator clips to attach the TX, RX, +, and - tabs on the LilyPad to the corresponding pins on the NG. Here's a photo.


Deleted lines 57-64:

**Connecting the LilyPad Arduino and Arduino NG**

You can also use an Arduino NG to connect the LilyPad Arduino to your computer, using the Arduino NG as a power supply and USB/Serial connection. Just remove the ATmega8 or ATmega168 from the NG and then use jumper wires and alligator clips to attach the TX, RX, +, and - tabs on the LilyPad to the corresponding pins on the NG. Here's a photo.



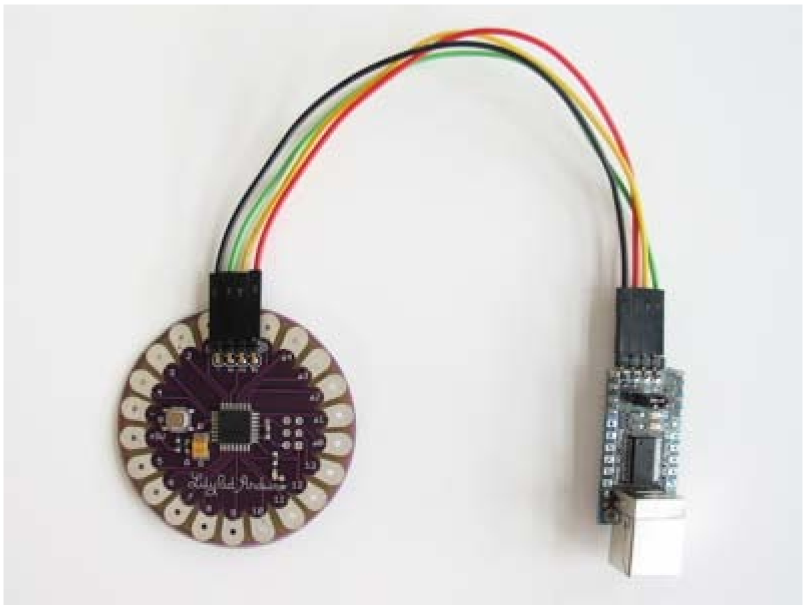
Restore
March 17, 2008, at 06:03 PM by Leah Buechley -
Changed lines 31-34 from:

Solder alligator clips to the TX, RX, +, and - pins on the front of the Mini USB Adapter. We're using a red clip for +, black for -, green for TX and yellow for RX.

To attach an alligator clip, cut it in half, strip the insulation off the wire, and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to cover the solder joints with plastic. This will prevent them from breaking. Here is a close-up view of the modified Mini USB Adapter before hot glue was applied.

to:

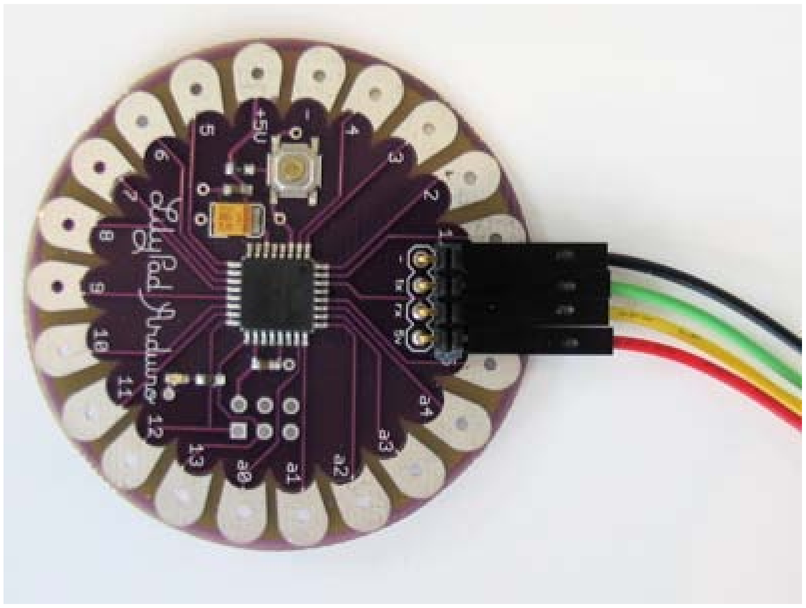
Solder a right angle male header to the Arduino mini USB adapter and then use female-female jumper cables to connect +,-,tx, and rx on the two boards. Right angle male headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553 and female-female jumper cables are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=8430



On the version 3 Arduino mini USB adapter you want to connect tx to tx and rx to rx. We're using a red jumper for +, black for -, green for TX and yellow for RX. Here is a close up view of the miniusb side of the connection:

Changed lines 41-45 from:
to:

And a close up of the LilyPad side of the connection:



Restore
February 20, 2008, at 08:50 AM by Leah Buechley -
Changed line 7 from:

Note: A more detailed guide to getting started with the LilyPad Arduino can be found here:

to:

Note: More information about getting started with the LilyPad Arduino can be found here:

Changed lines 21-22 from:

You have a few options for connecting the board to your computer: the Mini USB Adapter, an Arduino NG board, or your own power supply and USB/Serial adapter.

to:

You have a few options for connecting the board to your computer: the SparkFun LilyPad USB Link, the Mini USB Adapter, an

Arduino NG board, or your own power supply and USB/Serial adapter.

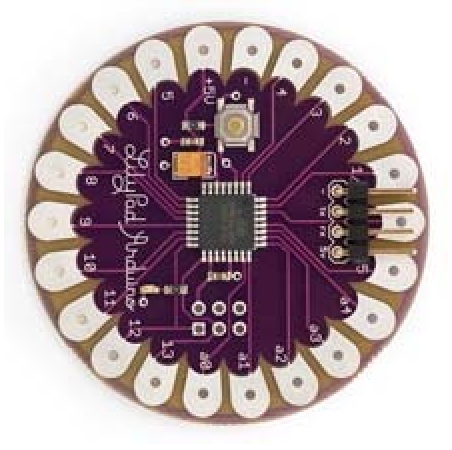
Restore
February 19, 2008, at 09:43 AM by Leah Buechley -
Changed lines 25-26 from:

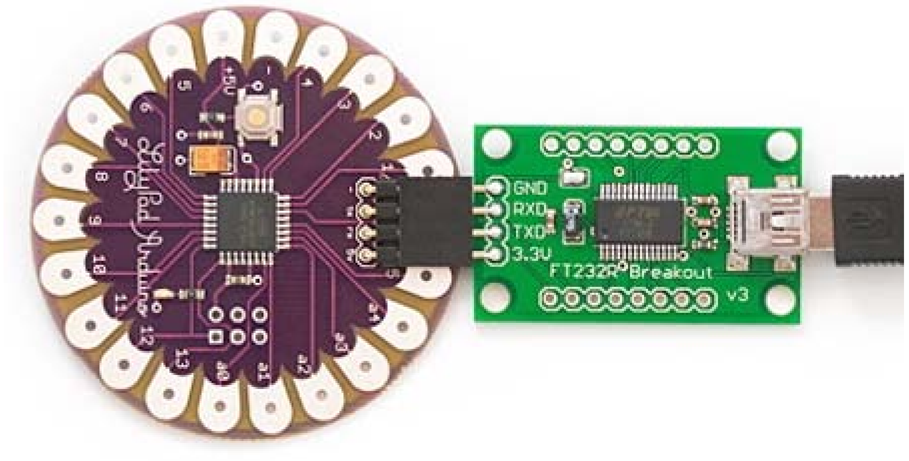
The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to the -, tx, rx, 5v labeled holes at the top of your LilyPad to make the connection. The LilyPad USB Link is available here: http://www.sparkfun.comcommerce/product_info.php?products_id=8604 and right angle male headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553

to:

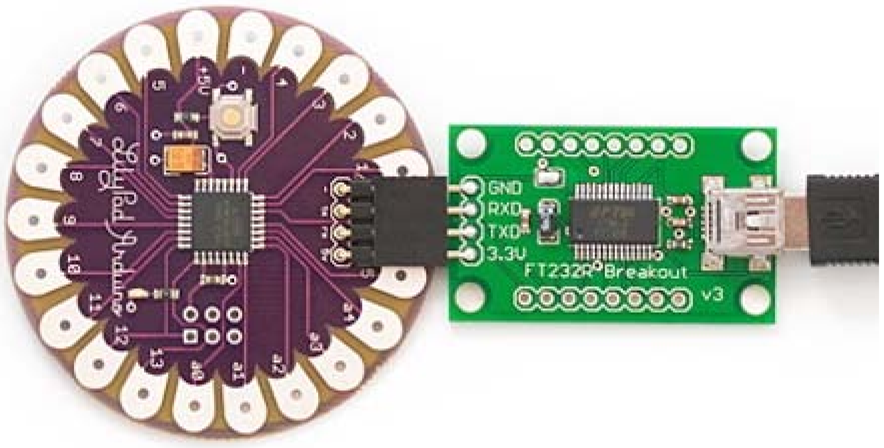
The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to the -, tx, rx, 5v labeled holes at the top of your LilyPad to make the connection. The LilyPad USB Link is available here: http://www.sparkfun.com/commerce/product_info.php?products_id=8604 and right angle male headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553

Restore
February 19, 2008, at 09:38 AM by Leah Buechley -
Restore
February 19, 2008, at 09:34 AM by Leah Buechley -
Changed lines 27-28 from:



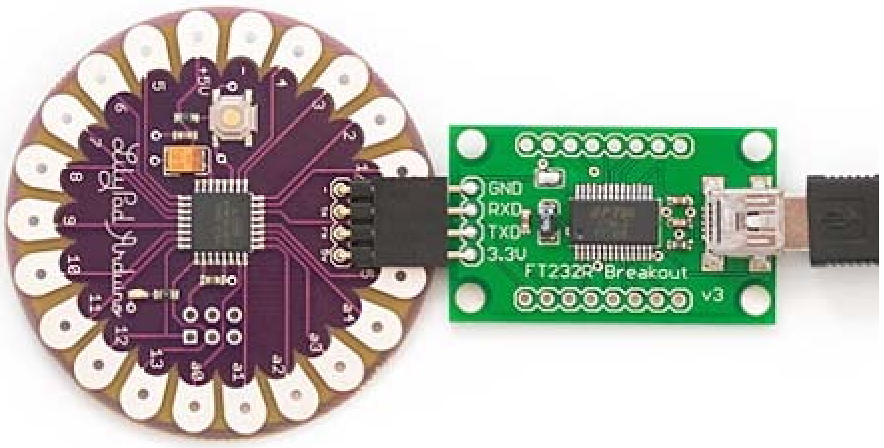


to:



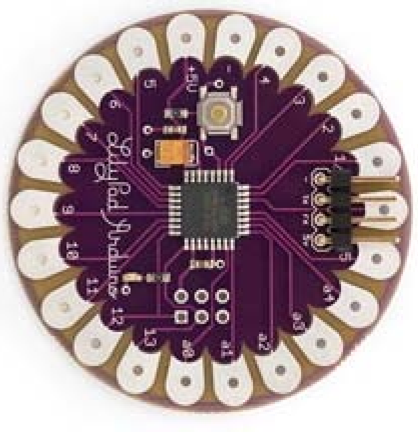
Restore
February 19, 2008, at 09:31 AM by Leah Buechley -
Changed lines 25-28 from:

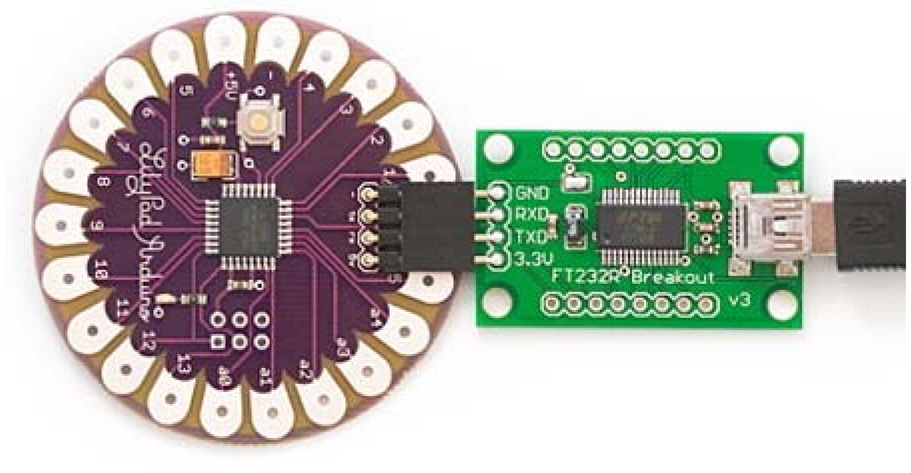
The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to your LilyPad to make the connection. The LilyPad USB Link is available here: http://www.sparkfun.comcommerce/product_info.php?products_id=8604 and right angle male headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553



to:

The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to the -, tx, rx, 5v labeled holes at the top of your LilyPad to make the connection. The LilyPad USB Link is available here: http://www.sparkfun.comcommerce/product_info.php?products_id=8604 and right angle male headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553





Restore
February 19, 2008, at 09:22 AM by Leah Buechley -
Changed lines 25-26 from:

The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to your LilyPad to make the connection. LilyPad USB Link available here: http://www.sparkfun.comcommerce/product_info.php?products_id=8604 and right angle male headers available here (just snap apart to get the 4 pins you need): http://www.sparkfun.com/commerce/product_info.php?products_id=553

to:

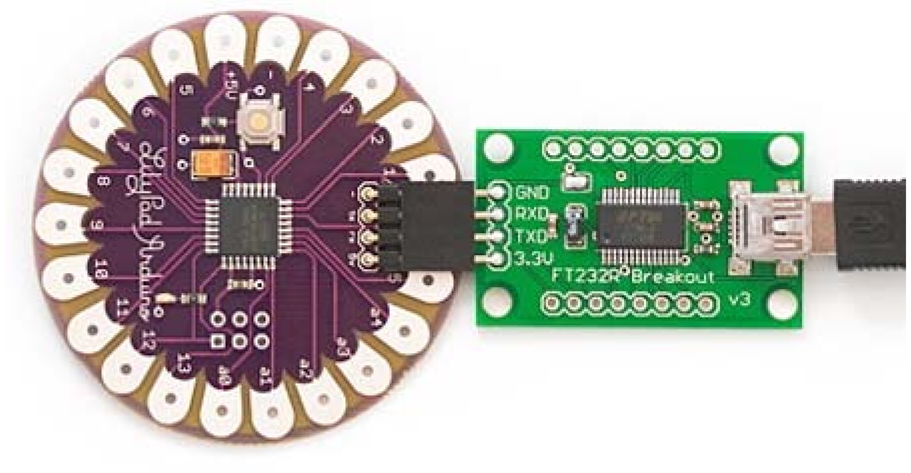
The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to your LilyPad to make the connection. The LilyPad USB Link is available here: http://www.sparkfun.comcommerce/product_info.php?products_id=8604 and right angle male headers are available here: http://www.sparkfun.com/commerce/product_info.php?products_id=553

Restore
February 19, 2008, at 09:21 AM by Leah Buechley -
Added lines 25-26:

The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to your LilyPad to make the connection. LilyPad USB Link available here: http://www.sparkfun.comcommerce/product_info.php?products_id=8604 and right angle male headers available here (just snap apart to get the 4 pins you need): http://www.sparkfun.com/commerce/product_info.php?products_id=553

Restore
February 19, 2008, at 09:08 AM by Leah Buechley -
Changed lines 23-26 from:
to:

**Use the SparkFun LilyPad USB Link**



Changed lines 40-41 from:

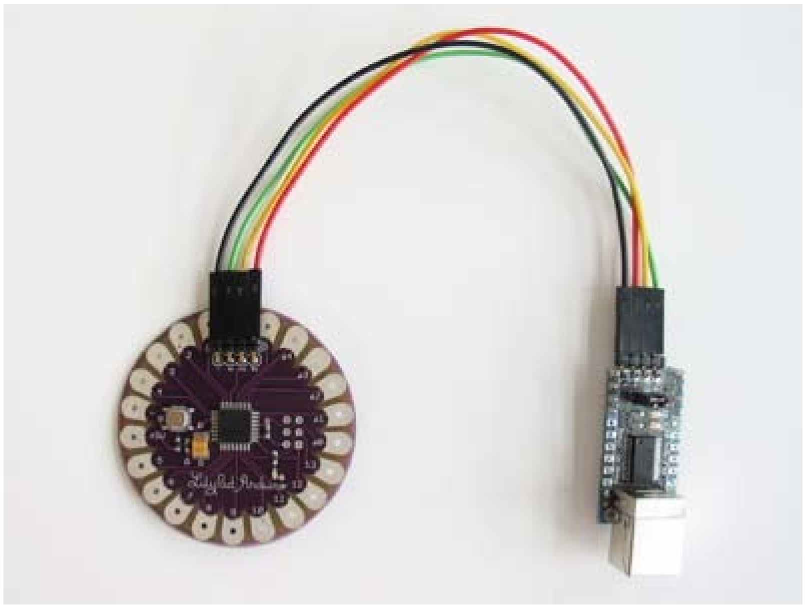
Now you can attach the LilyPad to your computer by plugging the Mini USB into your computer and clipping the alligator clips to the TX, RX, +, and - tabs on the LilyPad. You might want to cut a piece of felt to put under your LilyPad before attaching the clips. This will make them less prone to slipping. Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.

to:

Now you can attach the LilyPad to your computer by plugging the Mini USB into your computer and clipping the alligator clips to the TX, RX, +, and - tabs on the LilyPad. If you have a LilyPad whose TX and RX tabs are underneath a male header, you can trim the header pins to make room for your clips. You might want to cut a piece of foam or felt to put under your LilyPad before attaching the clips. This will make them less prone to slipping. Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.

Restore
February 17, 2008, at 08:49 PM by Leah Buechley -
Changed lines 32-35 from:

And here is a complete view after hot glue application.



to:
Restore
January 30, 2008, at 06:33 PM by Leah Buechley -
Changed line 65 from:

See the LilyPad Arduino tutorial on Leah's website for more information about building a working wearable: http://www.cs.colorado.edu/~buechley/LilyPad/lilypad.html. Pay special attention to the power supply tips. See http://www.sparkfun.com for more stitchable modules that you can use with your LilyPad Arduino.

to:

See the LilyPad Arduino tutorial on Leah's website for more information about building a working wearable: http://www.cs.colorado.edu/~buechley/LilyPad/lilypad.html. See http://www.sparkfun.com for more stitchable modules that you can use with your LilyPad Arduino.

Restore
January 30, 2008, at 06:32 PM by Leah Buechley -
Deleted lines 2-3:

For a more detailed guide to getting started, see: http://www.cs.colorado.edu/~buechley/LilyPad/lilypad.html,

Added lines 7-9:

Note: A more detailed guide to getting started with the LilyPad Arduino can be found here: http://www.cs.colorado.edu/~buechley/LilyPad/lilypad.html

Restore
January 30, 2008, at 06:30 PM by Leah Buechley -
Changed lines 3-4 from:

For more detailed instructions, see: http://www.cs.colorado.edu/~buechley/LilyPad/lilypad.html,

to:

For a more detailed guide to getting started, see: http://www.cs.colorado.edu/~buechley/LilyPad/lilypad.html,

Restore
January 30, 2008, at 06:29 PM by Leah Buechley -
Changed line 64 from:

See the LilyPad Arduino tutorial on Leah's website for more information about building a working wearable: http://www.cs.colorado.edu/~buechley/diy/diy_lilypad_arduino.html. Pay special attention to the power supply tips. See http://www.sparkfun.com for more stitchable modules that you can use with your LilyPad Arduino.

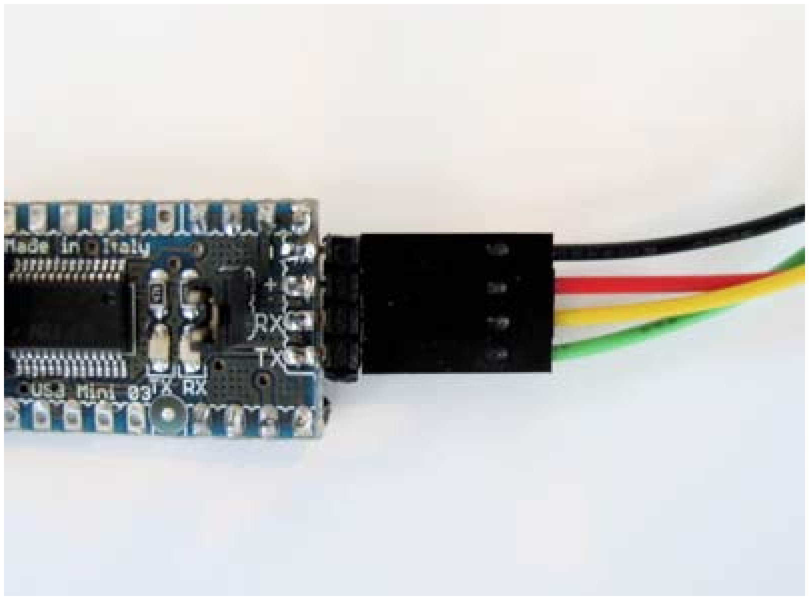
to:

See the LilyPad Arduino tutorial on Leah's website for more information about building a working wearable: http://www.cs.colorado.edu/~buechley/LilyPad/lilypad.html. Pay special attention to the power supply tips. See http://www.sparkfun.com for more stitchable modules that you can use with your LilyPad Arduino.

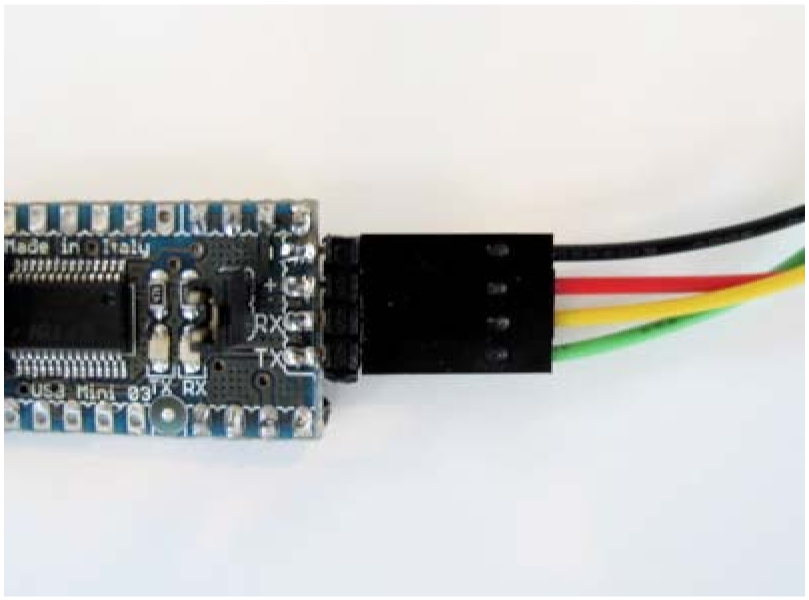
Restore
January 30, 2008, at 06:26 PM by Leah Buechley -
Changed lines 29-30 from:

Attach:miniusb-clips_close1.jpg Δ

to:



Restore
January 30, 2008, at 06:26 PM by Leah Buechley -
Changed lines 29-30 from:



to:

Attach:miniusb-clips_close1.jpg Δ

Restore
January 30, 2008, at 06:24 PM by Leah Buechley -
Changed lines 27-32 from:

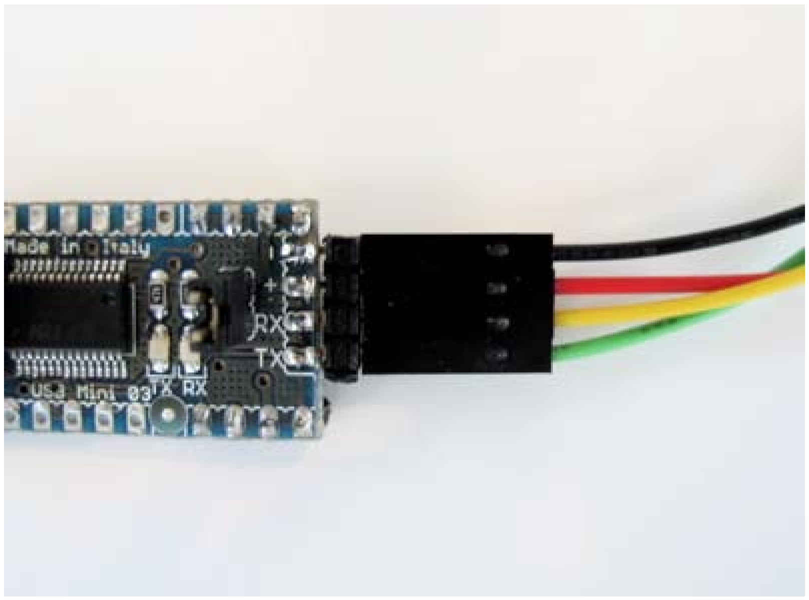
To attach an alligator clip, cut it in half, strip the insulation off the wire, and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to cover the solder joints with plastic. This will prevent them from breaking. Here is a close-up view of the modified Mini USB Adapter.

Attach:miniusb-clips_close2.jpg Δ

And here is a complete view.

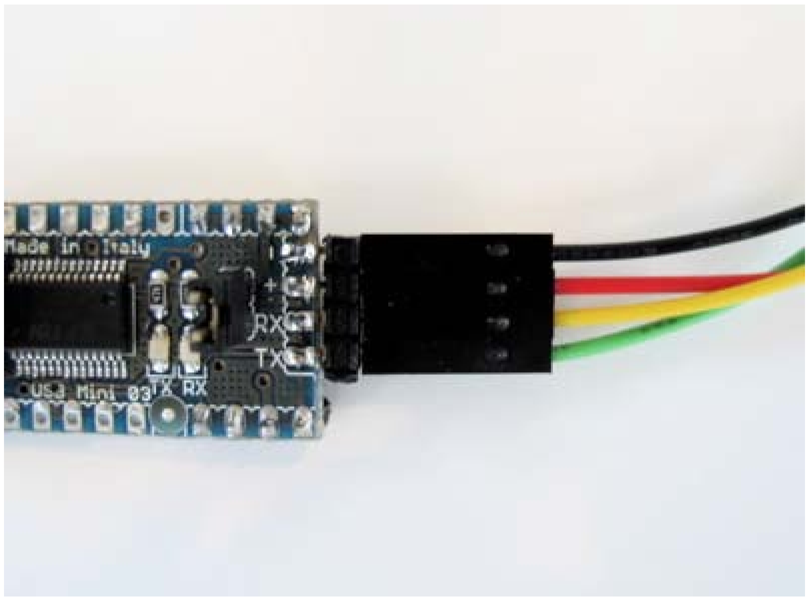
to:

To attach an alligator clip, cut it in half, strip the insulation off the wire, and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to cover the solder joints with plastic. This will prevent them from breaking. Here is a close-up view of the modified Mini USB Adapter before hot glue was applied.



And here is a complete view after hot glue application.

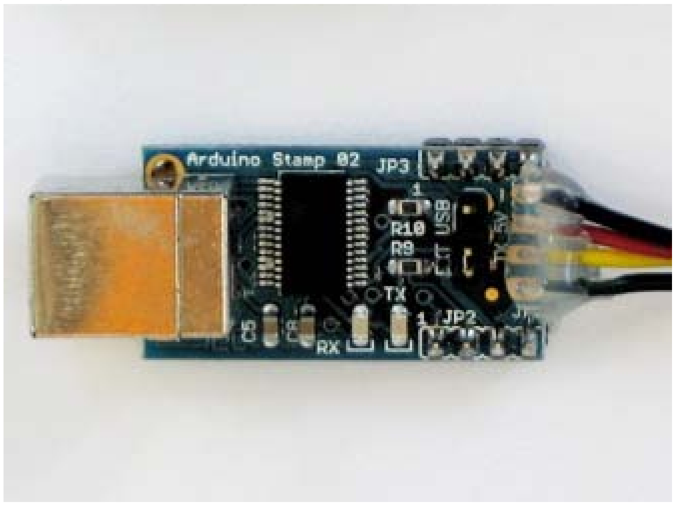
Restore
January 30, 2008, at 06:22 PM by Leah Buechley -
Changed lines 29-30 from:



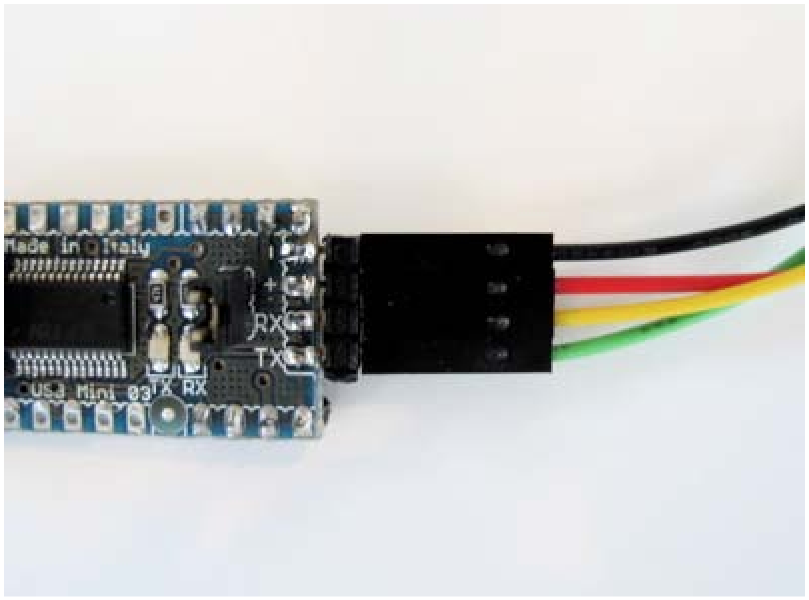
to:

Attach:miniusb-clips_close2.jpg Δ

Restore
January 30, 2008, at 06:21 PM by Leah Buechley -
Changed lines 29-30 from:



to:

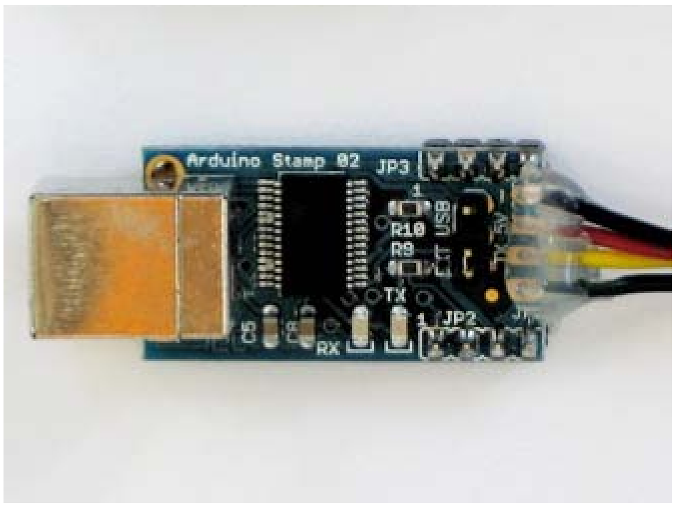


Restore

January 30, 2008, at 06:20 PM by Leah Buechley -
Changed lines 29-30 from:

Attach:miniUSB-clips2.jpg Δ

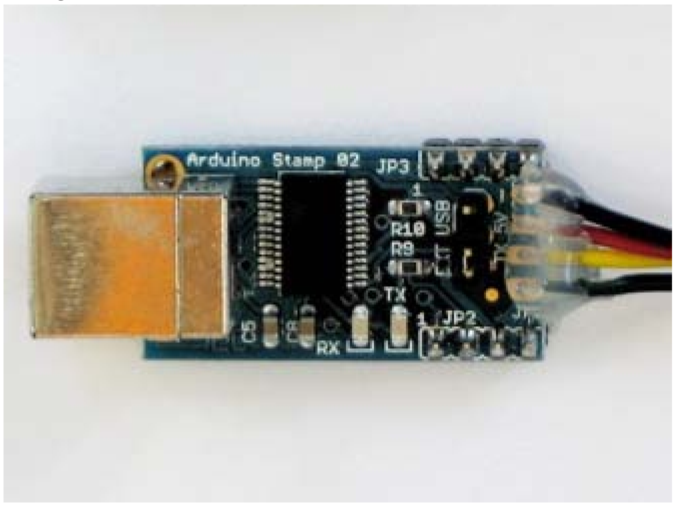
to:



Restore
January 30, 2008, at 06:18 PM by Leah Buechley -
Added lines 3-4:

For more detailed instructions, see: http://www.cs.colorado.edu/~buechley/LilyPad/lilypad.html,

Changed lines 29-30 from:



to:

Attach:miniUSB-clips2.jpg Δ

Restore
October 03, 2007, at 10:51 AM by Leah Buechley -
Changed line 62 from:

See http://www.sparkfun.com for more stitchable modules that you can use with your LilyPad Arduino.

to:

See the LilyPad Arduino tutorial on Leah's website for more information about building a working wearable: http://www.cs.colorado.edu/~buechley/diy/diy_lilypad_arduino.html. Pay special attention to the power supply tips. See http://www.sparkfun.com for more stitchable modules that you can use with your LilyPad Arduino.

Restore
October 01, 2007, at 07:39 PM by Leah Buechley -
Changed lines 37-38 from:

Now you can attach the LilyPad to your computer by plugging the Mini USB into your computer and clipping the alligator clips to the TX, RX, +, and - tabs on the LilyPad. You might want to cut a round piece of felt to go under your LilyPad before attaching the clips. This will make them less prone to slipping. Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.

to:

Now you can attach the LilyPad to your computer by plugging the Mini USB into your computer and clipping the alligator clips to the TX, RX, +, and - tabs on the LilyPad. You might want to cut a piece of felt to put under your LilyPad before attaching the clips. This will make them less prone to slipping. Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.

Restore
October 01, 2007, at 07:38 PM by Leah Buechley -
Changed lines 37-38 from:

Now you can attach the LilyPad to your computer by plugging the Mini USB into your computer and clipping the alligator clips to the TX, RX, +, and - tabs on the LilyPad. Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.

to:

Now you can attach the LilyPad to your computer by plugging the Mini USB into your computer and clipping the alligator clips to the TX, RX, +, and - tabs on the LilyPad. You might want to cut a round piece of felt to go under your LilyPad before attaching the clips. This will make them less prone to slipping. Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.

Changed lines 53-54 from:
to:

Changed lines 57-59 from:

The hole on each tab of the LilyPad is large enough for a sewing needle to pass comfortably through. Make electrical and physical connections with stitching in conductive thread. Go through each hole several times to insure good contact. Here's a picture showing a sewn LilyPad:

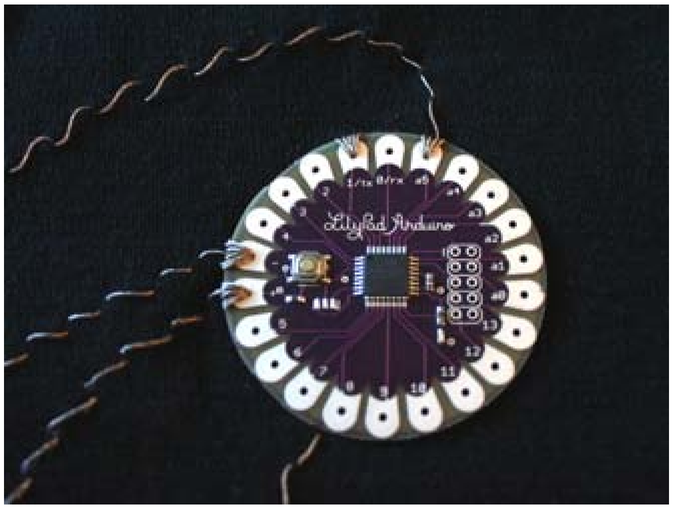
to:

The hole on each tab of the LilyPad is large enough for a sewing needle to pass through. You can make both electrical and physical connections with stitching in conductive thread. Sew through the holes several times to insure good contact. Here's a picture showing a sewn LilyPad:

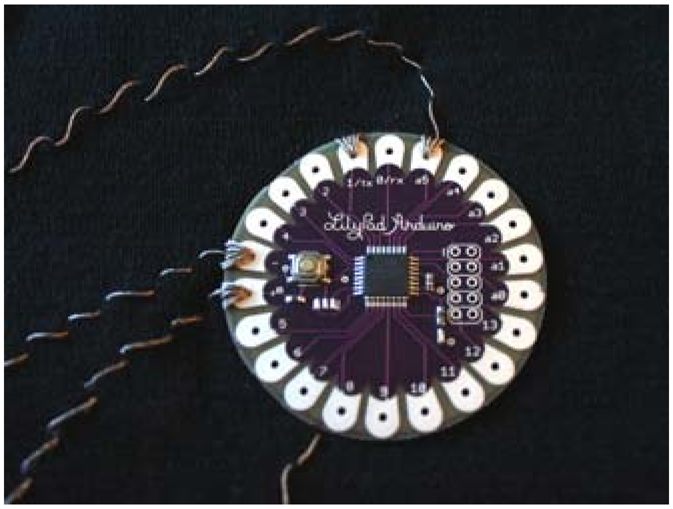
Restore
October 01, 2007, at 05:06 PM by Leah Buechley -
Changed lines 59-60 from:

Attach:lilypad_sewn2.jpg Δ

to:



Restore
October 01, 2007, at 05:05 PM by Leah Buechley -
Changed lines 59-60 from:



to:

Attach:lilypad_sewn2.jpg Δ

Restore
October 01, 2007, at 04:56 PM by Leah Buechley -
Changed line 61 from:

See http://www.sparkfun.com/commerce/categories.php?cPath=2_135? for more stitchable modules that you can use with your LilyPad Arduino.

to:

See http://www.sparkfun.com for more stitchable modules that you can use with your LilyPad Arduino.

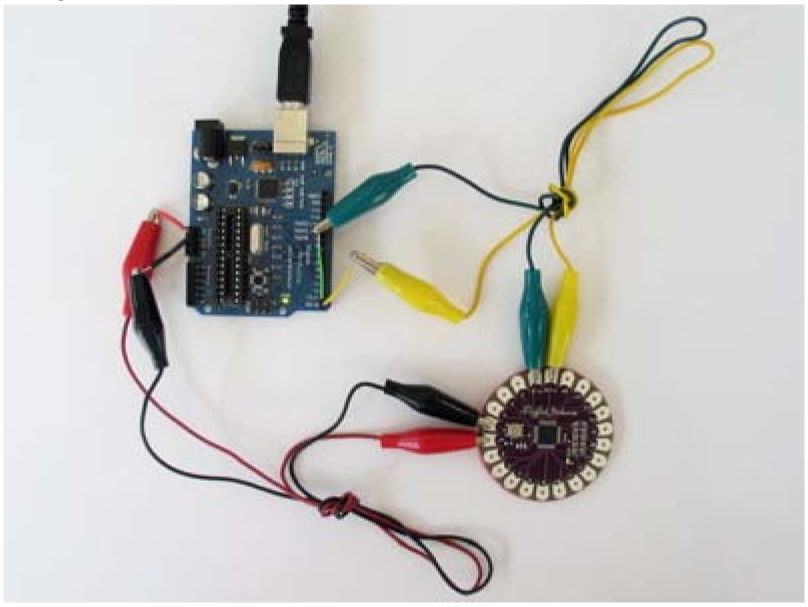
Restore
October 01, 2007, at 04:55 PM by Leah Buechley -
Changed line 61 from:

See Spark Fun Electronics?,http://www.sparkfun.com/commerce/categories.php?cPath=2_135 for more stitchable modules that you can use with your LilyPad Arduino.

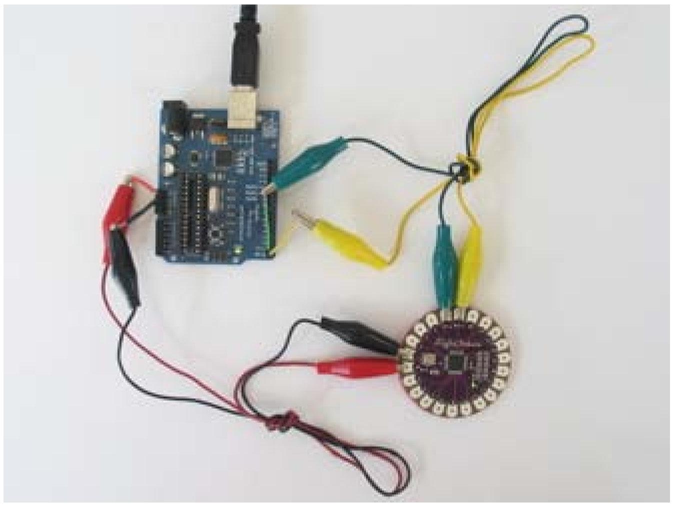
to:

See http://www.sparkfun.com/commerce/categories.php?cPath=2_135? for more stitchable modules that you can use with your LilyPad Arduino.

Restore
October 01, 2007, at 04:53 PM by Leah Buechley -
Changed lines 51-53 from:



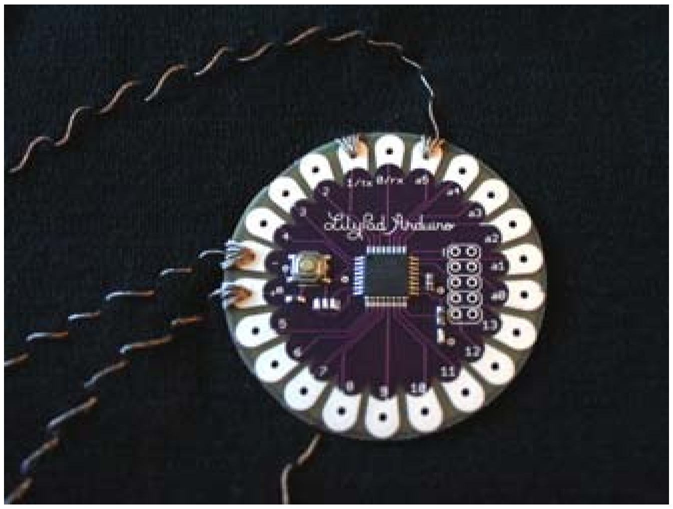
to:



Restore
October 01, 2007, at 04:40 PM by Leah Buechley -
Added lines 54-61:

**Sewing the LilyPad Arduino**

The hole on each tab of the LilyPad is large enough for a sewing needle to pass comfortably through. Make electrical and physical connections with stitching in conductive thread. Go through each hole several times to insure good contact. Here's a picture showing a sewn LilyPad:



See Spark Fun Electronics?,http://www.sparkfun.com/commerce/categories.php?cPath=2_135 for more stitchable modules that you can use with your LilyPad Arduino.

Restore
September 29, 2007, at 11:30 PM by Leah Buechley -
Changed lines 10-11 from:

To use the LilyPad Arduino, you need to connect:

to:

To program the LilyPad Arduino, you need to connect it to your computer. To do this, you'll need to connect:

Changed lines 18-20 from:

You have a few options for connecting the board: the Mini USB Adapter, a regular Arduino board, or your own power supply and USB/Serial adapter.

to:

You have a few options for connecting the board to your computer: the Mini USB Adapter, an Arduino NG board, or your own power supply and USB/Serial adapter.

Changed lines 23-24 from:

Clip the legs off of the Mini USB Adapter and solder alligator clips to the TX, RX, +, and - pins on the front of the board. We're using a red clip for +, black for -, green for TX and yellow for RX.

to:

Solder alligator clips to the TX, RX, +, and - pins on the front of the Mini USB Adapter. We're using a red clip for +, black for -, green for TX and yellow for RX.

Changed lines 37-38 from:

Now you can attach the LilyPad to your computer by clipping the alligator clips to the TX, RX, +, and - tabs on the LilyPad. Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.

to:

Now you can attach the LilyPad to your computer by plugging the Mini USB into your computer and clipping the alligator clips to the TX, RX, +, and - tabs on the LilyPad. Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.

Changed lines 49-50 from:

To connect the LilyPad Arduino to your computer via an Arduino NG, remove the ATmega8 or ATmega168 from the NG and then use jumper wires and alligator clips to attach the TX, RX, +, and - tabs on the LilyPad to the corresponding pins on the NG. Here's a photo.

to:

You can also use an Arduino NG to connect the LilyPad Arduino to your computer, using the Arduino NG as a power supply and USB/Serial connection. Just remove the ATmega8 or ATmega168 from the NG and then use jumper wires and alligator clips to attach the TX, RX, +, and - tabs on the LilyPad to the corresponding pins on the NG. Here's a photo.

Restore
September 27, 2007, at 07:15 PM by Leah Buechley -
Changed lines 23-26 from:

Clip the legs off of the Mini USB Adapter and solder alligator clips to the TX, RX, +, and - pins on the front of the board. I use a red clip for +, black for -, green for TX and yellow for RX.

To attach an alligator clip, cut it in half, strip the insulation off the wire, and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to prevent the solder joints from breaking. Here is a close-up view of the modified Mini USB Adapter.

to:

Clip the legs off of the Mini USB Adapter and solder alligator clips to the TX, RX, +, and - pins on the front of the board. We're using a red clip for +, black for -, green for TX and yellow for RX.

To attach an alligator clip, cut it in half, strip the insulation off the wire, and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to cover the solder joints with plastic. This will prevent them from breaking. Here is a close-up view of the modified Mini USB Adapter.

Changed lines 33-34 from:
to:

Changed lines 45-46 from:
to:

Restore
September 27, 2007, at 06:40 PM by Leah Buechley -
Changed lines 40-41 from:

And here's a close-up showing how the alligator clips connect to the LilyPad.

to:

And here's a close-up showing how the alligator clips attach to the LilyPad.

Restore

September 27, 2007, at 06:28 PM by Leah Buechley -
Changed lines 5-6 from:

The LilyPad Arduino is more **fragile and easy to break** than a regular Arduino board. Don't connect more than 5 volts to the + tab or reverse the power and ground pins of your power supply, or you will very likely kill the ATmega168V on the LilyPad Arduino. You can't remove the ATmega168V, so if you kill it, you need a new LilyPad.

to:

The LilyPad Arduino is more **fragile and easy to break** than a regular Arduino board. Don't connect more than 5.5 volts to the + tab or reverse the power and ground pins of your power supply, or you will very likely kill the ATmega168V on the LilyPad Arduino. You can't remove the ATmega168V, so if you kill it, you need a new LilyPad.

Changed lines 12-13 from:

- Power. Power should be connected to the + tab on the LilyPad Arduino. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or the + tab of a LilyPad power supply) or, another 3-5V power source (e.g. a 3.7V rechargeable Lithium Ion battery or 2 AA batteries in series).

to:

- Power. Power should be connected to the + tab on the LilyPad Arduino. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or the + tab of a LilyPad power supply) or, another 2.7-5.5V power source (e.g. a 3.7V rechargeable Lithium Ion battery or 2 AA batteries in series).

Restore
September 27, 2007, at 06:27 PM by Leah Buechley -
Changed lines 25-26 from:

To attach an alligator clip, cut it in half, strip the insulation off the wire from one of the halves and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to prevent the solder joints from breaking when the wires move. Here is a close-up view of the modified Mini USB Adapter.

to:

To attach an alligator clip, cut it in half, strip the insulation off the wire, and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to prevent the solder joints from breaking. Here is a close-up view of the modified Mini USB Adapter.

Restore
September 27, 2007, at 06:26 PM by Leah Buechley -
Changed lines 25-26 from:

To attach an alligator clip, cut it in half, strip the insulation off the wire from one of the halves and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to prevent the solder joints from breaking when the wires move. Here is a close-up view of the protected solder joints.

to:

To attach an alligator clip, cut it in half, strip the insulation off the wire from one of the halves and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to prevent the solder joints from breaking when the wires move. Here is a close-up view of the modified Mini USB Adapter.

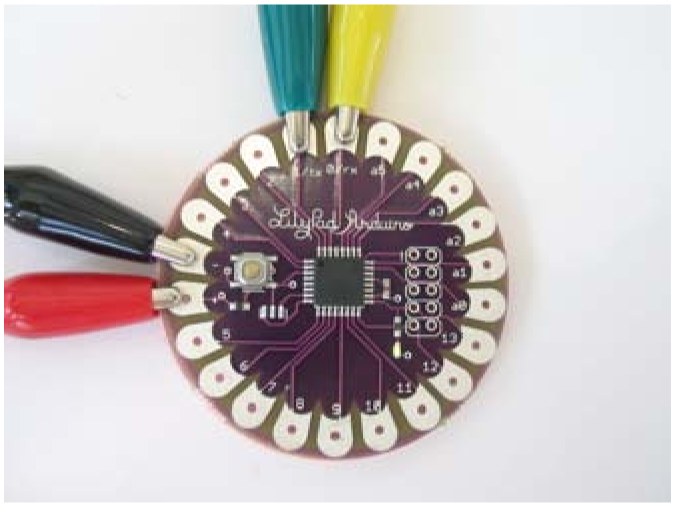
Changed lines 29-30 from:

And here is a photo showing a complete view of the modified Mini USB Adapter.

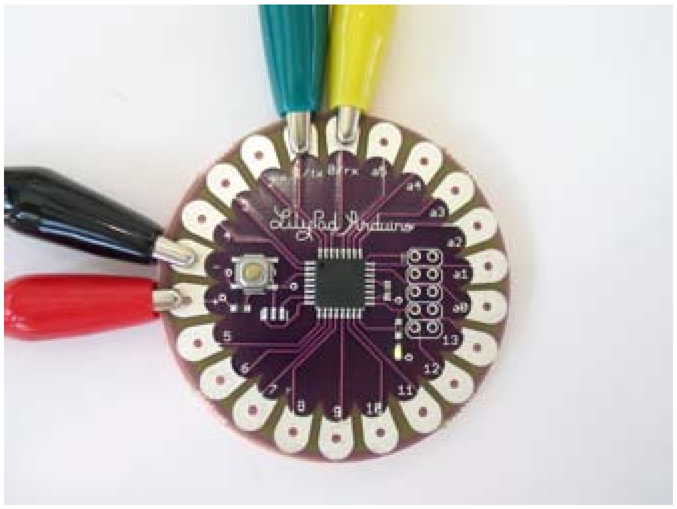
to:

And here is a complete view.

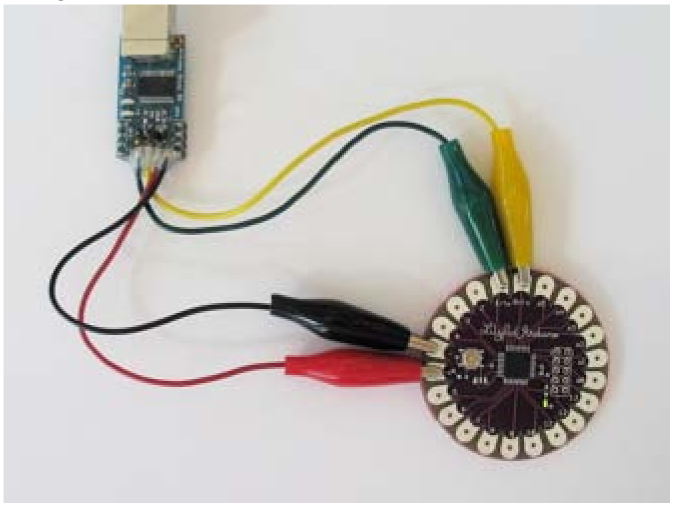
Restore
September 27, 2007, at 06:20 PM by Leah Buechley -
Changed lines 42-45 from:



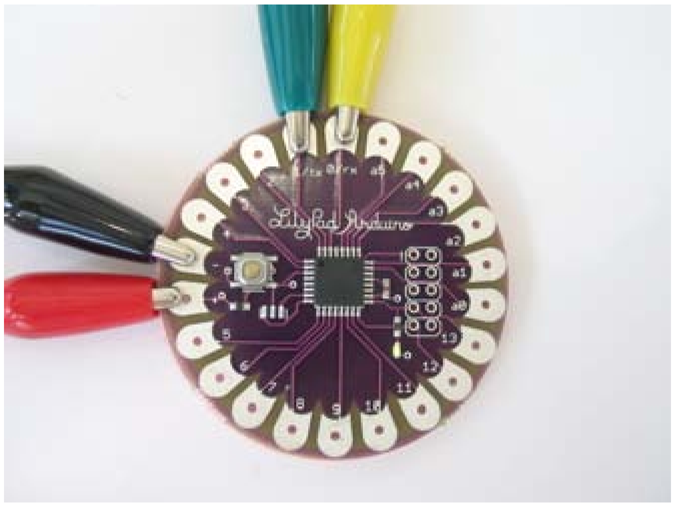
to:



Restore
September 27, 2007, at 06:18 PM by Leah Buechley -
Changed lines 42-45 from:

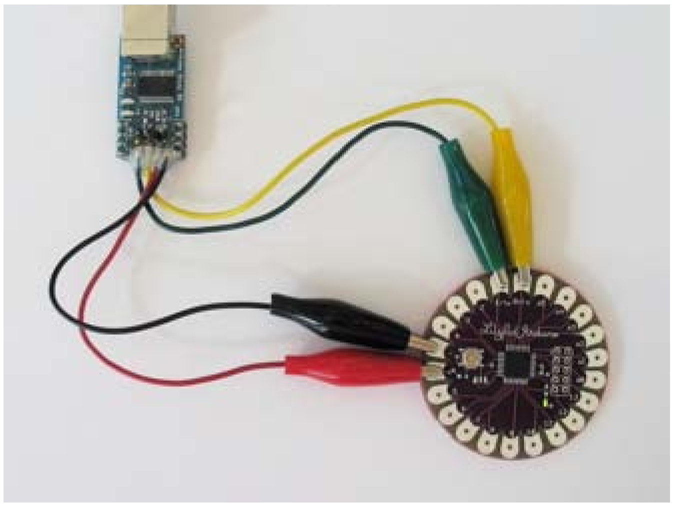


to:

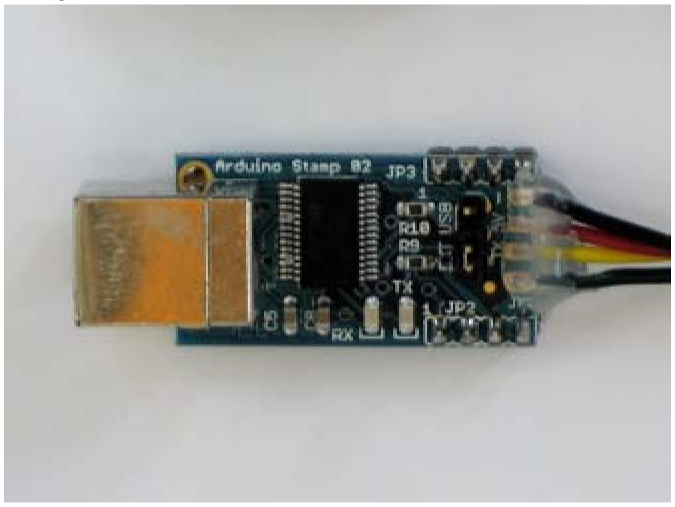


Restore
September 27, 2007, at 06:18 PM by Leah Buechley -
Changed lines 40-45 from:
to:

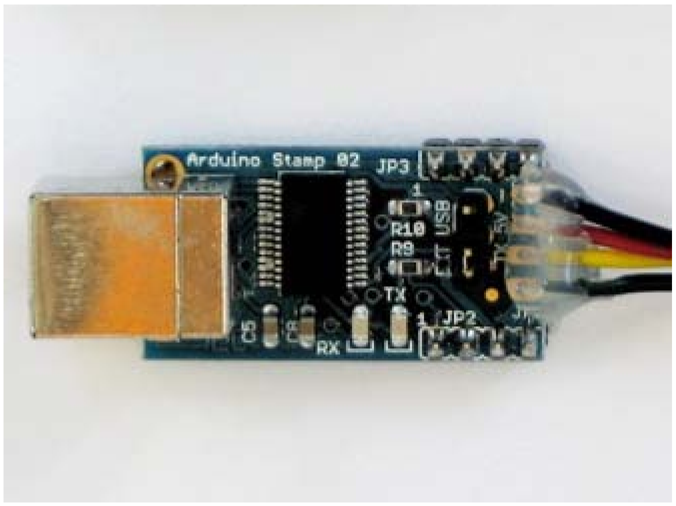
And here's a close-up showing how the alligator clips connect to the LilyPad.



Restore
September 27, 2007, at 06:15 PM by Leah Buechley -
Changed lines 27-28 from:



to:



Restore
September 27, 2007, at 06:08 PM by Leah Buechley -
Changed lines 43-44 from:

To connect the LilyPad Arduino to your computer via an Arduino NG, remove the ATmega8 or ATmega168 from the NG and then use jumper wires and alligator clips to attach the LilyPad Arduino to the TX, RX, +, and - pins on the NG. Here are photos of the LilyPad Arduino connected to a serial Arduino.

to:

To connect the LilyPad Arduino to your computer via an Arduino NG, remove the ATmega8 or ATmega168 from the NG and then use jumper wires and alligator clips to attach the TX, RX, +, and - tabs on the LilyPad to the corresponding pins on the NG. Here's a photo.

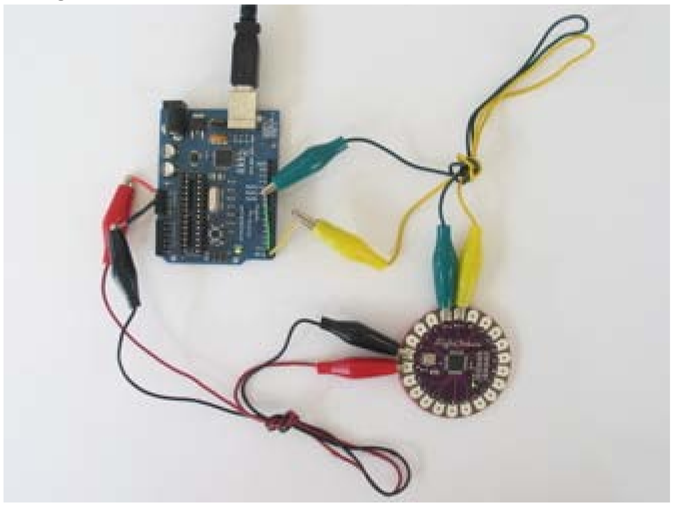
Restore
September 27, 2007, at 06:05 PM by Leah Buechley -
Changed lines 36-37 from:

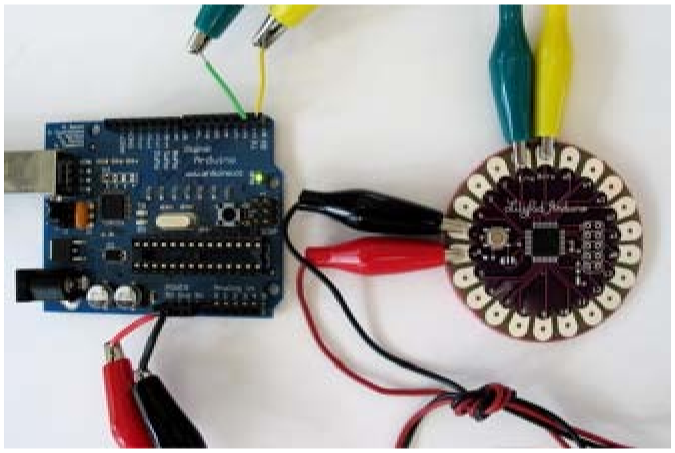
Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.

to:

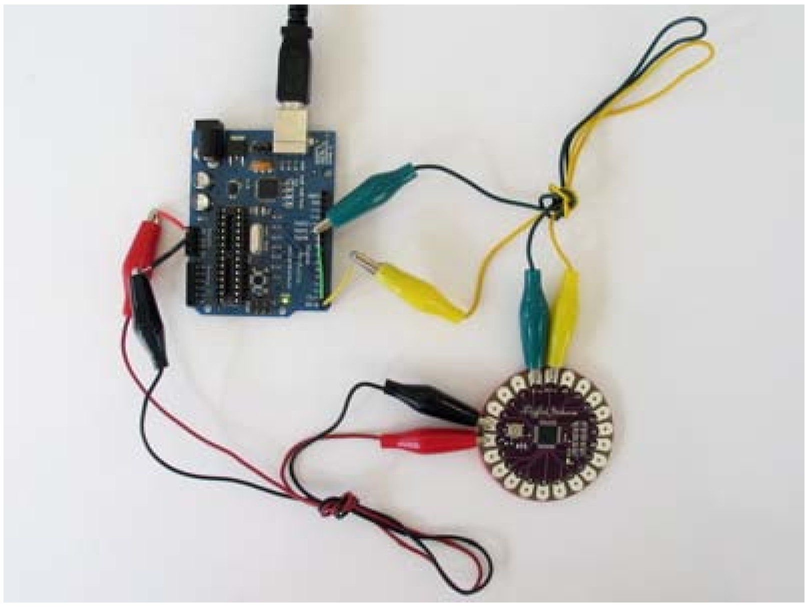
Now you can attach the LilyPad to your computer by clipping the alligator clips to the TX, RX, +, and - tabs on the LilyPad. Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.

Changed lines 45-48 from:





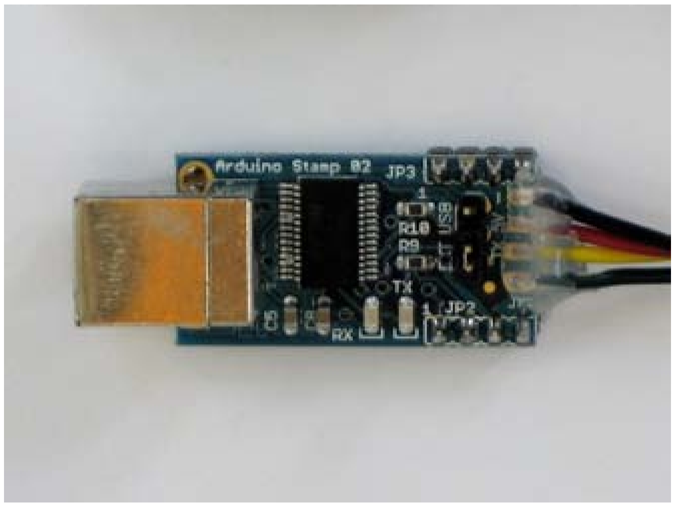
to:



Restore

September 27, 2007, at 06:01 PM by Leah Buechley -

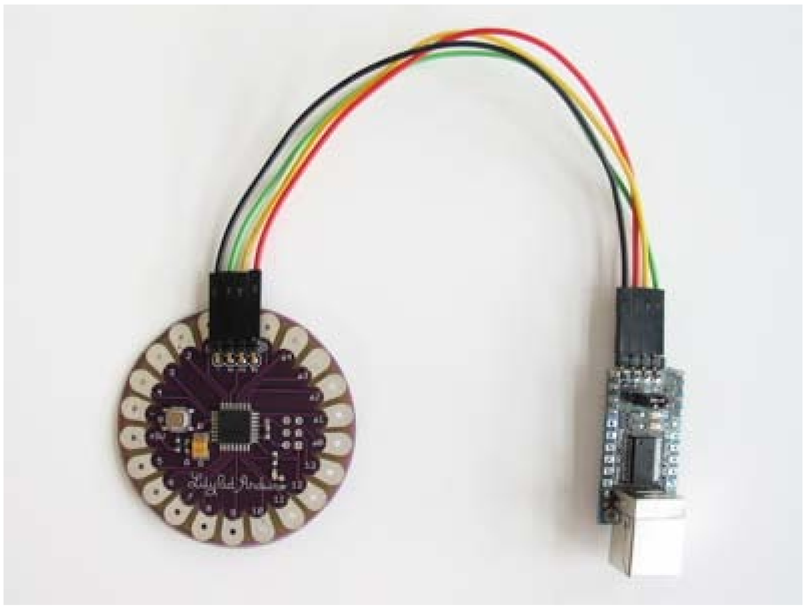
Added lines 27-30:



And here is a photo showing a complete view of the modified Mini USB Adapter.

Changed lines 33-37 from:

And here is a photo showing a complete view of the modified Mini USB Adapter.



to:
Restore
September 27, 2007, at 06:00 PM by Leah Buechley -
Changed lines 23-26 from:

Clip the legs off of the Mini USB Adapter and solder alligator clips to the TX, RX, +, and - pins on the front of the board. I use red for +, black for -, green for TX and yellow for RX.

To attach an alligator clip, cut it in half, strip the insulation off the wire from one of the halves and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to prevent the solder joints from breaking when the wires move. Here is a photo showing the modified Mini USB Adapter.

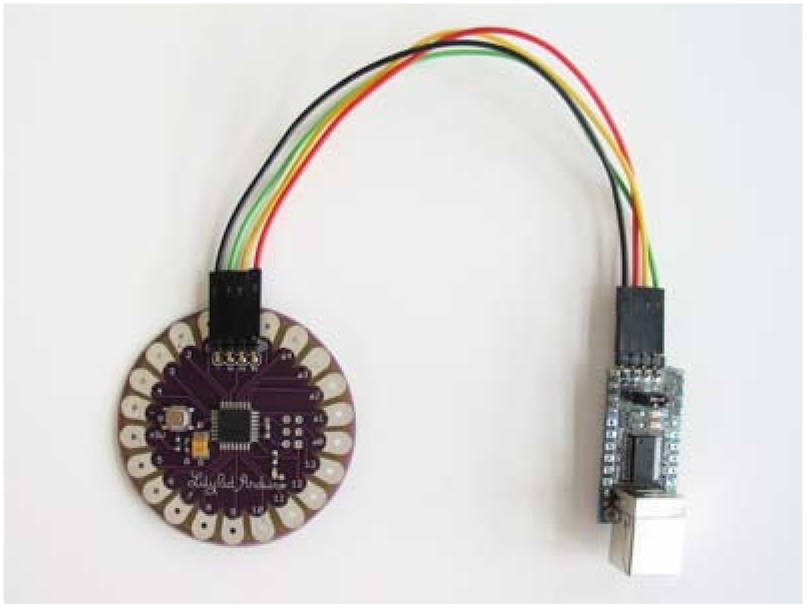
to:

Clip the legs off of the Mini USB Adapter and solder alligator clips to the TX, RX, +, and - pins on the front of the board. I use a red clip for +, black for -, green for TX and yellow for RX.

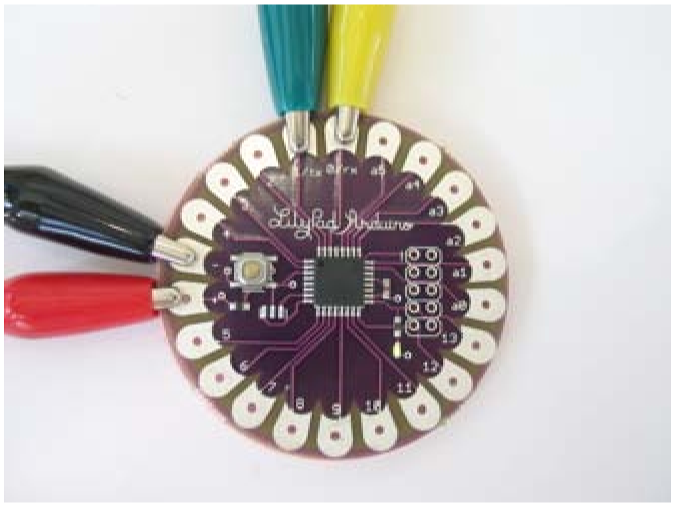
To attach an alligator clip, cut it in half, strip the insulation off the wire from one of the halves and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to prevent the solder joints from breaking when the wires move. Here is a close-up view of the protected solder joints.

Added lines 29-33:

And here is a photo showing a complete view of the modified Mini USB Adapter.



Changed lines 40-45 from:



**Connecting the LilyPad Arduino and a regular Arduino**

Here's a photo of the LilyPad Arduino connected to a regular Arduino. The regular Arduino has its ATmega8 removed and is being used for its USB connection and power source.

to:

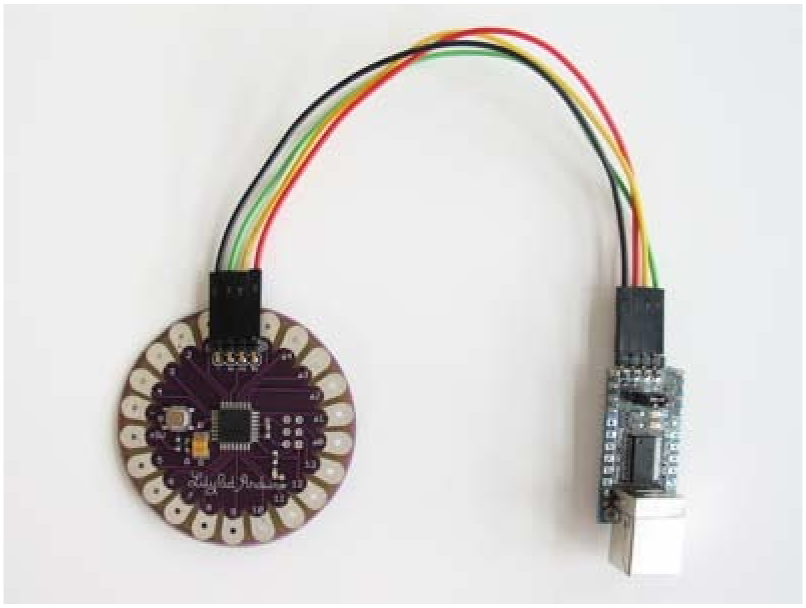
**Connecting the LilyPad Arduino and Arduino NG**

To connect the LilyPad Arduino to your computer via an Arduino NG, remove the ATmega8 or ATmega168 from the NG and then use jumper wires and alligator clips to attach the LilyPad Arduino to the TX, RX, +, and - pins on the NG. Here are photos of the LilyPad Arduino connected to a serial Arduino.

Added line 48:
Restore
September 27, 2007, at 05:45 PM by Leah Buechley -
Added lines 23-28:

Clip the legs off of the Mini USB Adapter and solder alligator clips to the TX, RX, +, and - pins on the front of the board. I use red for +, black for -, green for TX and yellow for RX.

To attach an alligator clip, cut it in half, strip the insulation off the wire from one of the halves and solder the wire to the Mini USB Adapter. Once all four clips are soldered on, use a hot glue gun to prevent the solder joints from breaking when the wires move. Here is a photo showing the modified Mini USB Adapter.



Restore
September 27, 2007, at 05:26 PM by Leah Buechley -
Changed lines 5-6 from:

The LilyPad Arduino is more **fragile and easy to break** than a regular Arduino board. Don't connect more than 5 volts to the + pin or reverse the power and ground pins of your power supply, or you will very likely kill the ATmega168V on the LilyPad Arduino. You can't remove the ATmega168V, so if you kill it, you need a new LilyPad.

to:

The LilyPad Arduino is more **fragile and easy to break** than a regular Arduino board. Don't connect more than 5 volts to the + tab or reverse the power and ground pins of your power supply, or you will very likely kill the ATmega168V on the LilyPad Arduino. You can't remove the ATmega168V, so if you kill it, you need a new LilyPad.

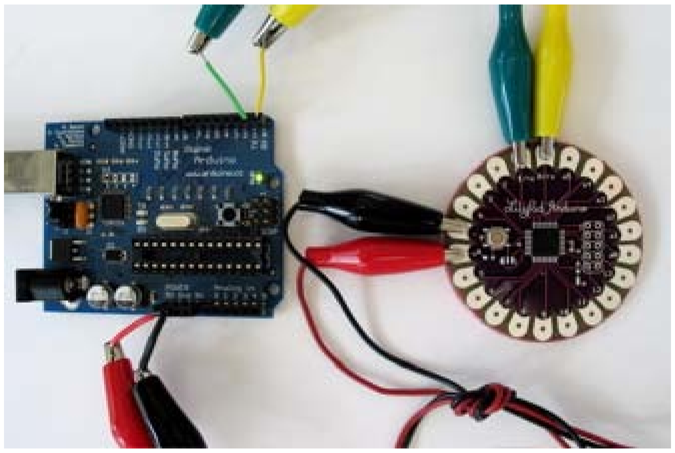
Deleted line 28:
Deleted line 36:
Restore
September 27, 2007, at 05:22 PM by Leah Buechley -
Changed lines 34-35 from:

Here's a photo of the LilyPad Arduino connected to an Arduino NG. The NG has its ATmega8 removed and is being used for its USB connection and power source.

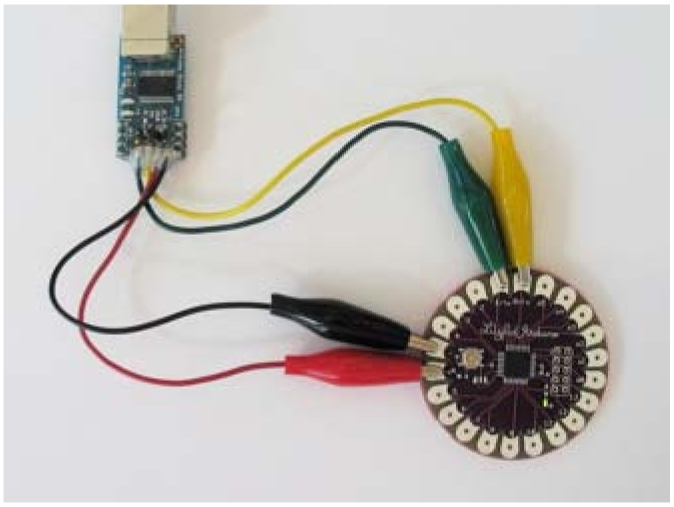
to:

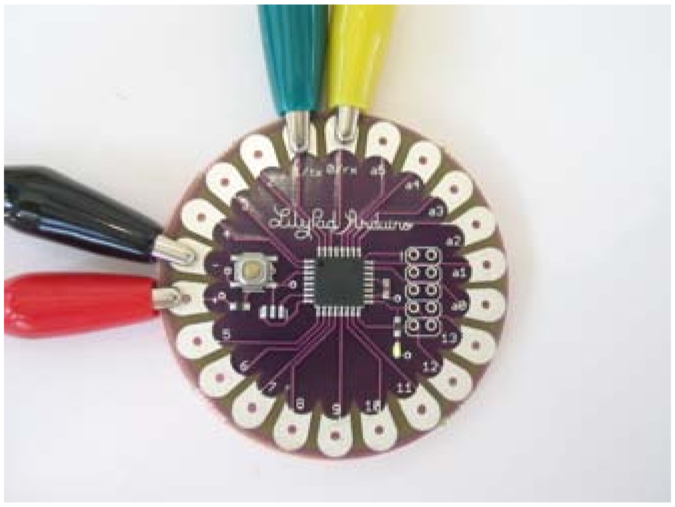
Here's a photo of the LilyPad Arduino connected to a regular Arduino. The regular Arduino has its ATmega8 removed and is being used for its USB connection and power source.

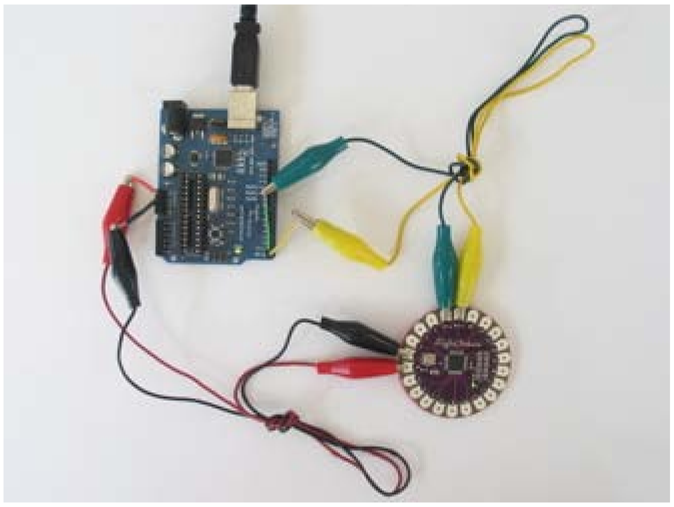
Added lines 37-39:



Restore
September 27, 2007, at 05:08 PM by Leah Buechley -
Added lines 27-31:




Added lines 35-36:


Restore
September 27, 2007, at 04:06 PM by Leah Buechley -
Restore
September 22, 2007, at 11:13 AM by Leah Buechley -
Changed lines 14-17 from:

- Ground. One of the ground pins on the Arduino Mini must be connected to ground of the power source.
- TX/RX. These pins are used both for uploading new sketches to the board and communicating with a computer or other device.

to:

- Ground. The ground tab on the LilyPad Arduino must be connected to ground of the power source.
- TX/RX. These tabs are used both for uploading new sketches to the board and communicating with a computer or other device.

Restore
September 22, 2007, at 11:11 AM by Leah Buechley -
Changed lines 5-6 from:

The Arduino Mini is more **fragile and easy to break** than a regular Arduino board. Don't connect more than 5 volts to the + pin or reverse the power and ground pins of your power supply, or you will very likely kill the ATmega168V on the LilyPad Arduino. You can't remove the ATmega168V, so if you kill it, you need a new LilyPad.

to:

The LilyPad Arduino is more **fragile and easy to break** than a regular Arduino board. Don't connect more than 5 volts to the + pin or reverse the power and ground pins of your power supply, or you will very likely kill the ATmega168V on the LilyPad Arduino. You can't remove the ATmega168V, so if you kill it, you need a new LilyPad.

Restore
September 22, 2007, at 11:08 AM by Leah Buechley -

Changed lines 12-13 from:

- Power. Power should be connected to the + tab on the LilyPad Arduino. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or the + tab of a LilyPad power supply) or, another 3-5V power source (e.g. a 3.7V rechargeable Lithium Ion cell or 2 AA batteries in series).

to:

- Power. Power should be connected to the + tab on the LilyPad Arduino. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or the + tab of a LilyPad power supply) or, another 3-5V power source (e.g. a 3.7V rechargeable Lithium Ion battery or 2 AA batteries in series).

Restore
September 22, 2007, at 11:08 AM by Leah Buechley -
Changed lines 12-13 from:

- Power. Power should be connected to the + tab on the LilyPad Arduino. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or the + tab of the LilyPad power supply) or, another 3-5V power source (e.g. a 3.7V rechargeable Lithium Ion cell or 2 AA batteries in series).

to:

- Power. Power should be connected to the + tab on the LilyPad Arduino. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or the + tab of a LilyPad power supply) or, another 3-5V power source (e.g. a 3.7V rechargeable Lithium Ion cell or 2 AA batteries in series).

Restore
September 22, 2007, at 10:49 AM by Leah Buechley -
Changed lines 12-13 from:

- Power. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or a regular Arduino board) connected to the +5V pin of the Arduino Mini. Or, a 3-5V power source (e.g. the LilyPad power supply or a 3.7V rechargeable Lithium Ion cell) connected to the + tab of the LilyPad Arduino.

to:

- Power. Power should be connected to the + tab on the LilyPad Arduino. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or the + tab of the LilyPad power supply) or, another 3-5V power source (e.g. a 3.7V rechargeable Lithium Ion cell or 2 AA batteries in series).

Restore
September 22, 2007, at 10:45 AM by Leah Buechley -
Changed lines 27-28 from:

**Connecting the Arduino Mini and a regular Arduino**

to:

**Connecting the LilyPad Arduino and a regular Arduino**

Restore
September 22, 2007, at 10:45 AM by Leah Buechley -
Changed lines 12-13 from:

- Power. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or an Arduino NG) connected to the +5V pin of the Arduino Mini. Or, a +3-5V power source (e.g. a 3 volt battery) connected to the + tab of the LilyPad Arduino.

to:

- Power. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or a regular Arduino board) connected to the +5V pin of the Arduino Mini. Or, a 3-5V power source (e.g. the LilyPad power supply or a 3.7V rechargeable Lithium Ion cell) connected to the + tab of the LilyPad Arduino.

Deleted line 20:
Changed line 29 from:

Here's a photo of the LilyPad Arduino connected to an Arduino NG. The NG has its ATmega8 removed and is being used for its USB connection, power source, and reset button. Thus, you can reset the Arduino Mini just by pressing the button on the NG.

to:

Here's a photo of the LilyPad Arduino connected to an Arduino NG. The NG has its ATmega8 removed and is being used for its USB connection and power source.

Restore
September 22, 2007, at 10:41 AM by Leah Buechley -
Changed lines 3-12 from:

To get started with the LilyPad Arduino, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux), with the following modifications:

- Connecting the LilyPad Arduino is a bit more complicated than a regular Arduino board (see below for instructions and photos).

The Arduino Mini is more **fragile and easy to break** than a regular Arduino board.

- Don't connect more than 5 volts to the + pin or reverse the power and ground pins of your power supply, or you will very likely kill the ATmega168V on the LilyPad Arduino.
- You can't remove the ATmega168V, so if you kill it, you need a new LilyPad.

to:

To get started with the LilyPad Arduino, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux. Connecting the LilyPad Arduino is a bit more complicated than a regular Arduino board (see below for instructions and photos).

The Arduino Mini is more **fragile and easy to break** than a regular Arduino board. Don't connect more than 5 volts to the + pin or reverse the power and ground pins of your power supply, or you will very likely kill the ATmega168V on the LilyPad Arduino. You can't remove the ATmega168V, so if you kill it, you need a new LilyPad.

Restore
September 22, 2007, at 10:39 AM by Leah Buechley -
Added lines 1-36:

## Guide to the LilyPad Arduino

To get started with the LilyPad Arduino, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux), with the following modifications:

- Connecting the LilyPad Arduino is a bit more complicated than a regular Arduino board (see below for instructions and photos).

The Arduino Mini is more **fragile and easy to break** than a regular Arduino board.

- Don't connect more than 5 volts to the + pin or reverse the power and ground pins of your power supply, or you will very likely kill the ATmega168V on the LilyPad Arduino.
- You can't remove the ATmega168V, so if you kill it, you need a new LilyPad.

### Connecting the LilyPad Arduino

To use the LilyPad Arduino, you need to connect:

- Power. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or an Arduino NG) connected to the +5V pin of the Arduino Mini. Or, a +3-5V power source (e.g. a 3 volt battery) connected to the + tab of the LilyPad Arduino.
- Ground. One of the ground pins on the Arduino Mini must be connected to ground of the power source.
- TX/RX. These pins are used both for uploading new sketches to the board and communicating with a computer or other device.

You have a few options for connecting the board: the Mini USB Adapter, a regular Arduino board, or your own power supply and USB/Serial adapter.

#### Modifying the Mini USB Adapter to Connect to the LilyPad Arduino

#### Connecting the LilyPad Arduino and Mini USB Adapter

Here is a photo showing the LilyPad Arduino connected to the Mini USB adapter.

**Connecting the Arduino Mini and a regular Arduino**

Here's a photo of the LilyPad Arduino connected to an Arduino NG. The NG has its ATmega8 removed and is being used for its USB connection, power source, and reset button. Thus, you can reset the Arduino Mini just by pressing the button on the NG.

Restore

# Guide to the LilyPad Arduino

To get started with the LilyPad Arduino, follow the directions for the Arduino NG on your operating system (Windows, Mac OS X, Linux. Connecting the LilyPad Arduino is a bit more complicated than a regular Arduino board (see below for instructions and photos).

The LilyPad Arduino is more **fragile and easy to break** than a regular Arduino board. Don't connect more than 5.5 volts to the + tab or reverse the power and ground pins of your power supply, or you will very likely kill the ATmega168V on the LilyPad Arduino. You can't remove the ATmega168V, so if you kill it, you need a new LilyPad.

Note: More information about getting started with the LilyPad Arduino can be found here: http://www.cs.colorado.edu/~buechley/LilyPad

## Connecting the LilyPad Arduino

To program the LilyPad Arduino, you need to connect it to your computer. To do this, you'll need to connect:

- Power. Power should be connected to the + tab on the LilyPad Arduino. This can be a regulated +5V power source (e.g. from the +5V pin of the Mini USB Adapter or the + tab of a LilyPad power supply) or, another 2.7-5.5V power source (e.g. a 3.7V rechargeable Lithium Ion battery or 2 AA batteries in series).

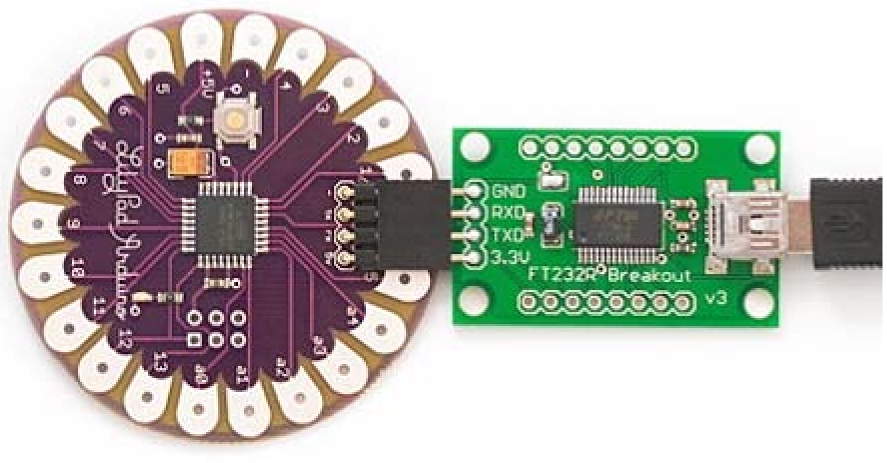
- Ground. The ground tab on the LilyPad Arduino must be connected to ground of the power source.

- TX/RX. These tabs are used both for uploading new sketches to the board and communicating with a computer or other device.

You have a few options for connecting the board to your computer: the SparkFun LilyPad USB Link, the Mini USB Adapter, a regular Arduino board, or your own power supply and USB/Serial adapter.

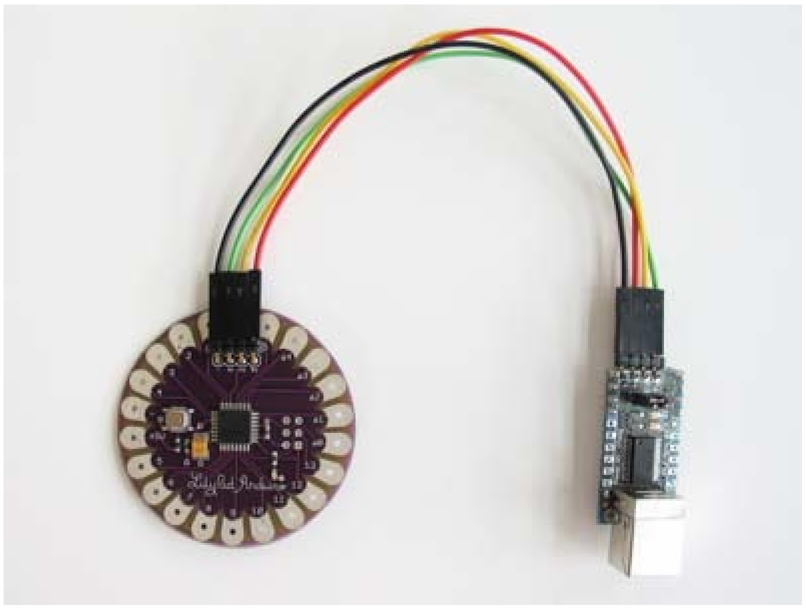
### Use the SparkFun LilyPad USB Link

The SparkFun LilyPad USB Link plugs into the male header pins on the newest version of the LilyPad. If you have an earlier LilyPad version, solder a right angle male header to the -, tx, rx, 5v labeled holes at the top of your LilyPad to make the connection. The LilyPad USB Link is available here and right angle male headers are available here.

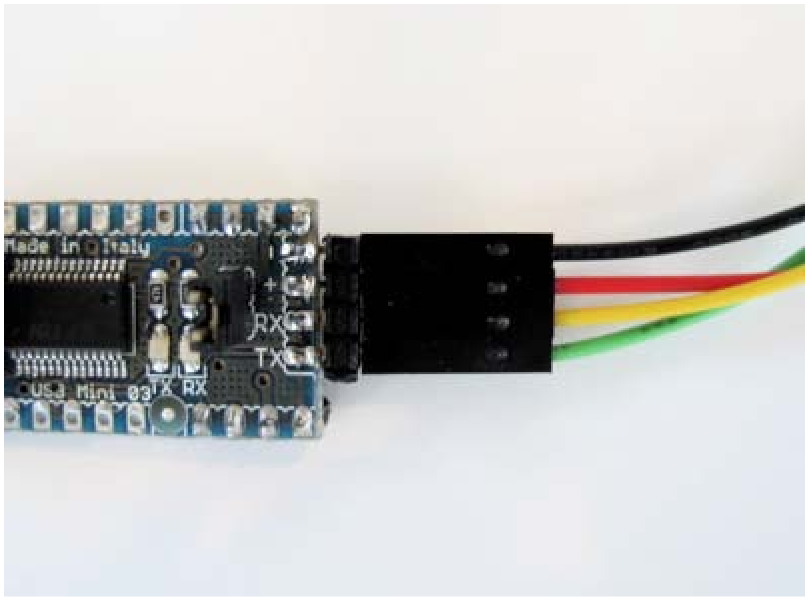


## Modifying the Mini USB Adapter to Connect to the LilyPad Arduino

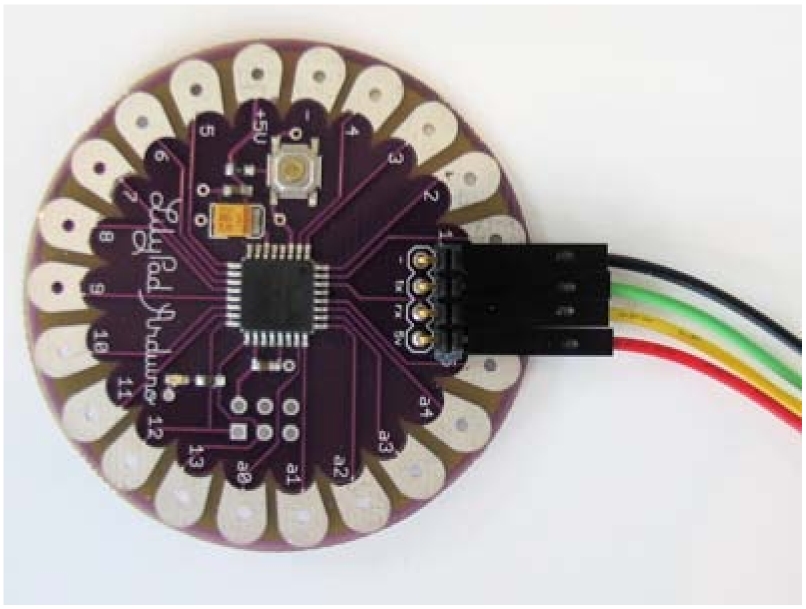
Solder a right angle male header to the Arduino mini USB adapter and then use female-female jumper cables to connect +,-,tx, and rx on the two boards. Right angle male headers are available here and female-female jumper cables are available here. On the version 3 Arduino mini USB adapter you want to connect tx to tx and rx to rx. We're using a red jumper for +, black for -, green for TX and yellow for RX.



Here is a close up view of the miniusb side of the connection:

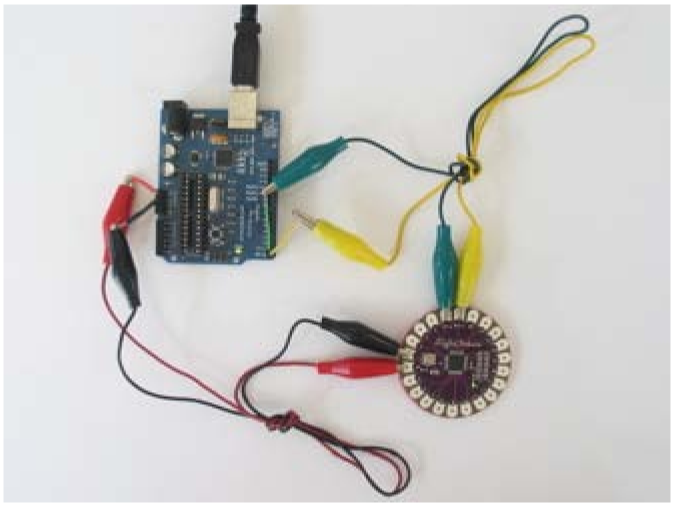


And a close up of the LilyPad side of the connection:



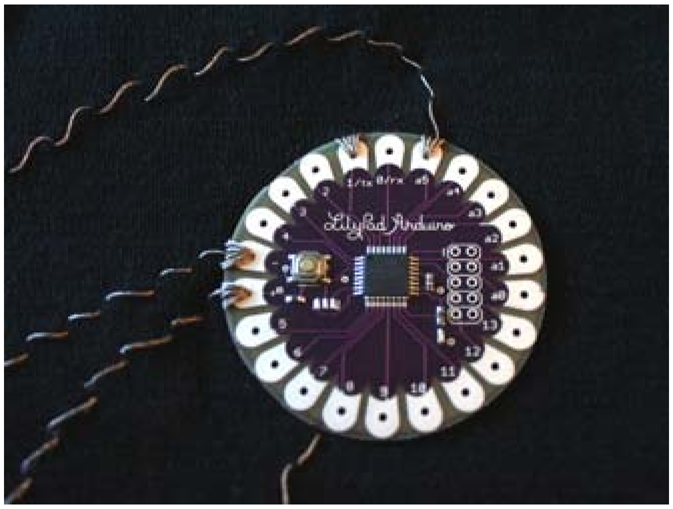
### Connecting the LilyPad Arduino and a regular Arduino

You can also use an Arduino NG to connect the LilyPad Arduino to your computer, using a regular Arduino as a power supply and USB/Serial connection. Just remove the ATmega8 or ATmega168 from the regular Arduino and then use jumper wires and alligator clips to attach the TX, RX, +, and - tabs on the LilyPad to the corresponding pins on the NG. Here's a photo.



### Sewing the LilyPad Arduino

The hole on each tab of the LilyPad is large enough for a sewing needle to pass through. You can make both electrical and physical connections with stitching in conductive thread. Sew through the holes several times to insure good contact. Here's a picture showing a sewn LilyPad:



See the LilyPad Arduino tutorial on Leah's website for more information about building a working wearable. See SparkFun for more stitchable modules that you can use with your LilyPad Arduino.

# Arduino

search

## Login to Arduino

|  |  |
|---|---|
| Username: |  |
| Password: |  |
| Keep me logged in: | ✓ |

Login

## Guide.ArduinoXbeeShield History

Hide minor edits - Show changes to markup

August 15, 2007, at 05:47 PM by David A. Mellis -
Changed lines 3-4 from:

The Arduino Xbee shield allows your Arduino board to communicate wirelessly using Zigbee. It was developed in collaboration with Libelium, a spin-off from the University of Zaragoza in Spain.

to:

The Arduino Xbee shield allows your Arduino board to communicate wirelessly using Zigbee. It was developed in collaboration with Libelium.

Restore
August 15, 2007, at 05:42 PM by David A. Mellis -
Changed lines 3-4 from:

The Arduino Xbee shield allows your Arduino board to communicate wirelessly using Zigbee.

to:

The Arduino Xbee shield allows your Arduino board to communicate wirelessly using Zigbee. It was developed in collaboration with Libelium, a spin-off from the University of Zaragoza in Spain.

Restore
August 15, 2007, at 05:41 PM by David A. Mellis -
Changed lines 97-101 from:

Note that like the other commands, the reset will not be permanent unless you follow it with the **ATWR** comamand.

to:

Note that like the other commands, the reset will not be permanent unless you follow it with the **ATWR** comamand.

### References

For more information, see: the hardware page for the Xbee shield, the Libelium SquidBee wiki, and the MaxStream Xbee page.

Restore
August 15, 2007, at 02:12 PM by David A. Mellis -
Changed lines 52-53 from:

Once in configuration mode, you can send AT commands to the module. Command strings have the form ATxx (where xx is the name of a setting). To read the current value of the setting, send the command string followed by a carriage return. To write a new value to the setting, send the command string, immediately followed by the new setting (with no spaces or newlines in-between), followed by a carriage return. For example, to read the network ID of the module (which determines which other Xbee modules it will communicate with):

to:

Once in configuration mode, you can send AT commands to the module. Command strings have the form ATxx (where xx is the name of a setting). To read the current value of the setting, send the command string followed by a carriage return. To write a new value to the setting, send the command string, immediately followed by the new setting (with no spaces or newlines in-between), followed by a carriage return. For example, to read the network ID of the module (which determines which other Xbee modules it will communicate with), use the *'ATID* command:

Changed lines 79-80 from:

Unless you tell the module to write the changes to non-volatile (long-term) memory, they will only be in effect until the module loses power. To save the changes permanently (until you explicitly modify them again):

to:

Unless you tell the module to write the changes to non-volatile (long-term) memory, they will only be in effect until the module loses power. To save the changes permanently (until you explicitly modify them again), use the **ATWR** command:

Changed lines 88-97 from:
to:

To reset the module to the factory settings, use the **ATRE** command:

(:table width="60%":) (:cellnr width="50%":) Send Command (:cell width="50%":) Expected Response (:cellnr:) ATRE <*enter*> (:cell:) OK<*CR*> (:tableend:)

Note that like the other commands, the reset will not be permanent unless you follow it with the **ATWR** comamand.

Restore
August 15, 2007, at 02:02 PM by David A. Mellis -
Changed lines 43-44 from:

To get the module into configuration mode, you need to send it three plus signs: +++ and there needs to be at least a second before and after when you send no other character to the module. Note that this includes newlines or carriage return characters. Thus, if you're trying to configure the module from the computer, you need to make sure your terminal software is configured to send characters as you type them, without waiting for you to press enter. Otherwise, it will send the plus signs immediately followed by a newline (i.e. you won't get the needed one second delay after the +++). If you successfully enter configuration mode, the module will send back the two characters 'OK', followed by a carriage return.

to:

To get the module into configuration mode, you need to send it three plus signs: +++ and there needs to be at least one second before and after during which you send no other character to the module. Note that this includes newlines or carriage return characters. Thus, if you're trying to configure the module from the computer, you need to make sure your terminal software is configured to send characters as you type them, without waiting for you to press enter. Otherwise, it will send the plus signs immediately followed by a newline (i.e. you won't get the needed one second delay after the +++). If you successfully enter configuration mode, the module will send back the two characters 'OK', followed by a carriage return.

Restore
August 15, 2007, at 02:02 PM by David A. Mellis -
Changed lines 43-44 from:

To get the module into configuration mode, you need to send it three plus signs: + and there needs to be at least a second before and after when you send no other character to the module. Note that this includes newlines or carriage return characters. Thus, if you're trying to configure the module from the computer, you need to make sure your terminal software is configured to send characters as you type them, without waiting for you to press enter. Otherwise, it will send the plus signs immediately followed by a newline (i.e. you won't get the needed one second delay after the +++). If you successfully enter configuration mode, the module will send back the two characters 'OK', followed by a carriage return.

to:

To get the module into configuration mode, you need to send it three plus signs: +++ and there needs to be at least a second before and after when you send no other character to the module. Note that this includes newlines or carriage return characters. Thus, if you're trying to configure the module from the computer, you need to make sure your terminal software is configured to send characters as you type them, without waiting for you to press enter. Otherwise, it will send the plus signs immediately followed by a newline (i.e. you won't get the needed one second delay after the +++). If you successfully enter configuration mode, the module will send back the two characters 'OK', followed by a carriage return.

Restore
August 15, 2007, at 02:01 PM by David A. Mellis -
Changed lines 39-40 from:

## Configuration the Xbee Module

to:

## Configuring the Xbee Module

Restore
August 15, 2007, at 02:00 PM by David A. Mellis -

Added lines 78-88:

Unless you tell the module to write the changes to non-volatile (long-term) memory, they will only be in effect until the module loses power. To save the changes permanently (until you explicitly modify them again):

(:table width="60%":) (:cellnr width="50%":) Send Command (:cell width="50%":) Expected Response (:cellnr:) ATWR <*enter*> (:cell:) OK <*CR*> (:tableend:)

Restore
August 15, 2007, at 01:58 PM by David A. Mellis -
Changed line 45 from:

(:table width="80%":)

to:

(:table width="60%":)

Changed line 54 from:

(:table width="80%":)

to:

(:table width="60%":)

Changed line 63 from:

(:table width="80%":)

to:

(:table width="60%":)

Changed line 72 from:

(:table width="80%":)

to:

(:table width="60%":)

Restore
August 15, 2007, at 01:57 PM by David A. Mellis -
Changed line 49 from:

(:cell:) OK

to:

(:cell:) OK <*CR*>

Changed lines 54-57 from:

| Send Command | Expected Response |
|---|---|
| ATID <*enter*> | 3332 |

to:

(:table width="80%":) (:cellnr width="50%":) Send Command (:cell width="50%":) Expected Response (:cellnr:) ATID <*enter*> (:cell:) 3332 <*CR*> (:tableend:)

Changed lines 63-66 from:

| Send Command | Expected Response |
|---|---|
| ATID3331 <*enter*> | OK |

to:

(:table width="80%":) (:cellnr width="50%":) Send Command (:cell width="50%":) Expected Response (:cellnr:) ATID3331 <*enter*> (:cell:) OK <*CR*> (:tableend:)

Changed lines 72-75 from:

| Send Command | Expected Response |
|---|---|
| ATID <*enter*> | 3331 |

to:

(:table width="80%":) (:cellnr width="50%":) Send Command (:cell width="50%":) Expected Response (:cellnr:) ATID <*enter*> (:cell:) 3331 <*CR*> (:tableend:)

Restore
August 15, 2007, at 01:56 PM by David A. Mellis -
Changed lines 46-53 from:

(:cellnr width="50%":) Send Command (:cell width="50%":) Expected Response (:cellnr:) +++ (:cell:) OK

to:

(:cellnr width="50%":) Send Command (:cell width="50%":) Expected Response (:cellnr:) +++ (:cell:) OK

Restore
August 15, 2007, at 01:55 PM by David A. Mellis -
Added lines 45-57:

(:table width="80%":) (:cellnr width="50%":) Send Command (:cell width="50%":) Expected Response (:cellnr:) +++ (:cell:) OK (:tableend:)

Once in configuration mode, you can send AT commands to the module. Command strings have the form ATxx (where xx is the name of a setting). To read the current value of the setting, send the command string followed by a carriage return. To write a new value to the setting, send the command string, immediately followed by the new setting (with no spaces or newlines in-between), followed by a carriage return. For example, to read the network ID of the module (which determines which other Xbee modules it will communicate with):

Deleted lines 59-64:

| | |
|---|---|
| +++ | OK |

Once in configuration mode, you can send AT commands to the module. Command strings have the form ATxx (where xx is the name of a setting). To read the current value of the setting, send the command string followed by a carriage return. To write a new value to the setting, send the command string, immediately followed by the new setting (with no spaces or newlines in-between), followed by a carriage return. For example, to read the network ID of the module (which determines which other Xbee modules it will communicate with):

| Send Command | Expected Response |
|---|---|

Restore
August 15, 2007, at 01:54 PM by David A. Mellis -
Added line 45:
Added line 51:
Added line 57:
Added line 63:
Restore
August 15, 2007, at 01:53 PM by David A. Mellis -
Changed lines 45-47 from:

| Send Command | Expected Response |
|---|---|
| +++ | OK |

to:

| Send Command | Expected Response |
|---|---|
| +++ | OK |

Changed lines 50-52 from:

| Send Command | Expected Response |
|---|---|
| ATID <*enter*> | 3332 |

to:

| Send Command | Expected Response |
|---|---|
| ATID <*enter*> | 3332 |

Changed lines 55-57 from:

| Send Command | Expected Response |
|---|---|
| ATID3331 <*enter*> | OK |

to:

| Send Command | Expected Response |
|---|---|

| | |
|---|---|
| ATID3331 <*enter*> | OK |

Changed lines 60-62 from:

| Send Command | Expected Response |
|---|---|
| ATID <*enter*> | 3331 |

to:

| Send Command | Expected Response |
|---|---|
| ATID <*enter*> | 3331 |

Restore
August 15, 2007, at 01:49 PM by David A. Mellis -
Changed line 45 from:

| Command Sent | Response Received |
|---|---|

to:

| Send Command | Expected Response |
|---|---|

Changed lines 48-73 from:

Once in configuration mode, you can send AT commands to the module. Command strings have the form ATxx (where xx is the name of a setting). To read the current value of the setting, send the command string followed by a carriage return. To write a new value to the setting, send the command string, immediately followed by the new setting (with no spaces or newlines in-between), followed by a carriage return. For example, to read the network ID of the module (which determines which other Xbee modules it will communicate with), go into command mode and then type:

```
ATID
```

followed by a carriage return (i.e. pressing enter from within a terminal program). You should see:

```
3332
```

(the default ID of the module) or whatever value the ID was last set to.

To change the network ID of the module, type:

```
ATID3331
```

and press enter. You should get back:

```
OK
```

Now, check that the setting has taken effect by again asking for the ID:

```
ATID
```

(and pressing enter). You should see:

```
3331
```

to:

Once in configuration mode, you can send AT commands to the module. Command strings have the form ATxx (where xx is the name of a setting). To read the current value of the setting, send the command string followed by a carriage return. To write a new value to the setting, send the command string, immediately followed by the new setting (with no spaces or newlines in-between), followed by a carriage return. For example, to read the network ID of the module (which determines which other Xbee modules it will communicate with):

| Send Command | Expected Response |
|---|---|
| ATID <*enter*> | 3332 |

To change the network ID of the module:

| Send Command | Expected Response |
|---|---|
| ATID3331 <*enter*> | OK |

Now, check that the setting has taken effect:

| Send Command | Expected Response |
|---|---|
| ATID <*enter*> | 3331 |

Restore

August 15, 2007, at 01:47 PM by David A. Mellis -
Changed lines 46-47 from:

| | |
|---|---|
| +++ | OK@@ |

to:

| | |
|---|---|
| +++ | OK |

Restore
August 15, 2007, at 01:47 PM by David A. Mellis -
Changed lines 43-46 from:

To get the module into configuration mode, you need to send it three plus signs: + and there needs to be at least a second before and after when you send no other character to the module. Note that this includes newlines or carriage return characters. Thus, if you're trying to configure the module from the computer, you need to make sure your terminal software is configured to send characters as you type them, without waiting for you to press enter. Otherwise, it will send the plus signs immediately followed by a newline (i.e. you won't get the needed one second delay after the +++). If you successfully enter configuration mode, the module will send back the two characters 'OK', followed by a carriage return. In a terminal, this will look like this (where the +++ was typed by you, and the OK returned by the module):

```
+++OK
```

to:

To get the module into configuration mode, you need to send it three plus signs: + and there needs to be at least a second before and after when you send no other character to the module. Note that this includes newlines or carriage return characters. Thus, if you're trying to configure the module from the computer, you need to make sure your terminal software is configured to send characters as you type them, without waiting for you to press enter. Otherwise, it will send the plus signs immediately followed by a newline (i.e. you won't get the needed one second delay after the +++). If you successfully enter configuration mode, the module will send back the two characters 'OK', followed by a carriage return.

| Command Sent | Response Received |
|---|---|
| +++ | OK@@ |

Restore
August 15, 2007, at 01:45 PM by David A. Mellis -
Changed lines 41-42 from:

You can configure the Xbee module from the code running on the Arduino board or from software on the computer. To configure it from the Arduino board, you'll need to have the jumpers in the Xbee position. To configure it from the computer, you'll need to have the jumpers in the USB configuration and have removed the microncontroller from your Arduino board.

to:

You can configure the Xbee module from code running on the Arduino board or from software on the computer. To configure it from the Arduino board, you'll need to have the jumpers in the Xbee position. To configure it from the computer, you'll need to have the jumpers in the USB configuration and have removed the microncontroller from your Arduino board.

Restore
August 15, 2007, at 01:44 PM by David A. Mellis -
Changed lines 37-72 from:

To allow your computer to communicate directly with the Xbee shield, connect it to an Arduino board whose microcontroller has been removed and place its jumpers in the USB configuration. Then you can send data to and receive data from the Xbee module from any terminal program. This allows you, for example, to see the data that the module is receiving from other Xbee shields (e.g. to collect sensor data wirelessly from a number of locations).

to:

To allow your computer to communicate directly with the Xbee shield, connect it to an Arduino board whose microcontroller has been removed and place its jumpers in the USB configuration. Then you can send data to and receive data from the Xbee module from any terminal program. This allows you, for example, to see the data that the module is receiving from other Xbee shields (e.g. to collect sensor data wirelessly from a number of locations).

### Configuration the Xbee Module

You can configure the Xbee module from the code running on the Arduino board or from software on the computer. To configure it from the Arduino board, you'll need to have the jumpers in the Xbee position. To configure it from the computer, you'll need to have the jumpers in the USB configuration and have removed the microncontroller from your Arduino board.

To get the module into configuration mode, you need to send it three plus signs: + and there needs to be at least a second before and after when you send no other character to the module. Note that this includes newlines or carriage return characters. Thus, if you're trying to configure the module from the computer, you need to make sure your terminal software is configured to send characters as you type them, without waiting for you to press enter. Otherwise, it will send the plus signs immediately followed by a newline (i.e. you won't get the needed one second delay after the +++). If you successfully enter configuration mode, the module will send back the two characters 'OK', followed by a carriage return. In a terminal, this will look like this (where the +++ was typed by you, and the OK returned by the module):

```
+++OK
```

Once in configuration mode, you can send AT commands to the module. Command strings have the form ATxx (where xx is the name of a setting). To read the current value of the setting, send the command string followed by a carriage return. To write a new value to the setting, send the command string, immediately followed by the new setting (with no spaces or newlines in-between), followed by a carriage return. For example, to read the network ID of the module (which determines which other Xbee modules it will communicate with), go into command mode and then type:

```
ATID
```

followed by a carriage return (i.e. pressing enter from within a terminal program). You should see:

```
3332
```

(the default ID of the module) or whatever value the ID was last set to.

To change the network ID of the module, type:

```
ATID3331
```

and press enter. You should get back:

```
OK
```

Now, check that the setting has taken effect by again asking for the ID:

```
ATID
```

(and pressing enter). You should see:

```
3331
```

Restore
August 12, 2007, at 07:05 PM by David A. Mellis -
Changed line 37 from:

To allow your computer to communicate directly with the Xbee shield, connect it to an Arduino board whose microcontroller has been removed and place its jumpers in the USB configuration. Then you can send data to and receive data from the Xbee module from any terminal program.

to:

To allow your computer to communicate directly with the Xbee shield, connect it to an Arduino board whose microcontroller has been removed and place its jumpers in the USB configuration. Then you can send data to and receive data from the Xbee module from any terminal program. This allows you, for example, to see the data that the module is receiving from other Xbee shields (e.g. to collect sensor data wirelessly from a number of locations).

Restore
August 12, 2007, at 06:59 PM by David A. Mellis -
Changed lines 33-34 from:

You can use any of the standard Arduino serial commands with the Xbee shield. With the shield's jumpers in the Xbee position, the print and println commands will send data over the Xbee shield and the USB connection (i.e. to other Xbee shields and to the computer at the same time). The Xbee module on the shield, however, is set up to work at 9600 baud by default, so unless you reconfigure it, you'll need to make sure you're passing 9600 to the Serial.begin() command in your sketch.

to:

You can use any of the standard Arduino serial commands with the Xbee shield. With the shield's jumpers in the Xbee position, the print and println commands will send data over the Xbee shield and the USB connection (i.e. to other Xbee shields and to the computer at the same time). In this configuration, however, the board will only receive data from the Xbee shield not from the USB connection (you'll need to switch the jumpers to allow the board to receive data from the computer).

The Xbee module on the shield is set up to work at 9600 baud by default, so unless you reconfigure it, you'll need to make sure you're passing 9600 to the Serial.begin() command in your sketch.

Restore
August 12, 2007, at 06:54 PM by David A. Mellis -
Changed line 35 from:
to:

To allow your computer to communicate directly with the Xbee shield, connect it to an Arduino board whose microcontroller has been removed and place its jumpers in the USB configuration. Then you can send data to and receive data from the Xbee module from any terminal program.

Restore
August 12, 2007, at 06:40 PM by David A. Mellis -
Added lines 31-35:

### A Few Notes

You can use any of the standard Arduino serial commands with the Xbee shield. With the shield's jumpers in the Xbee position, the print and println commands will send data over the Xbee shield and the USB connection (i.e. to other Xbee shields and to the computer at the same time). The Xbee module on the shield, however, is set up to work at 9600 baud by default, so unless you reconfigure it, you'll need to make sure you're passing 9600 to the Serial.begin() command in your sketch.

Restore
August 12, 2007, at 06:32 PM by David A. Mellis -
Changed lines 28-30 from:

When it's finished uploading, you can check that it's working with the Arduino serial monitor. You should see H's and L's arriving one a second. Turn off the serial monitor and unplug the board. Switch the jumpers to the Xbee setting. Now connect both boards to the computer. After a few seconds, you should see the LED on the first board turn on and off, once a second. (This is the LED on the Arduino board itself, not the one on the Xbee shield, which conveys information about the state of the Xbee module.) If so, congratulations, your Arduino boards are communicating wirelessly.

to:

When it's finished uploading, you can check that it's working with the Arduino serial monitor. You should see H's and L's arriving one a second. Turn off the serial monitor and unplug the board. Switch the jumpers to the Xbee setting. Now connect both boards to the computer. After a few seconds, you should see the LED on the first board turn on and off, once a second. (This is the LED on the Arduino board itself, not the one on the Xbee shield, which conveys information about the state of the Xbee module.) If so, congratulations, your Arduino boards are communicating wirelessly. This may not seem that exciting when both boards are connected to the same computer, but if you connect them to different computers (or power them with an external power supply - being sure to switch the power jumper on the Arduino board), they should still be able to communicate.

Restore
August 12, 2007, at 06:30 PM by David A. Mellis -
Added lines 1-28:

## Arduino Xbee Shield

The Arduino Xbee shield allows your Arduino board to communicate wirelessly using Zigbee.

### A Simple Example

You should be able to get two Arduino boards with Xbee shields talking to each other without any configuration, using just the standard Arduino serial commands (described in the reference).

To upload a sketch to an Arduino board with a Xbee shield, you'll need to put both jumpers on the shield to the "USB" setting (i.e. place them on the two pins closest to the edge of the board) or remove them completely (but be sure not to lose them!). Then, you can upload a sketch normally from the Arduino environment. In this case, upload the **Communication | Physical Pixel** sketch to one of the boards. This sketch instructs the board to turn on the LED attached to pin 13 whenever it receives an 'H' over its serial connection, and turn the LED off when it gets an 'L'. You can test it by connecting to the board with the Arduino serial monitor (be sure it's set at 9600 baud), typing an H, and pressing enter (or clicking send). The LED should turn on. Send an L and the LED should turn off. If nothing happens, you may have an Arduino board that doesn't have a built-in LED on pin 13 (see the board index to check for sure), in this case you'll need to supply your own.

Once you've uploaded the Physical Pixel sketch and made sure that it's working, unplug the first Arduino board from the computer. Switch the jumpers to the Xbee setting (i.e. place each on the center pin and the pin farthest from the edge of the board). Now, you need to upload a sketch to the other board. Make sure its jumpers are in the USB setting. Then upload the following sketch to the board:

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.print('H');
  delay(1000);
  Serial.print('L');
  delay(1000);
}
```

When it's finished uploading, you can check that it's working with the Arduino serial monitor. You should see H's and L's arriving one a second. Turn off the serial monitor and unplug the board. Switch the jumpers to the Xbee setting. Now connect both boards to the computer. After a few seconds, you should see the LED on the first board turn on and off, once a second. (This is the LED on the Arduino board itself, not the one on the Xbee shield, which conveys information about the state of the Xbee module.) If so, congratulations, your Arduino boards are communicating wirelessly.

Restore

# Arduino Xbee Shield

The Arduino Xbee shield allows your Arduino board to communicate wirelessly using Zigbee. It was developed in collaboration with Libelium.

## A Simple Example

You should be able to get two Arduino boards with Xbee shields talking to each other without any configuration, using just the standard Arduino serial commands (described in the reference).

To upload a sketch to an Arduino board with a Xbee shield, you'll need to put both jumpers on the shield to the "USB" setting (i.e. place them on the two pins closest to the edge of the board) or remove them completely (but be sure not to lose them!). Then, you can upload a sketch normally from the Arduino environment. In this case, upload the **Communication | Physical Pixel** sketch to one of the boards. This sketch instructs the board to turn on the LED attached to pin 13 whenever it receives an 'H' over its serial connection, and turn the LED off when it gets an 'L'. You can test it by connecting to the board with the Arduino serial monitor (be sure it's set at 9600 baud), typing an H, and pressing enter (or clicking send). The LED should turn on. Send an L and the LED should turn off. If nothing happens, you may have an Arduino board that doesn't have a built-in LED on pin 13 (see the board index to check for sure), in this case you'll need to supply your own.

Once you've uploaded the Physical Pixel sketch and made sure that it's working, unplug the first Arduino board from the computer. Switch the jumpers to the Xbee setting (i.e. place each on the center pin and the pin farthest from the edge of the board). Now, you need to upload a sketch to the other board. Make sure its jumpers are in the USB setting. Then upload the following sketch to the board:

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.print('H');
  delay(1000);
  Serial.print('L');
  delay(1000);
}
```

When it's finished uploading, you can check that it's working with the Arduino serial monitor. You should see H's and L's arriving one a second. Turn off the serial monitor and unplug the board. Switch the jumpers to the Xbee setting. Now connect both boards to the computer. After a few seconds, you should see the LED on the first board turn on and off, once a second. (This is the LED on the Arduino board itself, not the one on the Xbee shield, which conveys information about the state of the Xbee module.) If so, congratulations, your Arduino boards are communicating wirelessly. This may not seem that exciting when both boards are connected to the same computer, but if you connect them to different computers (or power them with an external power supply - being sure to switch the power jumper on the Arduino board), they should still be able to communicate.

## A Few Notes

You can use any of the standard Arduino serial commands with the Xbee shield. With the shield's jumpers in the Xbee position, the print and println commands will send data over the Xbee shield and the USB connection (i.e. to other Xbee shields and to the computer at the same time). In this configuration, however, the board will only receive data from the Xbee shield not from the USB connection (you'll need to switch the jumpers to allow the board to receive data from the computer).

The Xbee module on the shield is set up to work at 9600 baud by default, so unless you reconfigure it, you'll need to make sure you're passing 9600 to the Serial.begin() command in your sketch.

To allow your computer to communicate directly with the Xbee shield, connect it to an Arduino board whose microcontroller has been removed and place its jumpers in the USB configuration. Then you can send data to and receive data from the Xbee module from any terminal program. This allows you, for example, to see the data that the module is receiving from other Xbee shields (e.g. to collect sensor data wirelessly from a number of locations).

## Configuring the Xbee Module

You can configure the Xbee module from code running on the Arduino board or from software on the computer. To configure it from the Arduino board, you'll need to have the jumpers in the Xbee position. To configure it from the computer, you'll need to have the jumpers in the USB configuration and have removed the microncontroller from your Arduino board.

To get the module into configuration mode, you need to send it three plus signs: +++ and there needs to be at least one second before and after during which you send no other character to the module. Note that this includes newlines or carriage return characters. Thus, if you're trying to configure the module from the computer, you need to make sure your terminal software is configured to send characters as you type them, without waiting for you to press enter. Otherwise, it will send the plus signs immediately followed by a newline (i.e. you won't get the needed one second delay after the +++). If you successfully enter configuration mode, the module will send back the two characters 'OK', followed by a carriage return.

| Send Command | Expected Response |
|---|---|
| `+++` | `OK`<*CR*> |

Once in configuration mode, you can send AT commands to the module. Command strings have the form ATxx (where xx is the name of a setting). To read the current value of the setting, send the command string followed by a carriage return. To write a new value to the setting, send the command string, immediately followed by the new setting (with no spaces or newlines in-between), followed by a carriage return. For example, to read the network ID of the module (which determines which other Xbee modules it will communicate with), use the *'ATID* command:

| Send Command | Expected Response |
|---|---|
| `ATID`<*enter*> | `3332`<*CR*> |

To change the network ID of the module:

| Send Command | Expected Response |
|---|---|
| `ATID3331`<*enter*> | `OK`<*CR*> |

Now, check that the setting has taken effect:

| Send Command | Expected Response |
|---|---|
| `ATID`<*enter*> | `3331`<*CR*> |

Unless you tell the module to write the changes to non-volatile (long-term) memory, they will only be in effect until the module loses power. To save the changes permanently (until you explicitly modify them again), use the **ATWR** command:

| Send Command | Expected Response |
|---|---|
| `ATWR`<*enter*> | `OK`<*CR*> |

To reset the module to the factory settings, use the **ATRE** command:

| Send Command | Expected Response |
|---|---|
| `ATRE`<*enter*> | `OK`<*CR*> |

Note that like the other commands, the reset will not be permanent unless you follow it with the **ATWR** comamand.

## References

For more information, see: the hardware page for the Xbee shield, the Libelium SquidBee wiki, and the MaxStream Xbee page.

# Arduino

search

## Login to Arduino

Username:

Password:

Keep me logged in: ✓

Login

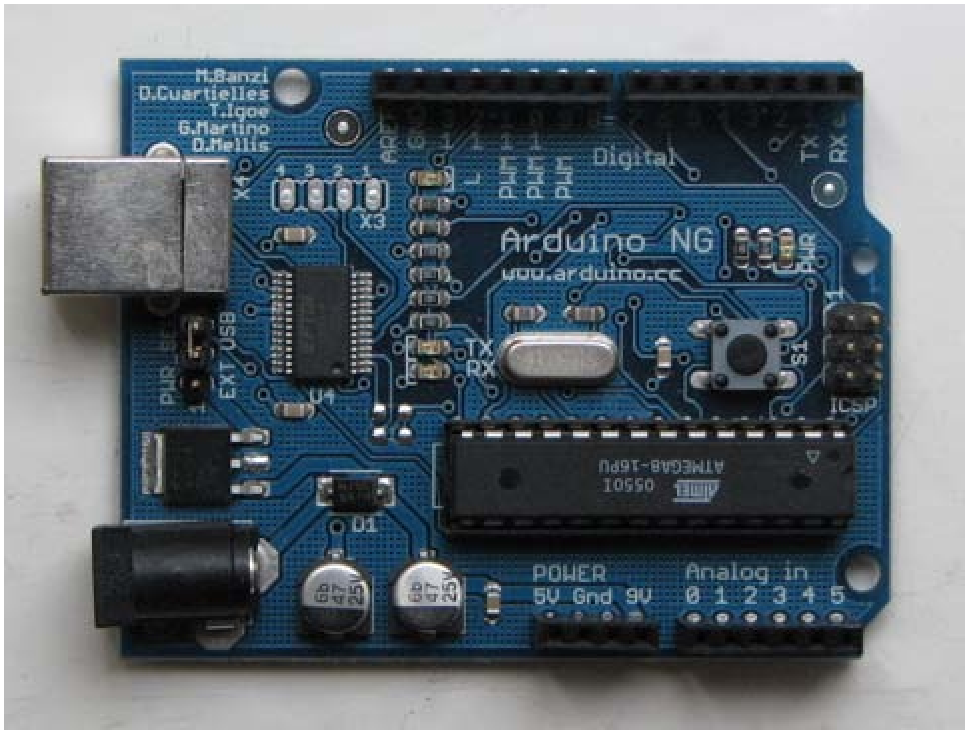
## Arduino Howto

These are the steps you need to follow in order to be up and running:

1. Get an Arduino board
2. Download the Arduino environment
3. Install the USB drivers
4. Connect the board
5. Upload a program

### 1 | Get an Arduino board

The Arduino i/o board is a simple circuit featuring the ATmega8 processor from Atmel. The board is composed of a printed circuit board (PCB) and electronic parts.



There are a few ways to get an Arduino board:

- **buy a ready made board**. See how you can buy a board or just the PCB.
  - European distributor
  - US distributor
- **build your own board**. If you want you can build your own PCB just by downloading the CAD files from the Hardware page. Extract the .brd file and send it to a PCB manufacturer. Be aware that manufacturing a single pcb will be very expensive. It's better to get together with other people and make 20 or 30 at a time. Since you get the full CAD files you can make your own customised version of Arduino. if you make modifications or fix bugs please send us your changes!
  - **purchase parts**. purchase the parts from any electronics store. The Serial version in particular has been designed to use the most basic parts that can be found anywhere in the world. The USB version on the other hand requires some advanced soldering skills because of the FTDI chip that is an smd part. Here is a list? of

parts for the serial board.

- **assemble the board**. We put together a step by step guide on how to build an arduino board. *Newbies: never soldered before? afraid of trashing thousands of boards before getting one properly soldered? fear not :) learn to master the art of soldering.*
- **program the bootloader**. In order for the development environment to be able to program the chip, this has to be programmed with a piece of code called *bootloader*. See the bootloader page on how to program it on your chip.

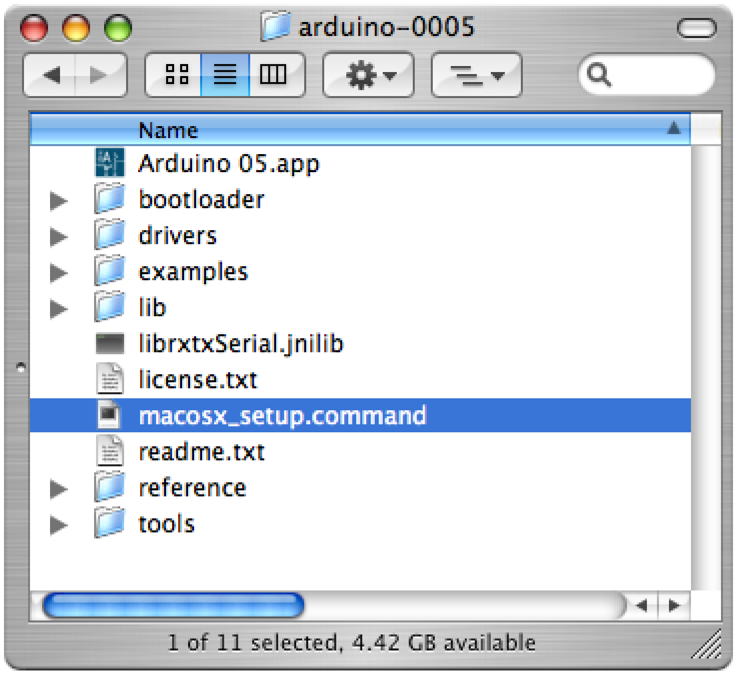
## 2 | Download the Arduino environment

To program the Arduino board you need the Arduino environment.

**Download Arduino**: From the software page.

*Linux note*: For help getting the Arduino IDE running on Debian, please see the FAQ#linux ("How can I run the Arduino IDE under Linux?").

*Mac OS X note:* After downloading the IDE, run the `macosx_setup.command`. It corrects permission on a few files for use with the serial port and will prompt you for your password. You may need to reboot after running this script.

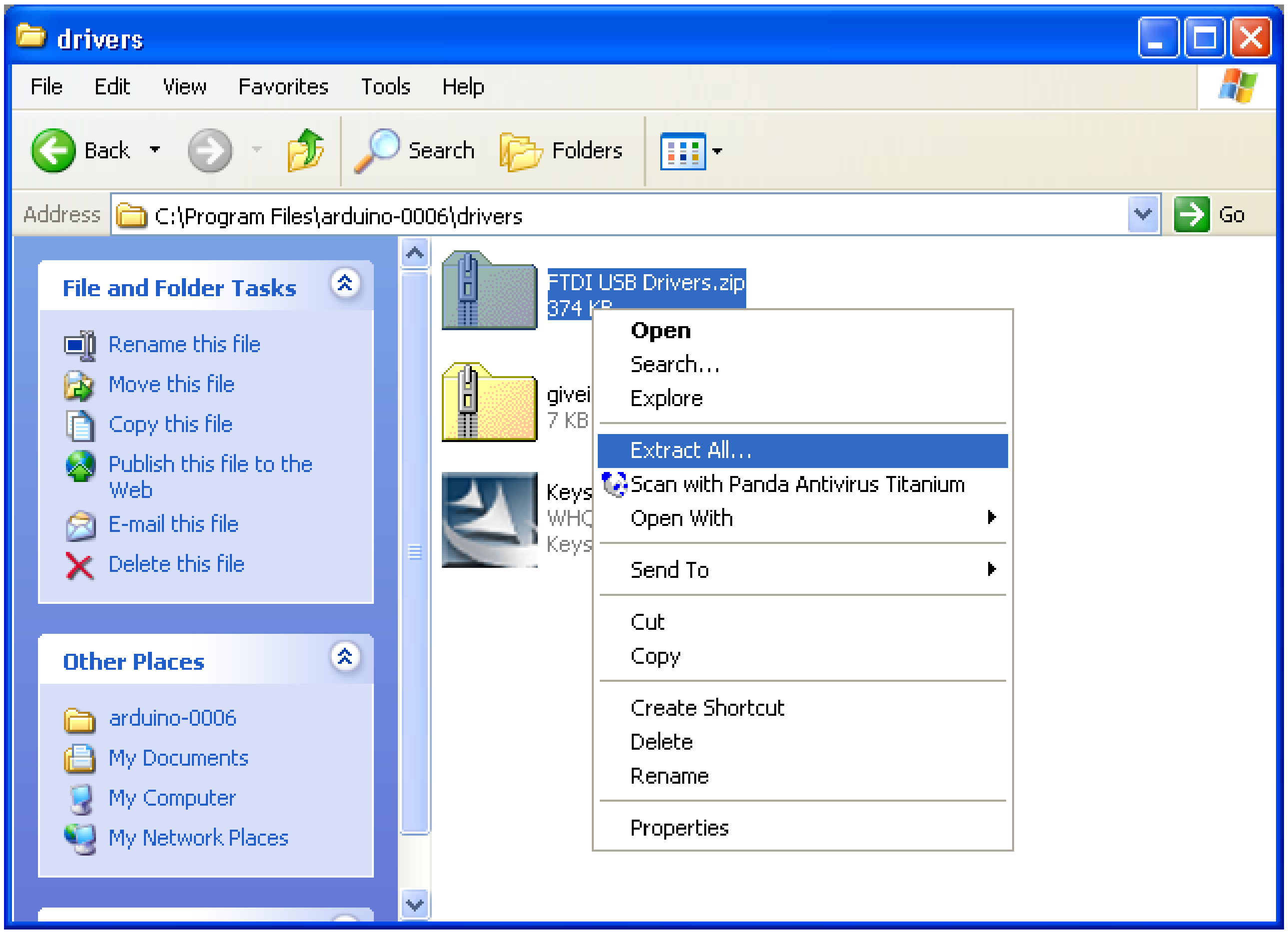


For more information, see the guide to the Arduino environment.

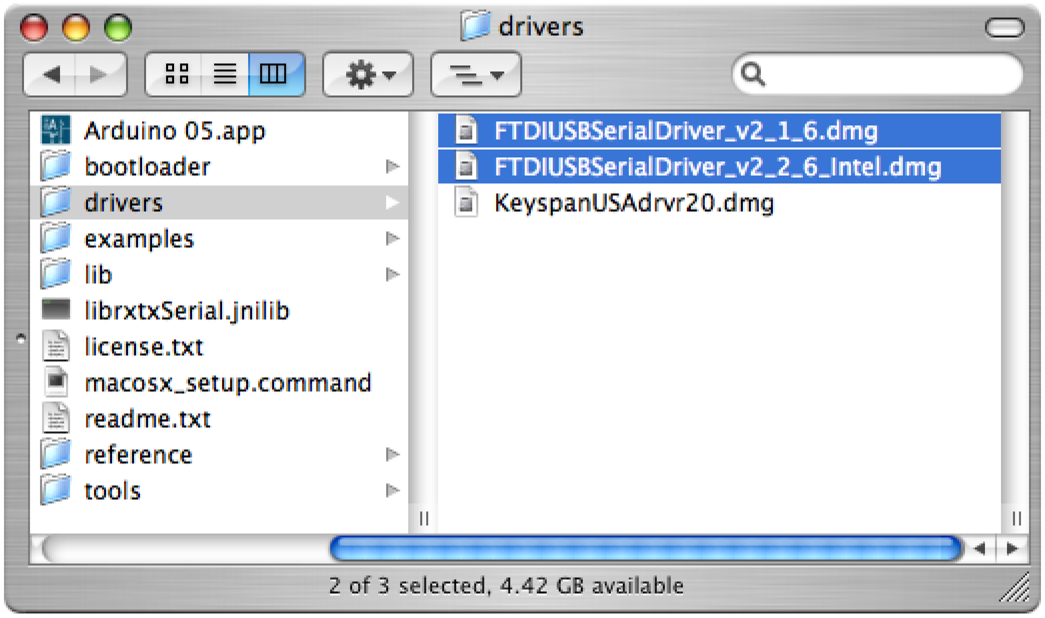
## 3 | Install the USB drivers

If you are using a USB Arduino, you will need to install the drivers for the FTDI chip on the board. These can be found in the `drivers` directory of the Arduino distribution.

On Windows, you will need to unzip `FTDI USB Drivers.zip`. Then, when you plug in the Arduino board, point the Windows *Add Hardware* wizard to the `FTDI USB Drivers` directory.



On the Mac, mount the `FTDIUSBSerialDriver_v2_1_6.dmg` (on PPC machines) or the `FTDIUSBSerialDriver_v2_2_6_Intel.dmg` (on Intel machines) disk image and run the included `FTDIUSBSerialDriver.pkg`.



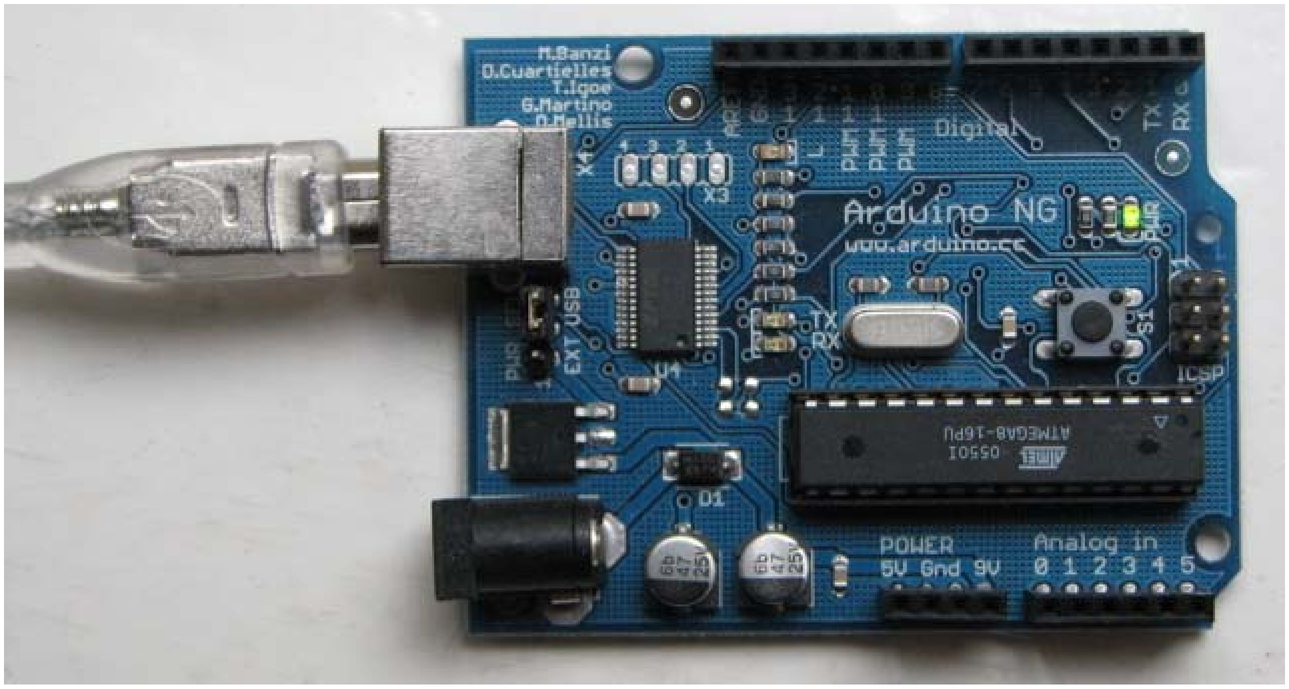
The latest version of the drivers can be found on the FTDI website.

### 4 | Connect the board

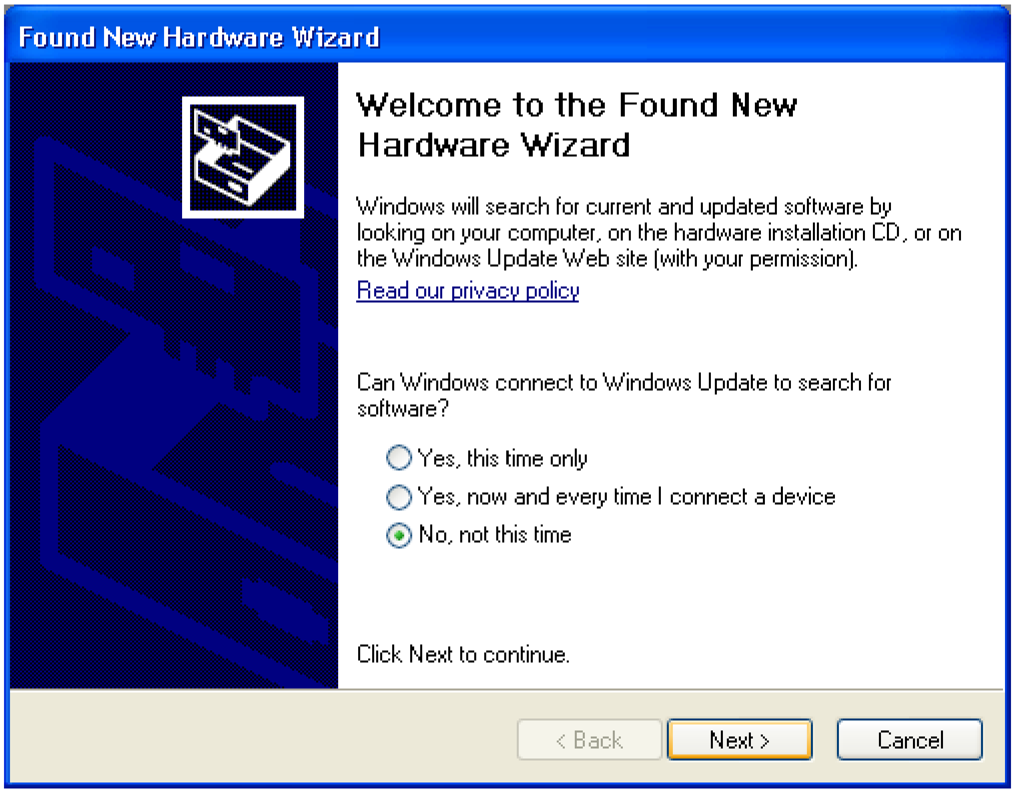
If you're using a serial board, power the board with an external power supply (6 to 25 volts DC, with the core of the connector positive). Connect the board to a serial port on your computer.

On the USB boards, the power source is selected by the jumper between the USB and power plugs. To power the board from the USB port (good for controlling low power devices like LEDs), place the jumper on the two pins closest to the USB plug. To power the board from an external power supply (needed for motors and other high current devices), place the jumper on the two pins closest to the power plug. Either way, connect the board to a USB port on your computer.

The power LED should go on.



On Windows, the Add New Hardware wizard will open. Tell it not to connect to Windows update and click next.



Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

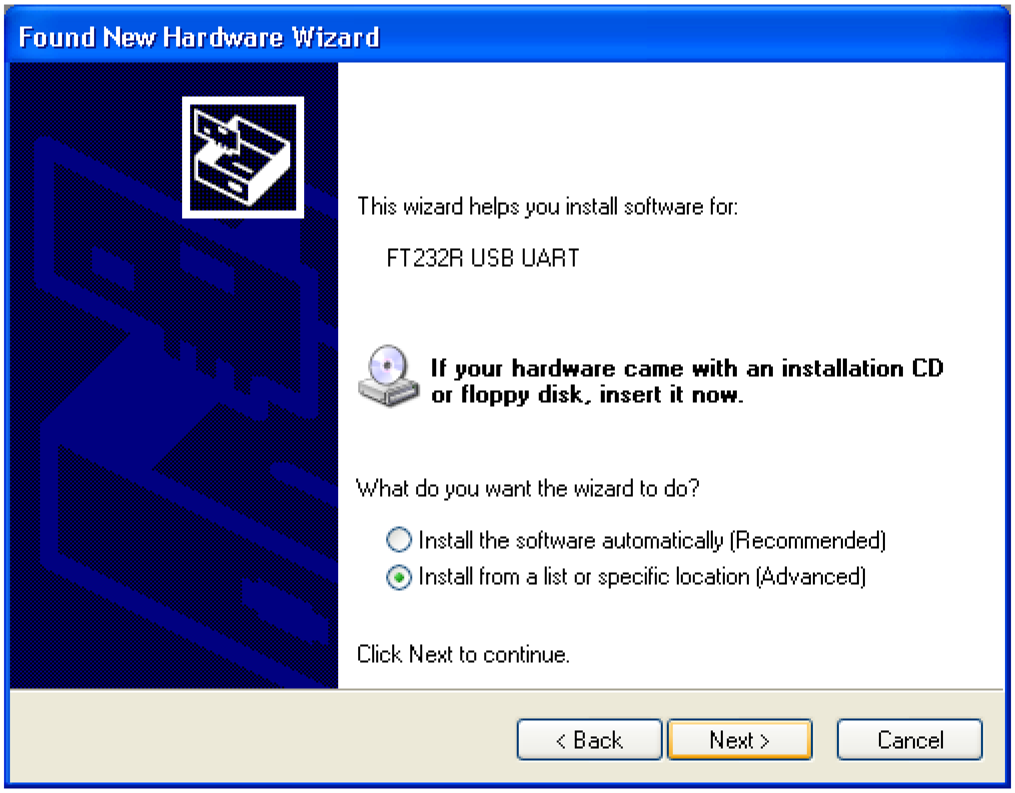
Read our privacy policy

Can Windows connect to Windows Update to search for software?

- ○ Yes, this time only
- ○ Yes, now and every time I connect a device
- [✓ No, not this time]

Click Next to continue.

Then select "Install from a list or specified location (Advanced)" and click next.



Found New Hardware Wizard

This wizard helps you install software for:

FT232R USB UART

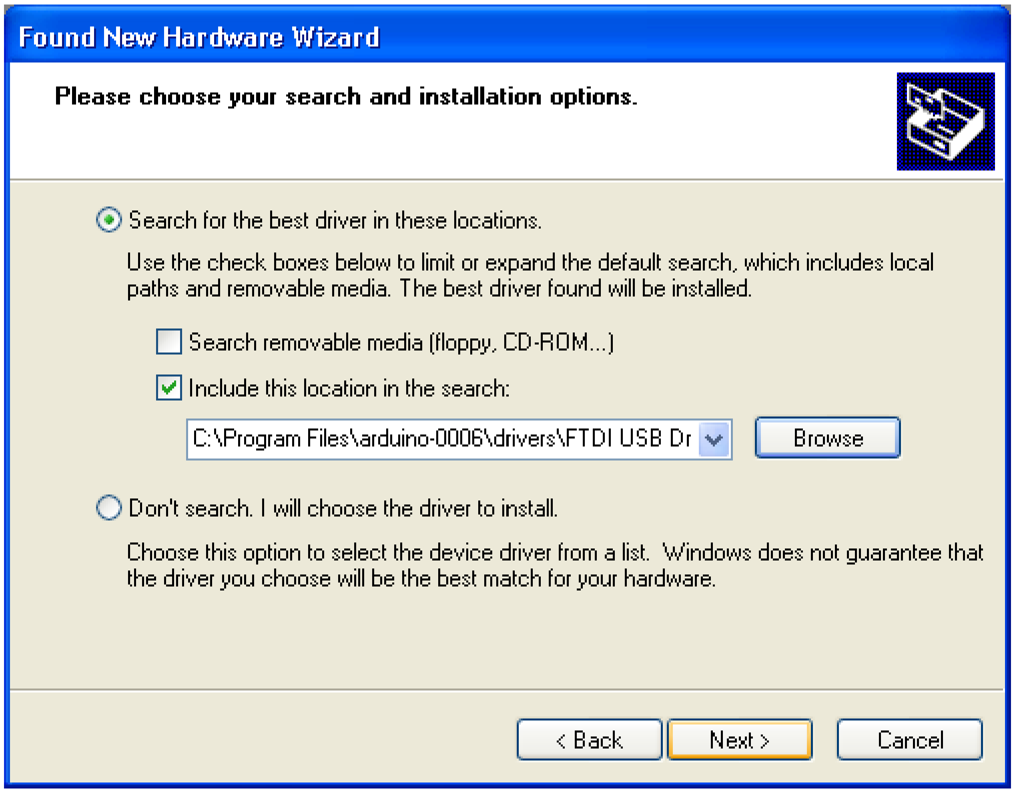
**If your hardware came with an installation CD or floppy disk, insert it now.**

What do you want the wizard to do?

- ○ Install the software automatically (Recommended)
- ◉ Install from a list or specific location (Advanced)

Click Next to continue.

Make sure that "Search for the best driver in these locations is checked"; uncheck "Search removable media"; check "Include this location in the search" and browse to the location you unzipped the USB drivers to in the previous step. Click next.



Found New Hardware Wizard

**Please choose your search and installation options.**

◉ Search for the best driver in these locations.

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

☐ Search removable media (floppy, CD-ROM...)

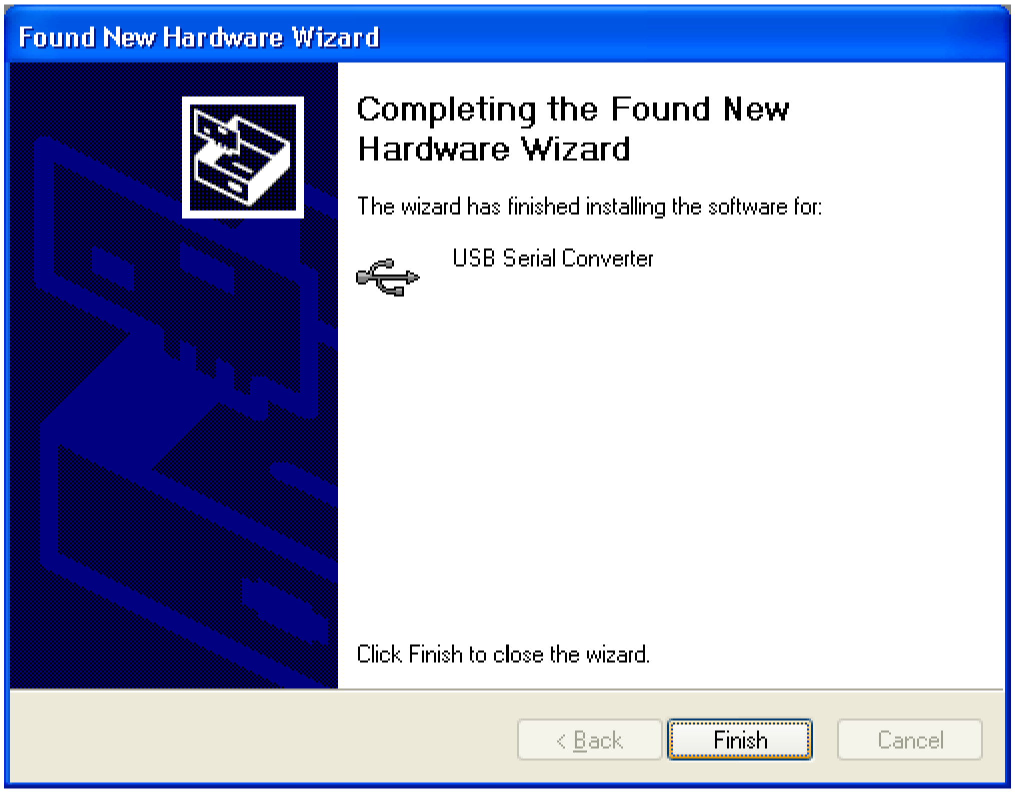
☑ Include this location in the search:

C:\Program Files\arduino-0006\drivers\FTDI USB Dr

○ Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

The wizard will search for the driver and then tell you that a "USB Serial Converter" was found. Click finish.


Found New Hardware Wizard

Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

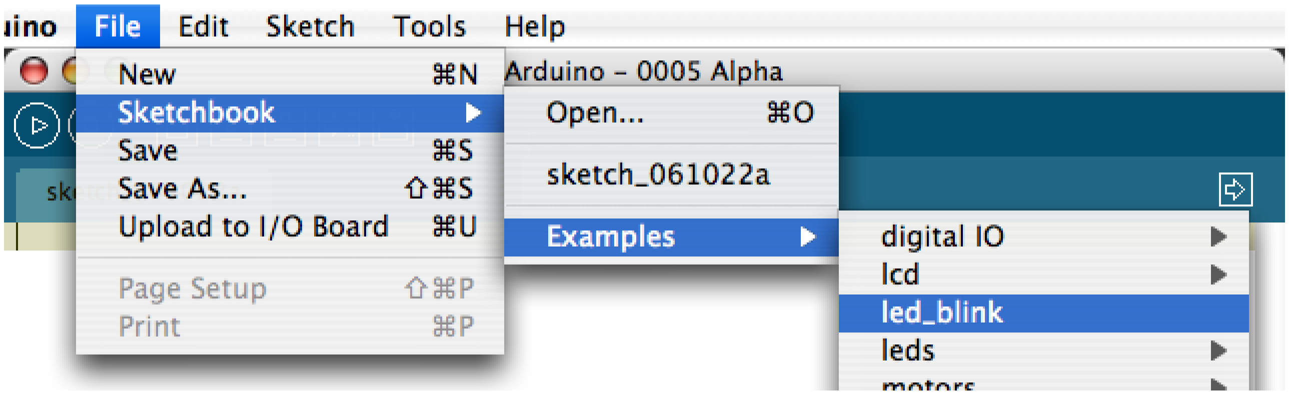
USB Serial Converter

Click Finish to close the wizard.

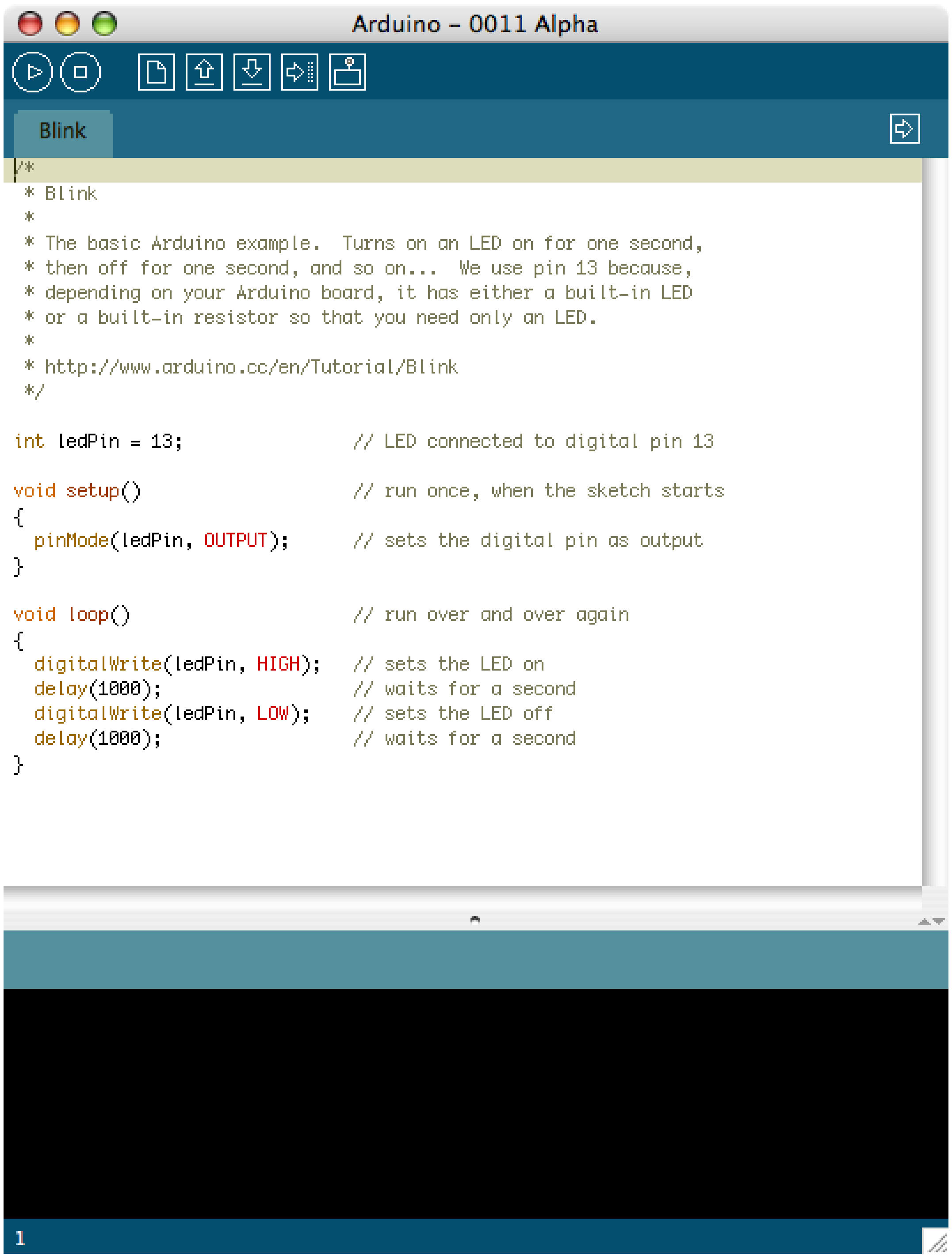
The new hardware wizard will appear again. Go through the same steps. This time, a "USB Serial Port" will be found.

**5 | Upload a program**

Open the LED blink example sketch: File > Sketchbook > Examples > led_blink.

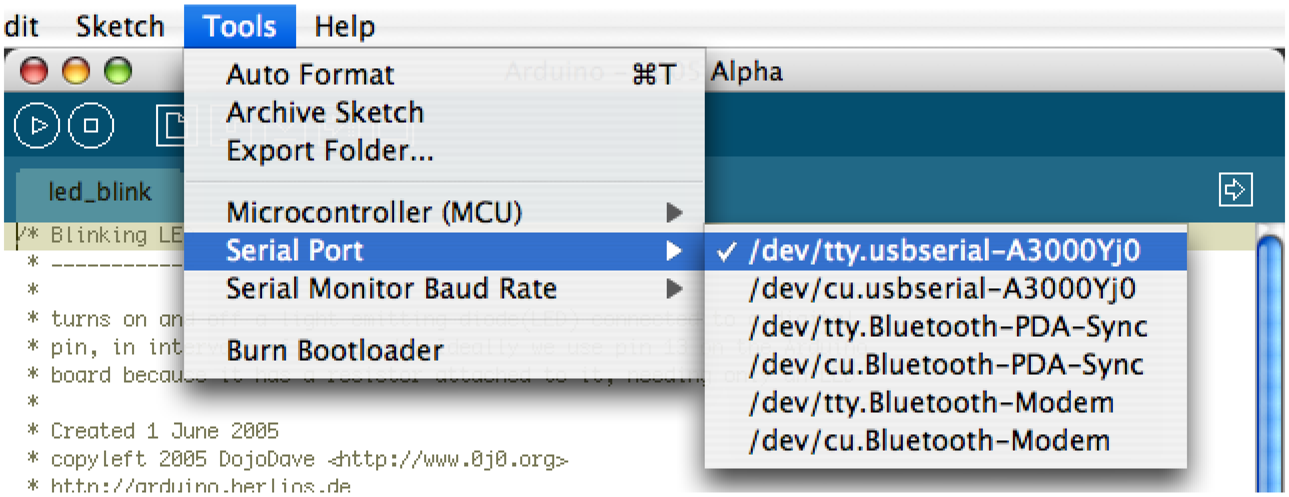


Here's what the code for the LED blink example looks like.



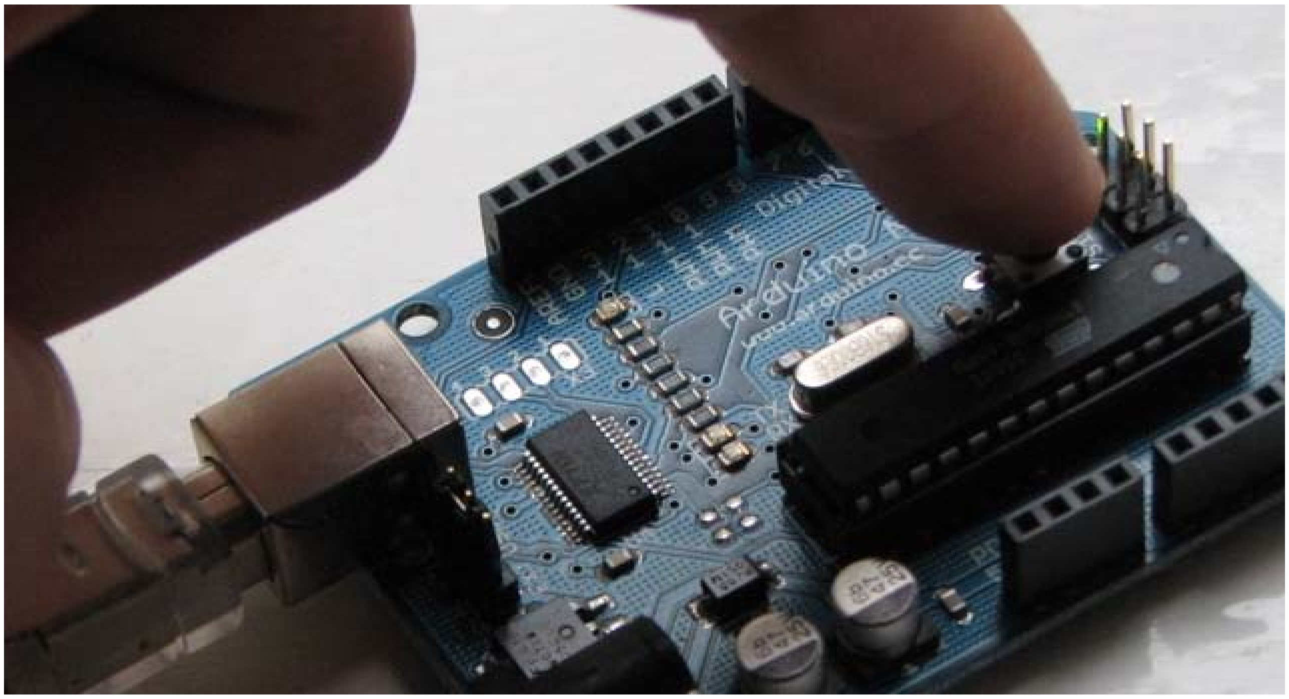
```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

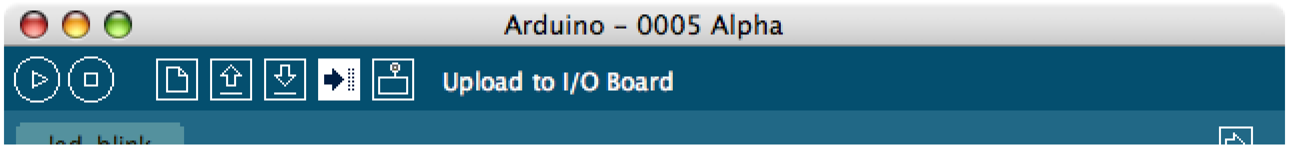
Select the serial device of the Arduino board from the Tools | Serial Port menu. On Windows, this should be `COM1` or `COM2` for a serial Arduino board, or `COM3`, `COM4`, or `COM5` for a USB board. On the Mac, this should be something like `/dev/cu.usbserial-1B1` for a USB board, or something like `/dev/cu.USA19QW1b1P1.1` if using a Keyspan adapter with a serial board (other USB-to-serial adapters use different names).



```
/* Blinking LE
 * -----------
 *
 * turns on an
 * pin, in int
 * board becau
 *
 * Created 1 June 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
```

Push the reset button on the board then click the *Upload* button in the IDE. Wait a few seconds. If successful, the message "Done uploading." will appear in the status bar.





If the Arduino board doesn't show up in the Tools | Serial Port menu, or you get an error while uploading, please see the troubleshooting suggestions.

A few seconds after the upload finishes, you should see the amber (yellow) LED on the board start to blink.

## Learn More

- Read about the Arduino Environment
- Learn about the parts of the Arduino board
- See the tutorials for some example programs. (There are also some examples available in the examples directory inside the arduino directory.)
- Look up specific Arduino functions and syntax in the reference
- The Arduino programming language is compatible with the Wiring language allowing porting applications from the Wiring board to Arduino. Please note the differences between the Wiring and Processing languages.
- If you're having problems, check the FAQ.
- If you don't find a solution there, try posting in the forums.

## Login to Arduino

| | |
|---|---|
| Username: | |
| Password: | |
| Keep me logged in: | ✓ |

Login

## Guide.Troubleshooting History

Hide minor edits - Show changes to markup

March 31, 2008, at 10:33 AM by David A. Mellis -
Added lines 53-56:

### Why does the Arduino software freeze when I try to upload a program? (on Windows)?

This might be caused by a conflict with the Logitech process 'LVPrcSrv.exe'. Open the Task Manager and see if this program is running, and if so, kill it before attempting the upload. more information

Restore
January 04, 2008, at 10:06 PM by David A. Mellis - expanding the upload troubleshooting procedure.
Added lines 29-32:

- Try uploading with nothing connected to the board (apart from the USB cable, of course).
- Make sure the board isn't touching anything metallic or conductive.

Added lines 43-52:

If it still doesn't work, you can ask for help in the forum. Please include the following information:

- Your operating system.
- What kind of board you have. If it's a Mini, LilyPad or other board that requires extra wiring, include a photo of your circuit, if possible.
- Whether or not you were ever able to upload to the board. If so, what were you doing with the board before / when it stopped working, and what software have you recently added or removed from your computer?
- The messages displayed when you try to upload with verbose output enabled. To do this, you'll need to set upload.verbose to true in your Arduino preferences file.

Restore
November 16, 2007, at 09:07 AM by David A. Mellis - adding some notes about errors caused by the automatic prototype generation.
Added lines 174-181:

### Why do I get errors about undeclared functions or undeclared types?

The Arduino environment attempts to automatically generate prototypes for your functions, so that you can order them as you like in your sketch. This process, however, isn't perfect, and sometimes leads to obscure error messages.

If you declare a custom type in your code and create a function that accepts or returns a value of that type, you'll get an error when you try to compile the sketch. This is because the automatically-generated prototype for that function will appear above the type definition.

If you declare a function with a two-word return type (e.g. "unsigned int") the environment will not realize it's a function and will not create a prototype for it. That means you need to provide your own, or place the definition of the function above any calls to it.

Restore
October 06, 2007, at 01:14 PM by David A. Mellis -
Added line 159:

Restore
September 01, 2007, at 08:49 AM by David A. Mellis -

Changed lines 21-22 from:

- Be sure that you are resetting the board a couple of seconds before uploading.

to:

- Be sure that you are resetting the board a couple of seconds before uploading (unless you have an Arduino Diecimila).
- However, note that some Diecimila were accidently burned with the wrong bootloader and may require you to physically press the reset button before uploading; see this question below.

Added line 48:

Changed lines 51-52 from:

Some of the Arduino Diecimila boards were accidently burned with the Arduino NG bootloader. It should work fine, but has a longer delay when the board is reset (because the NG doesn't have an automatic reset, so you have to time the uploads manually). You can recognize the NG bootloader because the LED on pin 13 will blink three times when you reset the board (as compared to once with the Diecimila bootloader). If your Diecimila has the NG bootloader on it, you may need to physically press the reset button on the board before uploading your sketch.

to:

Some of the Arduino Diecimila boards were accidently burned with the Arduino NG bootloader. It should work fine, but has a longer delay when the board is reset (because the NG doesn't have an automatic reset, so you have to time the uploads manually). You can recognize the NG bootloader because the LED on pin 13 will blink three times when you reset the board (as compared to once with the Diecimila bootloader). If your Diecimila has the NG bootloader on it, you may need to physically press the reset button on the board before uploading your sketch. You can burn the correct bootloader onto your Diecimila, see the bootloader page for details.

Restore
September 01, 2007, at 08:43 AM by David A. Mellis - adding note about Diecimila boards with wrong bootloader
Added lines 46-49:

### Why does my Diecimila take such a long time (6-8 seconds) to start my sketch?

Some of the Arduino Diecimila boards were accidently burned with the Arduino NG bootloader. It should work fine, but has a longer delay when the board is reset (because the NG doesn't have an automatic reset, so you have to time the uploads manually). You can recognize the NG bootloader because the LED on pin 13 will blink three times when you reset the board (as compared to once with the Diecimila bootloader). If your Diecimila has the NG bootloader on it, you may need to physically press the reset button on the board before uploading your sketch.

Restore
August 17, 2007, at 05:15 PM by David A. Mellis - adding a question about Arduino slowness and Tools menu freezing on Windows
Added lines 98-101:

### Why does the Arduino software run really slowly (on Windows)?

If the Arduino software takes a long time to start up and appears to freeze when you try to open the Tools menu, there by a conflict with another device on your system. The Arduino software, on startup and when you open the Tools menu, tries to get a list of all the COM ports on your computer. It's possible that a COM port created by one of the devices on your computer slows down this process. Take a look in the Device Manager. Try disabling the devices that provide COM ports (e.g. Bluetooth devices).

Restore
July 18, 2007, at 09:19 AM by David A. Mellis -
Changed lines 68-69 from:

### What do I do if I get the following error when launching Arduino?

to:

### What do I do if I get an UnsatisfiedLinkError error (about native library librxtxSerial.jnilib) when launching Arduino?

If you get an error like this when launching Arduino:

Changed lines 74-77 from:

You probably have an old version of the communications library lying around. Search for comm.jar or jcl.jar in

/System/Library/Frameworks/JavaVM.framework/ or in directories in your CLASSPATH or PATH environment variables. (reported by Anurag Sehgal)

### What about this error?

to:

you probably have an old version of the communications library lying around. Search for comm.jar or jcl.jar in /System/Library/Frameworks/JavaVM.framework/ or in directories in your CLASSPATH or PATH environment variables. (reported by Anurag Sehgal)

### What about the error "Could not find the main class."?

If you get this error when launching Arduino:

Changed lines 82-83 from:

Make sure that you correctly extracted the contents of the Arduino .zip file - in particular that the **lib** directory is directly inside of the Arduino directory and contains the file **pde.jar**.

to:

make sure that you correctly extracted the contents of the Arduino .zip file - in particular that the **lib** directory is directly inside of the Arduino directory and contains the file **pde.jar**.

Changed lines 115-116 from:

### What if I get this error when uploading code or using the serial monitor (on the Mac)?

to:

### What if I get a gnu.io.PortInUseException when uploading code or using the serial monitor (on the Mac)?

Changed lines 125-126 from:

You need to run the `macosx_setup.command` in the Arduino directory, and then restart your computer. Arduino 0004 includes a modified version of this script that all users need to run (even those who ran the one that came with Arduino 0003). You may also need to delete the contents of the **/var/spool/uucp** directory.

to:

This probably means that the port is actually in use by another application. Please make sure that you're not running other programs that access serial or USB ports, like PDA sync application, bluetooth device managers, certain firewalls, etc. Also, note that some programs (e.g. Max/MSP) keep the serial port open even when not using it - you may to need to close any patches that use the serial port or quit the application entirely.

If you get this error with Arduino 0004 or earlier, or with Processing, you'll need to run the `macosx_setup.command`, and then restart your computer. Arduino 0004 includes a modified version of this script that all users need to run (even those who ran the one that came with Arduino 0003). You may also need to delete the contents of the **/var/spool/uucp** directory.

Added lines 142-143:

Check for a noisy power supply. It's possible this could cause the chip to lose its sketch.

Restore
July 18, 2007, at 09:12 AM by David A. Mellis -
Added lines 5-36:

### Why doesn't my sketch start when I'm powering the board with an external power supply?

Because the RX pin is unconnected, the bootloader on the board may be seeing garbage data coming in, meaning that it never times out and starts your sketch. Try tying the RX pin to ground with a 10K resistor (or connecting it to the TX pin).

### Why I can't upload my programs to the Arduino board?

There are a few things that could be wrong.

- First make sure your board is on (the green LED is on) and connected to the computer (if it's not, see "what if my board doesn't turn on" above).
- Then, check that the proper port is selected in the "Tools | Serial Port" menu (if your port doesn't appear, restart the

IDE with the board connected to the computer).

- Make sure there's a bootloader burned on the Atmega8 on your Arduino board. To check, connect an LED to pin 13 and reset the board. The LED should blink. If it doesn't, see the Bootloader page for instructions on burning a bootloader to the board.
- Be sure that you are resetting the board a couple of seconds before uploading.
- However, on some computers, you may need to press the reset button on the board after you hit the upload button in the Arduino environment. Try different intervals of time between the two, up to 10 seconds or more.
- Disconnect digital pins 0 and 1 while uploading (they can connected and used after the code has been uploaded).
- If you get this error: `[VP 1] Device is not responding correctly.` try uploading again (i.e. reset the board and press the download button a second time).
- Check that you're not running any programs that scan all serial ports, like PDA sync applications, Bluetooth-USB drivers (e.g. BlueSoleil), virtual daemon tools, etc.
- Make sure you don't have firewall software that blocks access to the serial port (e.g. ZoneAlarm).
- You may need to quit Processing, PD, vvvv, etc. if you're using them to read data over the USB or serial connection to the Arduino board.
- If you have a really ancient Arduino board, you may need to change the baud rate at which sketches are uploaded to 9600 (from the normal 19200). You will have to change the speed in the *preferences* file directly. See the preferences page for instructions on finding the file. Look for the file in your computer and change the **serial.download_rate** property to match the one in your board. If you have such a board, it's recommended that you burn the latest bootloader (which works at 19200 baud). This can be done with the *'Tools | Burn Bootloader* menu item.

Changed lines 47-48 from:

### What do to if I get the following error when launching arduino.exe on Windows?

to:

### What should I do if I get an error when launching arduino.exe on Windows?

If you get an error when double-clicking the arduino.exe executable on Windows, for example:

Changed lines 53-54 from:

You'll need to launch Arduino using the run.bat file. Please be patient, the Arduino IDE may take some time to open.

to:

you'll need to launch Arduino using the run.bat file. Please be patient, the Arduino environment may take some time to open.

Deleted lines 110-136:

### Why I can't upload my programs to the Arduino board?

There are a few things that could be wrong.

- First make sure your board is on (the green LED is on) and connected to the computer (if it's not, see "what if my board doesn't turn on" above).
- Then, check that the proper port is selected in the "Tools | Serial Port" menu (if your port doesn't appear, restart the IDE with the board connected to the computer).
- Make sure there's a bootloader burned on the Atmega8 on your Arduino board. To check, connect an LED to pin 13 and reset the board. The LED should blink. If it doesn't, see the Bootloader page for instructions on burning a bootloader to the board.
- Be sure that you are resetting the board a couple of seconds before uploading.
- However, on some computers, you may need to press the reset button on the board after you hit the upload button in the Arduino environment. Try different intervals of time between the two, up to 10 seconds or more.
- Disconnect digital pins 0 and 1 while uploading (they can connected and used after the code has been uploaded).
- If you get this error: `[VP 1] Device is not responding correctly.` try uploading again (i.e. reset the board and press the download button a second time).

- Check that you're not running any programs that scan all serial ports, like PDA sync applications, Bluetooth-USB drivers (e.g. BlueSoleil), virtual daemon tools, etc.
- Make sure you don't have firewall software that blocks access to the serial port (e.g. ZoneAlarm).
- You may need to quit Processing, PD, vvvv, etc. if you're using them to read data over the USB or serial connection to the Arduino board.
- If you have a really ancient Arduino board, you may need to change the baud rate at which sketches are uploaded to 9600 (from the normal 19200). You will have to change the speed in the *preferences* file directly. See the preferences page for instructions on finding the file. Look for the file in your computer and change the **serial.download_rate** property to match the one in your board. If you have such a board, it's recommended that you burn the latest bootloader (which works at 19200 baud). This can be done with the *'Tools* | *Burn Bootloader* menu item.

Deleted lines 126-130:

### Why doesn't my sketch start when I'm powering the board with an external power supply?

Because the RX pin is unconnected, the bootloader on the board may be seeing garbage data coming in, meaning that it never times out and starts your sketch. Try tying the RX pin to ground with a 10K resistor (or connecting it to the TX pin).

Restore
July 18, 2007, at 09:08 AM by David A. Mellis - adding table of contents
Added lines 3-4:

(:*toc:)

Changed lines 6-7 from:

**What if my board doesn't turn on (the green power LED doesn't light up)?**

to:

### What if my board doesn't turn on (the green power LED doesn't light up)?

Changed lines 15-16 from:

**What do to if I get the following error when launching arduino.exe on Windows?**

to:

### What do to if I get the following error when launching arduino.exe on Windows?

Changed lines 21-22 from:

**Why won't Arduino run on old versions of Mac OS X?**

to:

### Why won't Arduino run on old versions of Mac OS X?

Changed lines 34-35 from:

**What do I do if I get the following error when launching Arduino?**

to:

### What do I do if I get the following error when launching Arduino?

Changed lines 40-41 from:

**What about this error?**

to:

### What about this error?

Changed lines 46-47 from:

**What can I do about cygwin conflicts on Windows?**

to:

### What can I do about cygwin conflicts on Windows?

Changed lines 61-62 from:

**Why doesn't my board show in the Tools | Serial Port menu ?**

to:

### Why doesn't my board show in the Tools | Serial Port menu ?

Changed lines 78-79 from:

**Why I can't upload my programs to the Arduino board?**

to:

### Why I can't upload my programs to the Arduino board?

Changed lines 104-105 from:

**What if I get this error when uploading code or using the serial monitor (on the Mac)?**

to:

### What if I get this error when uploading code or using the serial monitor (on the Mac)?

Changed lines 116-117 from:

**I'm having trouble with the FTDI USB drivers.**

to:

### I'm having trouble with the FTDI USB drivers.

Changed lines 121-122 from:

**Why doesn't my sketch start when I'm powering the board with an external power supply?**

to:

### Why doesn't my sketch start when I'm powering the board with an external power supply?

Changed lines 126-127 from:

**Why doesn't my sketch start when I power up or reset the Arduino board?**

to:

### Why doesn't my sketch start when I power up or reset the Arduino board?

Changed lines 130-131 from:

**Why does my sketch appear to upload successfully but not do anything?**

to:

### Why does my sketch appear to upload successfully but not do anything?

Changed lines 138-141 from:

**How can I reduce the size of my sketch?**

The ATmega8 chip on the Arduino board is cheap, but it has only 8 Kb of program code, which isn't very much (and 1 Kb is used by the bootloader).

to:

### How can I reduce the size of my sketch?

The ATmega168 chip on the Arduino board is cheap, but it has only 16 Kb of program code, which isn't very much (and 2 Kb is used by the bootloader).

Changed lines 148-160 from:

**Why don't I get a PWM (an analog output) when I call analogWrite() on pins other than 9, 10, or 11?**

The microcontroller on the Arduino board (the atmega8) only supports PWM/analogWrite() on certain pins. Calling analogWrite() on any other pins will give high (5 volts) for values greater than 128 and low (0 volts) for values less than 128.

## Troubleshooting Old Versions of the Arduino Software

These questions are for reference only, as older versions of the Arduino software are no longer supported. Please download the latest version from the software page?.

**Arduino 0005 won't start on Mac OS X.**

Check your console log (Applications > Utilities > Console). If you see a message like "unsupported major.minor version 49.0", your Java is too old. Run Software Update to upgrade to the latest version.

to:

## Why don't I get a PWM (an analog output) when I call analogWrite() on pins other than 3, 5, 6, 9, 10, or 11?

The microcontroller on the Arduino board (the ATmega168) only supports PWM/analogWrite() on certain pins. Calling analogWrite() on any other pins will give high (5 volts) for values greater than 128 and low (0 volts) for values less than 128. (Older Arduino boards with an ATmega8 only support PWM output on pins 9, 10, and 11.)

Restore
May 31, 2007, at 09:12 AM by David A. Mellis -
Changed lines 130-131 from:

The sketch may be too big for the board. When uploading your sketch, Arduino 0004 checks if it's too big for the ATmega8, but it bases its calculation on a 1 Kb bootloader. You may have a older bootloader that takes up 2 Kb of the 8 Kb of program space (flash) on the ATmega8 instead of the 1 Kb used by the current bootloader. If yours is bigger, only part of the sketch will be uploaded, but the software won't know, and your board will continually reset, pause, reset.

to:

You have selected the wrong item from the Tools > Microcontroller menu. Make sure the selected microcontroller corresponds to the one on your board (either ATmega8 or ATmega168) - the name will be written on the largest chip on the board.

Alternatively, the sketch may be too big for the board. When uploading your sketch, Arduino 0004 checks if it's too big for the ATmega8, but it bases its calculation on a 1 Kb bootloader. You may have a older bootloader that takes up 2 Kb of the 8 Kb of program space (flash) on the ATmega8 instead of the 1 Kb used by the current bootloader. If yours is bigger, only part of the sketch will be uploaded, but the software won't know, and your board will continually reset, pause, reset.

Restore
May 02, 2007, at 07:14 PM by Paul Badger -
Added lines 155-157:

Guide Home

Restore
March 23, 2007, at 05:54 PM by David A. Mellis - adding ftdi question
Added lines 114-117:

**I'm having trouble with the FTDI USB drivers.**

Try installing the latest drivers from FTDI or contacting their support at support1@ftdichip.com.

Restore
February 03, 2007, at 02:22 AM by David A. Mellis - adding note about sketch not starting with external power supply.
Added lines 114-118:

**Why doesn't my sketch start when I'm powering the board with an external power supply?**

Because the RX pin is unconnected, the bootloader on the board may be seeing garbage data coming in, meaning that it never times out and starts your sketch. Try tying the RX pin to ground with a 10K resistor (or connecting it to the TX pin).

Restore
December 22, 2006, at 10:28 PM by David A. Mellis -
Changed line 23 from:

@@Link (dyld) error:

to:

[@Link (dyld) error:

Changed lines 26-27 from:

/Applications/arduino-0004/librxtxSerial.jnilib undefined reference to _printf$LDBL128 expected to be defined in /usr/lib/libSystem.B.dylib @@

to:

/Applications/arduino-0004/librxtxSerial.jnilib undefined reference to _printf$LDBL128 expected to be defined in /usr/lib/libSystem.B.dylib @]

Restore
December 22, 2006, at 10:28 PM by David A. Mellis -
Deleted lines 142-147:

**The Arduino software won't run on Intel Mac machines.**

Arduino 0003 uses a native library for doing serial communication that was only compiled for PPC. You'll need to build your own version of RXTX. See this forum thread for more details.

Arduino 0004 includes a universal version of the RXTX library and should run without modifications on Intel Mac machines. Please report success or failure to the forum.

Restore
December 22, 2006, at 10:27 PM by David A. Mellis -
Changed lines 133-134 from:

In new releases of the software, we'll try to make more efficient use of the program space and give an error message if your program is too big.

to:

We're always working to reduce the size of the Arduino core to leave more room for your sketches.

Restore
December 22, 2006, at 10:26 PM by David A. Mellis - updating upload troubleshooting suggestions
Changed lines 88-89 from:

- Also, on the serial boards, be sure that digital pins 0 and 1 are not connected to anything while uploading (they can connected and used after the code has been uploaded).

to:

- However, on some computers, you may need to press the reset button on the board after you hit the upload button in the Arduino environment. Try different intervals of time between the two, up to 10 seconds or more.

- Disconnect digital pins 0 and 1 while uploading (they can connected and used after the code has been uploaded).

Deleted lines 99-100:

- Or (especially on Intel Macs), the bootloader on the Arduino board might time out before the compilation of the new sketch finishes and the upload begins. Here's plaidTortoise's fix: "To get it to work I hold down the reset button, click the Upload button and wait a couple of seconds. Then I release the button and a second or so later both the TX and RX Leds on the board light up and both start flashing. The upload then occurs."

Restore
December 22, 2006, at 10:21 PM by David A. Mellis -
Changed lines 100-101 from:

- You may need change the baud rate at which sketches are uploaded. It should be 9600 for older boards (those with no version numbers) and 19200 for newer boards (MOST OF -but not all- version 2.0 or greater, and most of the Arduino Extreme, which has two red LEDs marked "RX" and "TX"). You will have to change the speed in the *preferences* file directly. See the preferences page for instructions on finding the file. Look for the file in your computer and change the **serial.download_rate** property to match the one in your board. Though this is a trick to fix this issue, it is recommended to upgrade the bootloader, the one that downloads always at 19200. This can be done with the *'Tools | Burn Bootloader* menu item.

to:

- If you have a really ancient Arduino board, you may need to change the baud rate at which sketches are uploaded to 9600 (from the normal 19200). You will have to change the speed in the *preferences* file directly. See the preferences page for instructions on finding the file. Look for the file in your computer and change the **serial.download_rate** property to match the one in your board. If you have such a board, it's recommended that you burn the latest bootloader (which works at 19200 baud). This can be done with the *'Tools | Burn Bootloader* menu item.

Restore
December 02, 2006, at 04:26 AM by David A. Mellis -
Changed line 114 from:

to:

Restore
December 02, 2006, at 04:25 AM by David A. Mellis -
Deleted lines 12-17:

**The Arduino software won't run on Intel Mac machines.**

Arduino 0003 uses a native library for doing serial communication that was only compiled for PPC. You'll need to build your own version of RXTX. See this forum thread for more details.

Arduino 0004 includes a universal version of the RXTX library and should run without modifications on Intel Mac machines. Please report success or failure to the forum.

Deleted lines 18-21:

**Arduino 0005 won't start on Mac OS X.**

Check your console log (Applications > Utilities > Console). If you see a message like "unsupported major.minor version 49.0", your Java is too old. Run Software Update to upgrade to the latest version.

Added lines 138-151:

## Troubleshooting Old Versions of the Arduino Software

These questions are for reference only, as older versions of the Arduino software are no longer supported. Please download the latest version from the software page?.

**The Arduino software won't run on Intel Mac machines.**

Arduino 0003 uses a native library for doing serial communication that was only compiled for PPC. You'll need to build your own version of RXTX. See this forum thread for more details.

Arduino 0004 includes a universal version of the RXTX library and should run without modifications on Intel Mac machines. Please report success or failure to the forum.

**Arduino 0005 won't start on Mac OS X.**

Check your console log (Applications > Utilities > Console). If you see a message like "unsupported major.minor version 49.0", your Java is too old. Run Software Update to upgrade to the latest version.

Restore
December 02, 2006, at 04:22 AM by David A. Mellis -
Added lines 124-128:

**Why doesn't my sketch start when I power up or reset the Arduino board?**

Most likely because you are sending serial data to the board when it firsts turns on. During the first few seconds, the bootloader (a program pre-burned onto the chip on the board) listens for the computer to send it a new sketch to be uploaded to the board. After a few seconds without communication, the bootloader will time out and start the sketch that's already on the board. If you continue to send data to the bootloader, it will never time out and your sketch will never start. You'll either need to find a way to stop serial data from arriving for the first few seconds when the board powers (e.g. by enabling the chip that sends the data from within your setup() function) or burn your sketch onto the board with an external programmer, replacing the bootloader.

Restore
November 16, 2006, at 05:09 PM by David A. Mellis -
Changed lines 94-95 from:

- Make sure there's a bootloader burned on the Atmega8 on your Arduino board. To check, connect an LED to pin 13 and reset the board. The LED should blink. If it doesn't, see the Bootloader? page for instructions on burning a bootloader to the board.

to:

- Make sure there's a bootloader burned on the Atmega8 on your Arduino board. To check, connect an LED to pin 13 and reset the board. The LED should blink. If it doesn't, see the Bootloader page for instructions on burning a bootloader to the board.

Changed lines 110-111 from:

- You may need change the baud rate at which sketches are uploaded. It should be 9600 for older boards (those with no version numbers) and 19200 for newer boards (MOST OF -but not all- version 2.0 or greater, and most of the Arduino Extreme, which has two red LEDs marked "RX" and "TX"). You will have to change the speed in the *preferences* file directly. See the preferences? page for instructions on finding the file. Look for the file in your computer and change the **serial.download_rate** property to match the one in your board. Though this is a trick to fix this issue, it is recommended to upgrade the bootloader, the one that downloads always at 19200. This can be done with the *'Tools | Burn Bootloader* menu item.

to:

- You may need change the baud rate at which sketches are uploaded. It should be 9600 for older boards (those with no version numbers) and 19200 for newer boards (MOST OF -but not all- version 2.0 or greater, and most of the Arduino Extreme, which has two red LEDs marked "RX" and "TX"). You will have to change the speed in the *preferences* file directly. See the preferences page for instructions on finding the file. Look for the file in your computer and change the **serial.download_rate** property to match the one in your board. Though this is a trick to fix this issue, it is recommended to upgrade the bootloader, the one that downloads always at 19200. This can be done with the *'Tools | Burn Bootloader* menu item.

Changed lines 128-129 from:

If you have access to an AVR-ISP or parallel port programmer, you can burn the latest version of the bootloader to your board with the **Tools | Burn Bootloader** menu item. Otherwise, you can tell the Arduino environment the amount of space available for sketches by editing the upload.maximum_size variable in your preferences file (see: instructions on finding the file?). Change 7168 to 6144, and the environment should correctly warn you when your sketch is too big.

to:

If you have access to an AVR-ISP or parallel port programmer, you can burn the latest version of the bootloader to your board with the **Tools | Burn Bootloader** menu item. Otherwise, you can tell the Arduino environment the amount of space available for sketches by editing the upload.maximum_size variable in your preferences file (see: instructions on finding the file). Change 7168 to 6144, and the environment should correctly warn you when your sketch is too big.

Restore
November 10, 2006, at 01:59 AM by David A. Mellis -
Changed line 58 from:

@@ 6 [main] ? (3512) C:\Dev\arduino-0006\tools\avr\bin\avr-gcc.exe: *** fatal error - C:\Dev\arduino-0006\tools\avr\bin\avr-gcc.exe: *** system shared memory version mismatch detected - 0x75BE0084/0x75BE009C.

to:

```
6 [main] ? (3512) C:\Dev\arduino-0006\tools\avr\bin\avr-gcc.exe: *** fatal error - C:\Dev\arduino-
0006\tools\avr\bin\avr-gcc.exe: *** system shared memory version mismatch detected - 0x75BE0084/0x75BE009C.
```

Changed line 60 from:

This problem is probably due to using incompatible versions of the cygwin DLL.

to:

```
This problem is probably due to using incompatible versions of the cygwin DLL.
```

Changed lines 62-63 from:

Search for cygwin1.dll using the Windows Start->Find/Search facility and delete all but the most recent version. The most recent version *should* reside in x:\cygwin\bin, where 'x' is the drive on which you have installed the cygwin distribution. Rebooting is also suggested if you are unable to find another cygwin DLL.@@

to:

```
Search for cygwin1.dll using the Windows Start->Find/Search facility and delete all but the most recent
version. The most recent version *should* reside in x:\cygwin\bin, where 'x' is the drive on which you
have installed the cygwin distribution. Rebooting is also suggested if you are unable to find another
cygwin DLL.
```

Restore
November 10, 2006, at 01:58 AM by David A. Mellis -
Added lines 54-67:

**What can I do about cygwin conflicts on Windows?**

If you already have cygwin installed on your machine, you might get an error like this when you try to compile a sketch in Arduino:

@@ 6 [main] ? (3512) C:\Dev\arduino-0006\tools\avr\bin\avr-gcc.exe: *** fatal error - C:\Dev\arduino-0006\tools\avr\bin\avr-gcc.exe: *** system shared memory version mismatch detected - 0x75BE0084/0x75BE009C.

This problem is probably due to using incompatible versions of the cygwin DLL.

Search for cygwin1.dll using the Windows Start->Find/Search facility and delete all but the most recent version. The most recent version *should* reside in x:\cygwin\bin, where 'x' is the drive on which you have installed the cygwin distribution. Rebooting is also suggested if you are unable to find another cygwin DLL.@@

If so, first make sure that you don't have cygwin running when you use Arduino. If that doesn't help, you can try deleting cygwin1.dll from the Arduino directory and replacing it with the cygwin1.dll from your existing cygwin install (probably in c:\cygwin\bin).

*Thanks to karlcswanson for the suggestion.*

Restore
October 26, 2006, at 03:22 PM by David A. Mellis - creating troubleshooting page from faq questions.
Added lines 1-128:

# Arduino Troubleshooting

**What if my board doesn't turn on (the green power LED doesn't light up)?**

If you're using a USB board, make sure that the jumper (little plastic piece near the USB plug) is on the correct pins. If you're powering the board with an external power supply (plugged into the power plug), the jumper should be on the two pins closest to the power plug. If you're powering the board through the USB, the jumper should be on the two pins closest to the USB plug. This picture shows the arrangment for powering the board from the USB port.

Attach:jumper.jpg Δ

(thanks to mrbbp for report and picture)

**The Arduino software won't run on Intel Mac machines.**

Arduino 0003 uses a native library for doing serial communication that was only compiled for PPC. You'll need to build your own version of RXTX. See this forum thread for more details.

Arduino 0004 includes a universal version of the RXTX library and should run without modifications on Intel Mac machines. Please report success or failure to the forum.

**What do to if I get the following error when launching arduino.exe on Windows?**

```
Arduino has encountered a problem and needs to close.
```

You'll need to launch Arduino using the run.bat file. Please be patient, the Arduino IDE may take some time to open.

**Arduino 0005 won't start on Mac OS X.**

Check your console log (Applications > Utilities > Console). If you see a message like "unsupported major.minor version 49.0", your Java is too old. Run Software Update to upgrade to the latest version.

**Why won't Arduino run on old versions of Mac OS X?**

If you get an error like this:

@@Link (dyld) error:

dyld: /Applications/arduino-0004/Arduino 04.app/Contents/MacOS/Arduino Undefined symbols: /Applications/arduino-0004/librxtxSerial.jnilib undefined reference to _printf$LDBL128 expected to be defined in /usr/lib/libSystem.B.dylib @@

you probably need to upgrade to Max OS X 10.3.9 or later. Older versions have incompatible versions of some system libraries.

Thanks to Gabe462 for the report.

**What do I do if I get the following error when launching Arduino?**

```
Uncaught exception in main method: java.lang.UnsatisfiedLinkError: Native Library
/Users/anu/Desktop/arduino-0002/librxtxSerial.jnilib already loaded in another classloader
```

You probably have an old version of the communications library lying around. Search for comm.jar or jcl.jar in /System/Library/Frameworks/JavaVM.framework/ or in directories in your CLASSPATH or PATH environment variables. (reported by Anurag Sehgal)

**What about this error?**

```
Java Virtual Machine Launcher: Could not find the main class. Program will exit.
```

Make sure that you correctly extracted the contents of the Arduino .zip file - in particular that the **lib** directory is directly inside of the Arduino directory and contains the file **pde.jar**.

**Why doesn't my board show in the Tools | Serial Port menu ?**

If you're using a USB Arduino board, make sure you installed the FTDI drivers (see the Howto for directions). If you're using a USB-to-Serial adapter with a serial board, make sure you installed its drivers.

Make sure that the board is plugged in: the serial port menu refreshes whenever you open the **Tools** menu, so if you just unplugged the board, it won't be in the menu.

Check that you're not running any programs that scan all serial ports, like PDA sync applications, Bluetooth-USB drivers (e.g. BlueSoleil), virtual daemon tools, etc.

On Windows, the COM port assigned to the board may be too high. From zeveland:

"One little note if you aren't able to export and your USB board is trying to use a high COM port number: try changing the FTDI chip's COM port assignment to a lower one.

"I had a bunch of virtual COM ports set up for Bluetooth so the board was set to use COM17. The IDE wasn't able to find the board so I deleted the other virtual ports in Control Panel (on XP) and moved the FTDI's assignment down to COM2. Make sure to set Arduino to use the new port and good luck."

On the Mac, if you have an old version of the FTDI drivers, you may need to remove them and reinstall the latest version. See this forum thread for directions (thanks to gck).

**Why I can't upload my programs to the Arduino board?**

There are a few things that could be wrong.

- First make sure your board is on (the green LED is on) and connected to the computer (if it's not, see "what if my board doesn't turn on" above).
- Then, check that the proper port is selected in the "Tools | Serial Port" menu (if your port doesn't appear, restart the IDE with the board connected to the computer).
- Make sure there's a bootloader burned on the Atmega8 on your Arduino board. To check, connect an LED to pin 13 and reset the board. The LED should blink. If it doesn't, see the Bootloader? page for instructions on burning a bootloader to the board.
- Be sure that you are resetting the board a couple of seconds before uploading.
- Also, on the serial boards, be sure that digital pins 0 and 1 are not connected to anything while uploading (they can connected and used after the code has been uploaded).
- If you get this error: `[VP 1] Device is not responding correctly.` try uploading again (i.e. reset the board and press the download button a second time).
- Check that you're not running any programs that scan all serial ports, like PDA sync applications, Bluetooth-USB drivers (e.g. BlueSoleil), virtual daemon tools, etc.
- Make sure you don't have firewall software that blocks access to the serial port (e.g. ZoneAlarm).
- You may need to quit Processing, PD, vvvv, etc. if you're using them to read data over the USB or serial connection to the Arduino board.
- Or (especially on Intel Macs), the bootloader on the Arduino board might time out before the compilation of the new sketch finishes and the upload begins. Here's plaidTortoise's fix: "To get it to work I hold down the reset button, click

the Upload button and wait a couple of seconds. Then I release the button and a second or so later both the TX and RX Leds on the board light up and both start flashing. The upload then occurs."

- You may need change the baud rate at which sketches are uploaded. It should be 9600 for older boards (those with no version numbers) and 19200 for newer boards (MOST OF -but not all- version 2.0 or greater, and most of the Arduino Extreme, which has two red LEDs marked "RX" and "TX"). You will have to change the speed in the *preferences* file directly. See the preferences? page for instructions on finding the file. Look for the file in your computer and change the **serial.download_rate** property to match the one in your board. Though this is a trick to fix this issue, it is recommended to upgrade the bootloader, the one that downloads always at 19200. This can be done with the *'Tools | Burn Bootloader* menu item.

**What if I get this error when uploading code or using the serial monitor (on the Mac)?**

```
Error inside Serial.<init>()
gnu.io.PortInUseException: Unknown Application
     at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:354)
     at processing.app.Serial.<init>(Serial.java:127)
     at processing.app.Serial.<init>(Serial.java:72)
```

You need to run the macosx_setup.command in the Arduino directory, and then restart your computer. Arduino 0004 includes a modified version of this script that all users need to run (even those who ran the one that came with Arduino 0003). You may also need to delete the contents of the **/var/spool/uucp** directory.

**Why does my sketch appear to upload successfully but not do anything?**

The sketch may be too big for the board. When uploading your sketch, Arduino 0004 checks if it's too big for the ATmega8, but it bases its calculation on a 1 Kb bootloader. You may have a older bootloader that takes up 2 Kb of the 8 Kb of program space (flash) on the ATmega8 instead of the 1 Kb used by the current bootloader. If yours is bigger, only part of the sketch will be uploaded, but the software won't know, and your board will continually reset, pause, reset.

If you have access to an AVR-ISP or parallel port programmer, you can burn the latest version of the bootloader to your board with the **Tools | Burn Bootloader** menu item. Otherwise, you can tell the Arduino environment the amount of space available for sketches by editing the upload.maximum_size variable in your preferences file (see: instructions on finding the file?). Change 7168 to 6144, and the environment should correctly warn you when your sketch is too big.

**How can I reduce the size of my sketch?**

The ATmega8 chip on the Arduino board is cheap, but it has only 8 Kb of program code, which isn't very much (and 1 Kb is used by the bootloader).

If you're using floating point, try to rewrite your code with integer math, which should save you about 2 Kb. Delete any **#include** statements at the top of your sketch for libraries that you're not using.

Otherwise, see if you can make your program shorter.

In new releases of the software, we'll try to make more efficient use of the program space and give an error message if your program is too big.

**Why don't I get a PWM (an analog output) when I call analogWrite() on pins other than 9, 10, or 11?**

The microcontroller on the Arduino board (the atmega8) only supports PWM/analogWrite() on certain pins. Calling analogWrite() on any other pins will give high (5 volts) for values greater than 128 and low (0 volts) for values less than 128.

Restore

# Arduino Troubleshooting

**On this page...** (hide)

## Why doesn't my sketch start when I'm powering the board with an external power supply?

Because the RX pin is unconnected, the bootloader on the board may be seeing garbage data coming in, meaning that it never times out and starts your sketch. Try tying the RX pin to ground with a 10K resistor (or connecting it to the TX pin).

## Why I can't upload my programs to the Arduino board?

There are a few things that could be wrong.

- First make sure your board is on (the green LED is on) and connected to the computer (if it's not, see "what if my board doesn't turn on" above).
- Then, check that the proper port is selected in the "Tools | Serial Port" menu (if your port doesn't appear, restart the IDE with the board connected to the computer).
- Make sure there's a bootloader burned on the Atmega8 on your Arduino board. To check, connect an LED to pin 13 and reset the board. The LED should blink. If it doesn't, see the Bootloader page for instructions on burning a bootloader to the board.
- Be sure that you are resetting the board a couple of seconds before uploading (unless you have an Arduino Diecimila).
- However, note that some Diecimila were accidently burned with the wrong bootloader and may require you to physically press the reset button before uploading; see this question below.
- However, on some computers, you may need to press the reset button on the board after you hit the upload button in the Arduino environment. Try different intervals of time between the two, up to 10 seconds or more.
- Disconnect digital pins 0 and 1 while uploading (they can connected and used after the code has been uploaded).
- Try uploading with nothing connected to the board (apart from the USB cable, of course).
- Make sure the board isn't touching anything metallic or conductive.
- If you get this error: `[VP 1] Device is not responding correctly.` try uploading again (i.e. reset the board and press the download button a second time).
- Check that you're not running any programs that scan all serial ports, like PDA sync applications, Bluetooth-USB drivers (e.g. BlueSoleil), virtual daemon tools, etc.
- Make sure you don't have firewall software that blocks access to the serial port (e.g. ZoneAlarm).
- You may need to quit Processing, PD, vvvv, etc. if you're using them to read data over the USB or serial connection to the Arduino board.
- If you have a really ancient Arduino board, you may need to change the baud rate at which sketches are uploaded to 9600 (from the normal 19200). You will have to change the speed in the *preferences* file directly. See the preferences page for instructions on finding the file. Look for the file in your computer and change the **serial.download_rate** property to match the one in your board. If you have such a board, it's recommended that you burn the latest bootloader (which works at 19200 baud). This can be done with the *'Tools | Burn Bootloader* menu item.

If it still doesn't work, you can ask for help in the forum. Please include the following information:

- Your operating system.
- What kind of board you have. If it's a Mini, LilyPad or other board that requires extra wiring, include a photo of your circuit, if possible.
- Whether or not you were ever able to upload to the board. If so, what were you doing with the board before / when it stopped working, and what software have you recently added or removed from your computer?
- The messages displayed when you try to upload with verbose output enabled. To do this, you'll need to set upload.verbose to true in your Arduino preferences file.

## Why does the Arduino software freeze when I try to upload a program? (on Windows)?

This might be caused by a conflict with the Logitech process 'LVPrcSrv.exe'. Open the Task Manager and see if this program is running, and if so, kill it before attempting the upload. more information

## What if my board doesn't turn on (the green power LED doesn't light up)?

If you're using a USB board, make sure that the jumper (little plastic piece near the USB plug) is on the correct pins. If you're powering the board with an external power supply (plugged into the power plug), the jumper should be on the two pins closest to the power plug. If you're powering the board through the USB, the jumper should be on the two pins closest to the USB plug. This picture shows the arrangment for powering the board from the USB port.

Attach:jumper.jpg Δ

(thanks to mrbbp for report and picture)

## Why does my Diecimila take such a long time (6-8 seconds) to start my sketch?

Some of the Arduino Diecimila boards were accidently burned with the Arduino NG bootloader. It should work fine, but has a longer delay when the board is reset (because the NG doesn't have an automatic reset, so you have to time the uploads manually). You can recognize the NG bootloader because the LED on pin 13 will blink three times when you reset the board (as compared to once with the Diecimila bootloader). If your Diecimila has the NG bootloader on it, you may need to physically press the reset button on the board before uploading your sketch. You can burn the correct bootloader onto your Diecimila, see the bootloader page for details.

## What should I do if I get an error when launching arduino.exe on Windows?

If you get an error when double-clicking the arduino.exe executable on Windows, for example:

```
Arduino has encountered a problem and needs to close.
```

you'll need to launch Arduino using the run.bat file. Please be patient, the Arduino environment may take some time to open.

## Why won't Arduino run on old versions of Mac OS X?

If you get an error like this:

```
Link (dyld) error:

dyld: /Applications/arduino-0004/Arduino 04.app/Contents/MacOS/Arduino Undefined symbols:
/Applications/arduino-0004/librxtxSerial.jnilib undefined reference to _printf$LDBL128 expected to be defined in /usr/lib/libSystem.B.dylib
```

you probably need to upgrade to Max OS X 10.3.9 or later. Older versions have incompatible versions of some system libraries.

Thanks to Gabe462 for the report.

## What do I do if I get an UnsatisfiedLinkError error (about native library librxtxSerial.jnilib) when launching Arduino?

If you get an error like this when launching Arduino:

```
Uncaught exception in main method: java.lang.UnsatisfiedLinkError: Native Library /Users/anu/Desktop/arduino-0002/librxtxSerial.jnilib already loaded in another classloader
```

you probably have an old version of the communications library lying around. Search for comm.jar or jcl.jar in /System/Library/Frameworks/JavaVM.framework/ or in directories in your CLASSPATH or PATH environment variables. (reported by Anurag Sehgal)

## What about the error "Could not find the main class."?

If you get this error when launching Arduino:

```
Java Virtual Machine Launcher: Could not find the main class. Program will exit.
```

make sure that you correctly extracted the contents of the Arduino .zip file - in particular that the **lib** directory is directly inside of the Arduino directory and contains the file **pde.jar**.

## What can I do about cygwin conflicts on Windows?

If you already have cygwin installed on your machine, you might get an error like this when you try to compile a sketch in Arduino:

```
6 [main] ? (3512) C:\Dev\arduino-0006\tools\avr\bin\avr-gcc.exe: *** fatal error - C:\Dev\arduino-0006\tools\avr\bin\avr-gcc.exe: *** system shared memory version mismatch detected - 0x75BE0084/0x75BE009C.
```

This problem is probably due to using incompatible versions of the cygwin DLL.

Search for cygwin1.dll using the Windows Start->Find/Search facility and delete all but the most recent version. The most recent version *should* reside in x:\cygwin\bin, where 'x' is the drive on which you have installed the cygwin distribution. Rebooting is also suggested if you are unable to find another cygwin DLL.

If so, first make sure that you don't have cygwin running when you use Arduino. If that doesn't help, you can try deleting cygwin1.dll from the Arduino directory and replacing it with the cygwin1.dll from your existing cygwin install (probably in c:\cygwin\bin).

*Thanks to karlcswanson for the suggestion.*

## Why does the Arduino software run really slowly (on Windows)?

If the Arduino software takes a long time to start up and appears to freeze when you try to open the Tools menu, there by a conflict with another device on your system. The Arduino software, on startup and when you open the Tools menu, tries to get a list of all the COM ports on your computer. It's possible that a COM port created by one of the devices on your computer slows down this process. Take a look in the Device Manager. Try disabling the devices that provide COM ports (e.g. Bluetooth devices).

## Why doesn't my board show in the Tools | Serial Port menu ?

If you're using a USB Arduino board, make sure you installed the FTDI drivers (see the Howto for directions). If you're using a USB-to-Serial adapter with a serial board, make sure you installed its drivers.

Make sure that the board is plugged in: the serial port menu refreshes whenever you open the **Tools** menu, so if you just unplugged the board, it won't be in the menu.

Check that you're not running any programs that scan all serial ports, like PDA sync applications, Bluetooth-USB drivers (e.g. BlueSoleil), virtual daemon tools, etc.

On Windows, the COM port assigned to the board may be too high. From zeveland:

"One little note if you aren't able to export and your USB board is trying to use a high COM port number: try changing the FTDI chip's COM port assignment to a lower one.

"I had a bunch of virtual COM ports set up for Bluetooth so the board was set to use COM17. The IDE wasn't able to find the board so I deleted the other virtual ports in Control Panel (on XP) and moved the FTDI's assignment down to COM2. Make sure to set Arduino to use the new port and good luck."

On the Mac, if you have an old version of the FTDI drivers, you may need to remove them and reinstall the latest version. See this forum thread for directions (thanks to gck).

## What if I get a gnu.io.PortInUseException when uploading code or using the serial monitor (on the Mac)?

```
Error inside Serial.<init>()
gnu.io.PortInUseException: Unknown Application
     at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:354)
     at processing.app.Serial.<init>(Serial.java:127)
     at processing.app.Serial.<init>(Serial.java:72)
```

This probably means that the port is actually in use by another application. Please make sure that you're not running other programs that access serial or USB ports, like PDA sync application, bluetooth device managers, certain firewalls, etc. Also, note that some programs (e.g. Max/MSP) keep the serial port open even when not using it - you may to need to close any patches that use the serial port or quit the application entirely.

If you get this error with Arduino 0004 or earlier, or with Processing, you'll need to run the macosx_setup.command, and then restart your computer. Arduino 0004 includes a modified version of this script that all users need to run (even those who ran the one that came with Arduino 0003). You may also need to delete the contents of the **/var/spool/uucp** directory.

## I'm having trouble with the FTDI USB drivers.

Try installing the latest drivers from FTDI or contacting their support at support1@ftdichip.com.

## Why doesn't my sketch start when I power up or reset the Arduino board?

Most likely because you are sending serial data to the board when it firsts turns on. During the first few seconds, the bootloader (a program pre-burned onto the chip on the board) listens for the computer to send it a new sketch to be uploaded to the board. After a few seconds without communication, the bootloader will time out and start the sketch that's already on the board. If you continue to send data to the bootloader, it will never time out and your sketch will never start. You'll either need to find a way to stop serial data from arriving for the first few seconds when the board powers (e.g. by enabling the chip that sends the data from within your setup() function) or burn your sketch onto the board with an external programmer, replacing the bootloader.

## Why does my sketch appear to upload successfully but not do anything?

You have selected the wrong item from the Tools > Microcontroller menu. Make sure the selected microcontroller corresponds to the one on your board (either ATmega8 or ATmega168) - the name will be written on the largest chip on the board.

Check for a noisy power supply. It's possible this could cause the chip to lose its sketch.

Alternatively, the sketch may be too big for the board. When uploading your sketch, Arduino 0004 checks if it's too big for the ATmega8, but it bases its calculation on a 1 Kb bootloader. You may have a older bootloader that takes up 2 Kb of the 8 Kb of program space (flash) on the ATmega8 instead of the 1 Kb used by the current bootloader. If yours is bigger, only part of the sketch will be uploaded, but the software won't know, and your board will continually reset, pause, reset.

If you have access to an AVR-ISP or parallel port programmer, you can burn the latest version of the bootloader to your board with the **Tools | Burn Bootloader** menu item. Otherwise, you can tell the Arduino environment the amount of space available for sketches by editing the upload.maximum_size variable in your preferences file (see: instructions on finding the file). Change 7168 to 6144, and the environment should correctly warn you when your sketch is too big.

## How can I reduce the size of my sketch?

The ATmega168 chip on the Arduino board is cheap, but it has only 16 Kb of program code, which isn't very much (and 2 Kb is used by the bootloader).

If you're using floating point, try to rewrite your code with integer math, which should save you about 2 Kb. Delete any **#include** statements at the top of your sketch for libraries that you're not using.

Otherwise, see if you can make your program shorter.

We're always working to reduce the size of the Arduino core to leave more room for your sketches.

## Why don't I get a PWM (an analog output) when I call analogWrite() on pins other than 3, 5, 6, 9, 10, or 11?

The microcontroller on the Arduino board (the ATmega168) only supports PWM/analogWrite() on certain pins. Calling analogWrite() on any other pins will give high (5 volts) for values greater than 128 and low (0 volts) for values less than 128. (Older Arduino boards with an ATmega8 only support PWM output on pins 9, 10, and 11.)

## Why do I get errors about undeclared functions or undeclared types?

The Arduino environment attempts to automatically generate prototypes for your functions, so that you can order them as you like in your sketch. This process, however, isn't perfect, and sometimes leads to obscure error messages.

If you declare a custom type in your code and create a function that accepts or returns a value of that type, you'll get an error when you try to compile the sketch. This is because the automatically-generated prototype for that function will appear above the type definition.

If you declare a function with a two-word return type (e.g. "unsigned int") the environment will not realize it's a function and will not create a prototype for it. That means you need to provide your own, or place the definition of the function above any calls to it.

Guide Home

Arduino

search

## Login to Arduino

|  |  |
|---|---|
| Username: |  |
| Password: |  |
| Keep me logged in: | ✓ |

Login

## Guide.Environment History

Hide minor edits - Show changes to markup

April 23, 2008, at 10:28 PM by David A. Mellis -
Changed line 3 from:

(:table width=90% border=0 cellpadding=5 cellspacing=0:)

to:

(:table width=100% border=0 cellpadding=5 cellspacing=0:)

Restore
January 30, 2008, at 09:41 AM by David A. Mellis -
Added lines 52-53:

**Tab Menu**

Changed lines 78-80 from:

Adds another source file to the sketch. The new file appears in a new tab in the sketch window. This facilitates cut and paste between sketches, and larger projects with multiple source files. Deleting extra files in a sketch must be done manually be opening the sketch folder and deleting the unwanted file.

to:

Adds another source file to the sketch. The new file appears in a new tab in the sketch window. This facilitates and larger projects with multiple source files. Files can be removed from a sketch using the tab menu.

Restore
January 30, 2008, at 07:12 AM by Paul Badger -
Changed lines 76-78 from:

Adds another source file to the sketch. This facilitates cut and paste between sketches, and larger projects with multiple source files. Deleting extra files in a sketch must be done manually be opening the sketch folder and deleting the unwanted file.

to:

Adds another source file to the sketch. The new file appears in a new tab in the sketch window. This facilitates cut and paste between sketches, and larger projects with multiple source files. Deleting extra files in a sketch must be done manually be opening the sketch folder and deleting the unwanted file.

Restore
January 30, 2008, at 07:09 AM by Paul Badger -
Changed lines 70-71 from:

''Show Sketch Folder"

to:

*Show Sketch Folder*

Changed lines 74-75 from:

''Add File..."

to:

*Add File...*

Restore
January 30, 2008, at 07:08 AM by Paul Badger -
Added lines 70-78:

''Show Sketch Folder"

Opens the sketch folder on the desktop.

''Add File..."

Adds another source file to the sketch. This facilitates cut and paste between sketches, and larger projects with multiple source files. Deleting extra files in a sketch must be done manually be opening the sketch folder and deleting the unwanted file.

Restore
January 04, 2008, at 10:29 PM by David A. Mellis -
Added lines 54-55:

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no extension), C files (.c extension), C++ files (.cpp), or header files (.h). See the description of the build process for details of how these are handled.

Restore
January 04, 2008, at 10:25 PM by David A. Mellis -
Added lines 52-53:
⇨
Restore
January 04, 2008, at 10:13 PM by David A. Mellis -
Added lines 72-75:

*Copy for Discourse*

Copies the code of your sketch to the clipboard in a forum suitable for posting to the forum, complete with syntax coloring.

Restore
November 02, 2007, at 04:24 PM by David A. Mellis -
Changed line 90 from:

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preferences files.

to:

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preference files.

Restore
November 02, 2007, at 04:24 PM by David A. Mellis -
Changed line 90 from:

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preferences.txt file.

to:

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preferences files.

Restore
November 02, 2007, at 04:24 PM by David A. Mellis -
Changed lines 64-65 from:

Uses a library in your sketch. Works by adding **#include**s to the top of your code. This makes extra functionality available to your sketch, but increases its size. To stop using a library, delete the appropriate **#include**s from the top of your sketch. For more details, see the page on libraries.

to:

Uses a library in your sketch. Works by adding **#include**s to the top of your code. This makes extra functionality available to your sketch, but increases its size. To stop using a library, delete the appropriate **#include**s from the top of your sketch. For more details, see the page on Libraries.

Restore
October 22, 2007, at 10:52 AM by David A. Mellis -
Changed lines 72-75 from:

*Microcontroller (MCU)*

This menu lets you choose which microcontroller you're using; it should match the name (up to the dash) of the chip on your Arduino board (e.g. if your chip says "ATMEGA168-20PU", you would choose "atmega168"). New Arduino boards use the ATmega168, but some older ones have ATmega8s.

to:

*Board*

Select the board that you're using. This controls the way that your sketch is compiled and uploaded as well as the behavior of the Burn Bootloader menu items.

Deleted lines 81-88:

*Burn Diecimila Bootloader*

This burns the Arduino Diecimila ATmega168 bootloader to your Arduino board using an AVRISP mkII. Only available if you have ATmega168 selected in the **Microcontroller** submenu. For more details see the bootloader page.

*Burn Mini/NG Bootloader*

This burns the Arduino Mini/NG bootloader to your Arduino board using an AVRISP mkII. Only available if you have ATmega168 selected in the **Microcontroller** submenu. For more details see the bootloader page.

Changed lines 84-89 from:

This burns the ATmega8 bootloader to your Arduino board using an AVRISP and the serial port you've selected in the **Serial Port** submenu. Only available if you have ATmega8 selected in the **Microcontroller** submenu. For more details see the bootloader page.

*Burn Bootloader (parallel)*

*Windows and Linux only*. Burns the bootloader to your Arduino board, using a parallel programmer. This only works with the atmega8 (not the atmega168).

to:

The items in this menu allow you to burn a bootloader onto your board with a variety of programmers. This is not required for normal use of an Arduino board, but may be useful if you purchase additional ATmega's or are building a board yourself. Ensure that you've selected the correct board from the Boards menu beforehand. To burn a bootloader with the AVR ISP, you need to select the item corresponding to your programmer from the Serial Port menu. Instructions are available for building a parallel programmer.

Restore
August 07, 2007, at 10:39 AM by David A. Mellis -
Changed lines 48-49 from:

Displays serial data being sent from the Arduino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to **Serial.begin** in your sketch.

to:

Displays serial data being sent from the Arduino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to **Serial.begin** in your sketch. Note that on Mac or Linux, the Arduino board will reset (rerun your sketch from the beginning) when you connect with the serial monitor.

Restore
August 06, 2007, at 08:56 PM by David A. Mellis - updating burn bootloader documentation for arduino 0009
Added lines 82-89:

*Burn Diecimila Bootloader*

This burns the Arduino Diecimila ATmega168 bootloader to your Arduino board using an AVRISP mkII. Only available if you have ATmega168 selected in the **Microcontroller** submenu. For more details see the bootloader page.

*Burn Mini/NG Bootloader*

This burns the Arduino Mini/NG bootloader to your Arduino board using an AVRISP mkII. Only available if you have ATmega168 selected in the **Microcontroller** submenu. For more details see the bootloader page.

Changed lines 92-93 from:

This burns the bootloader to your Arduino board, using an an AVR-ISP connected to the serial port selected in the **Serial Port** submenu. This only works with the atmega8 (not the atmega168). For more details see the bootloader page.

to:

This burns the ATmega8 bootloader to your Arduino board using an AVRISP and the serial port you've selected in the **Serial Port** submenu. Only available if you have ATmega8 selected in the **Microcontroller** submenu. For more details see the bootloader page.

Restore
June 15, 2007, at 05:36 PM by David A. Mellis - removing references to ancient versions of the software
Changed lines 48-49 from:

Displays serial data being sent from the Arduino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter (in Arduino 0005, pressing enter appends a newline to your text, this was removed in Arduino 0006). Choose the baud rate from the drop-down that matches the rate passed to **Serial.begin** in your sketch (in version of Arduino prior to 0006, the baud rate is specified in the **Tools | Serial Monitor Baud Rate** menu).

to:

Displays serial data being sent from the Arduino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to **Serial.begin** in your sketch.

Deleted lines 81-86:

*Serial Monitor Baud Rate* (Arduino 0005 and earlier)

This menu item controls the baud rate (speed) that the serial monitor uses to communicate with a sketch running on the Arduino board. It must match the value passed to in the code of the sketch. In Arduino 0006, this baud rate is set from a drop-down in the status bar when the serial monitor is enabled.

This baud rate does not affect the process of uploading sketches to the Arduino board; see the FAQ? if you need to change that.

Restore
June 15, 2007, at 05:35 PM by David A. Mellis -
Changed lines 74-75 from:

This menu lets you choose which microcontroller you're using; it should match the name (up to the dash) of the chip on your Arduino board (e.g. if your chip says "ATMEGA8-16PI", you would choose "atmega8"). Almost all Arduino boards use the atmega8, but the new Arduino stamps use the atmega168 (which can hold programs which are twice as big).

to:

This menu lets you choose which microcontroller you're using; it should match the name (up to the dash) of the chip on your Arduino board (e.g. if your chip says "ATMEGA168-20PU", you would choose "atmega168"). New Arduino boards use the ATmega168, but some older ones have ATmega8s.

Restore
November 04, 2006, at 10:36 AM by David A. Mellis -
Changed lines 64-65 from:

Uses a library in your sketch. Works by adding **#include**s to the top of your code. This makes extra functionality available to your sketch, but increases its size. To stop using a library, delete the appropriate **#include**s from the top of your sketch. For more details, see the page on libraries?.

to:

Uses a library in your sketch. Works by adding **#include**s to the top of your code. This makes extra functionality available to your sketch, but increases its size. To stop using a library, delete the appropriate **#include**s from the top of your sketch. For more details, see the page on libraries.

Changed lines 90-91 from:

This burns the bootloader to your Arduino board, using an an AVR-ISP connected to the serial port selected in the **Serial Port** submenu. This only works with the atmega8 (not the atmega168). For more details see the bootloader? page.

to:

This burns the bootloader to your Arduino board, using an an AVR-ISP connected to the serial port selected in the **Serial Port** submenu. This only works with the atmega8 (not the atmega168). For more details see the bootloader page.

Changed lines 94-95 from:

*Windows and Linux only*. Burns the bootloader to your Arduino board, using a parallel programmer?. This only works with the atmega8 (not the atmega168).

to:

*Windows and Linux only*. Burns the bootloader to your Arduino board, using a parallel programmer. This only works with the atmega8 (not the atmega168).

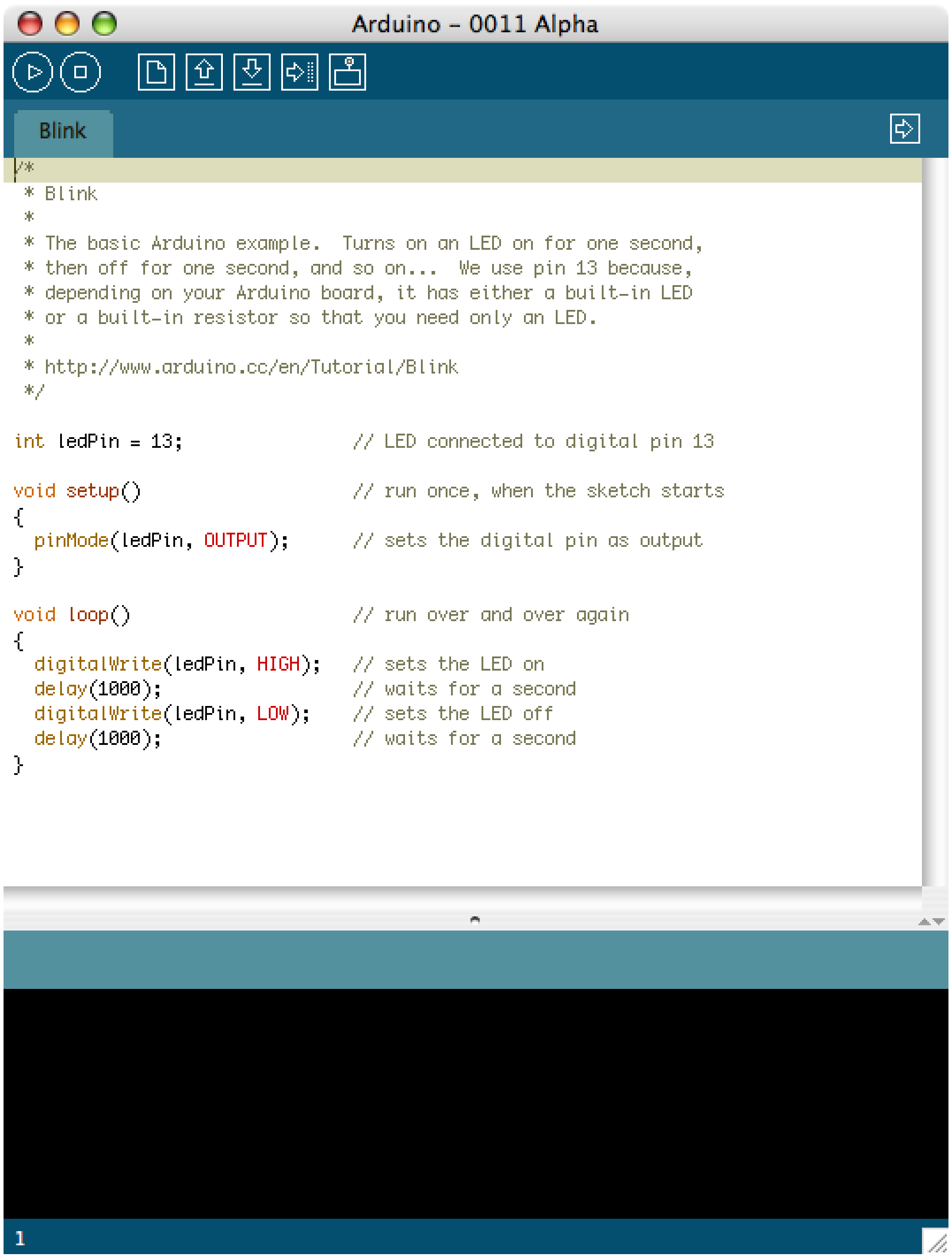
Changed line 100 from:

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preferences.txt file?.

to:

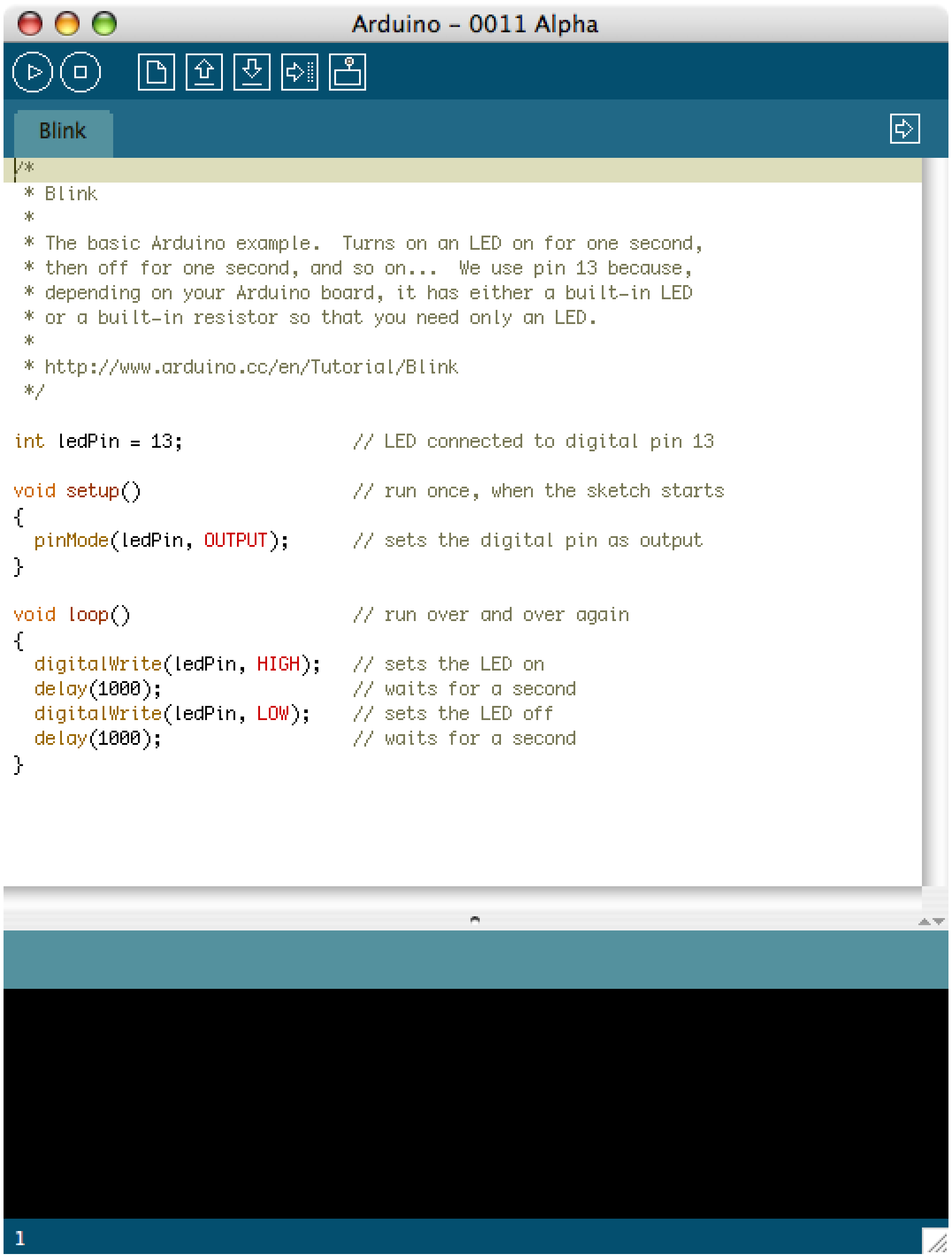
Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preferences.txt file.

Restore
November 04, 2006, at 10:20 AM by David A. Mellis -
Deleted lines 2-3:



```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

Restore
November 04, 2006, at 10:20 AM by David A. Mellis -
Added lines 3-4:



```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

Restore
October 26, 2006, at 03:26 PM by David A. Mellis -
Added lines 1-100:

## Introduction to the Arduino Environment

(:table width=90% border=0 cellpadding=5 cellspacing=0:) (:cell width=50%:)

### Toolbar

*Verify/Compile*

Checks your code for errors.

*Stop*

Stops the serial monitor, or unhighlight other buttons.

*New*

Creates a new sketch.

*Open*

Presents a menu of all the sketches in your sketchbook. Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the **File | Sketchbook** menu instead.

*Save*

Saves your sketch.

*Upload to I/O Board*

Uploads your code to the Arduino I/O board. Make sure to save or verify your sketch before uploading it.

*Serial Monitor*

Displays serial data being sent from the Arduino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter (in Arduino 0005, pressing enter appends a newline to your text, this was removed in Arduino 0006). Choose the baud rate from the drop-down that matches the rate passed to **Serial.begin** in your sketch (in version of Arduino prior to 0006, the baud rate is specified in the **Tools | Serial Monitor Baud Rate** menu).

You can also talk to the board from Processing, Flash, MaxMSP, etc (see the interfacing page for details).

(:cell width=50%:)

## Menus

**Sketch**

*Verify/Compile*

Checks your sketch for errors.

*Import Library*

Uses a library in your sketch. Works by adding **#include**s to the top of your code. This makes extra functionality available to your sketch, but increases its size. To stop using a library, delete the appropriate **#include**s from the top of your sketch. For more details, see the page on libraries?.

**Tools**

*Auto Format*

This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements instead curly braces are indented more.

*Microcontroller (MCU)*

This menu lets you choose which microcontroller you're using; it should match the name (up to the dash) of the chip on your Arduino board (e.g. if your chip says "ATMEGA8-16PI", you would choose "atmega8"). Almost all Arduino boards use the atmega8, but the new Arduino stamps use the atmega168 (which can hold programs which are twice as big).

*Serial Port*

This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

Before uploading your sketch, you need to select the item from this menu that represents your Arduino board. On the Mac, this is probably something like **/dev/tty.usbserial-1B1** (for a USB board), or **/dev/tty.USA19QW1b1P1.1** (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably **COM1** or **COM2** (for a serial board) or **COM4**, **COM5**, **COM7**, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager.

*Serial Monitor Baud Rate* (Arduino 0005 and earlier)

This menu item controls the baud rate (speed) that the serial monitor uses to communicate with a sketch running on the Arduino board. It must match the value passed to in the code of the sketch. In Arduino 0006, this baud rate is set from a drop-down in the status bar when the serial monitor is enabled.

This baud rate does not affect the process of uploading sketches to the Arduino board; see the FAQ? if you need to change that.

*Burn Bootloader*

This burns the bootloader to your Arduino board, using an an AVR-ISP connected to the serial port selected in the **Serial Port** submenu. This only works with the atmega8 (not the atmega168). For more details see the bootloader? page.

*Burn Bootloader (parallel)*

*Windows and Linux only*. Burns the bootloader to your Arduino board, using a parallel programmer?. This only works with the atmega8 (not the atmega168).

(:tableend:)

## Preferences

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preferences.txt file?.

Restore

**Guide** Contents | Introduction | How To: Windows, Mac OS X, Linux; Arduino Nano, Arduino Mini, Arduino BT, LilyPad Arduino; Xbee shield | Troubleshooting | Environment

# Introduction to the Arduino Environment

## Toolbar

*Verify/Compile*

Checks your code for errors.

*Stop*

Stops the serial monitor, or unhighlight other buttons.

*New*

Creates a new sketch.

*Open*

Presents a menu of all the sketches in your sketchbook. Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the **File | Sketchbook** menu instead.

*Save*

Saves your sketch.

*Upload to I/O Board*

Uploads your code to the Arduino I/O board. Make sure to save or verify your sketch before uploading it.

*Serial Monitor*

Displays serial data being sent from the Arduino board

## Menus

### Sketch

*Verify/Compile*

Checks your sketch for errors.

*Import Library*

Uses a library in your sketch. Works by adding **#include**s to the top of your code. This makes extra functionality available to your sketch, but increases its size. To stop using a library, delete the appropriate **#include**s from the top of your sketch. For more details, see the page on Libraries.

*Show Sketch Folder*

Opens the sketch folder on the desktop.

*Add File...*

Adds another source file to the sketch. The new file appears in a new tab in the sketch window. This facilitates and larger projects with multiple source files. Files can be removed from a sketch using the tab menu.

### Tools

*Auto Format*

This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements instead curly braces are indented more.

*Copy for Discourse*

Copies the code of your sketch to the clipboard in a forum suitable for posting to the forum, complete with syntax coloring.

*Board*

Select the board that you're using. This controls the way that your sketch is compiled and uploaded as well as the behavior of the Burn Bootloader menu items.

(USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to **Serial.begin** in your sketch. Note that on Mac or Linux, the Arduino board will reset (rerun your sketch from the beginning) when you connect with the serial monitor.

You can also talk to the board from Processing, Flash, MaxMSP, etc (see the interfacing page for details).

### Tab Menu

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no extension), C files (.c extension), C++ files (.cpp), or header files (.h). See the description of the build process for details of how these are handled.

*Serial Port*

This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

Before uploading your sketch, you need to select the item from this menu that represents your Arduino board. On the Mac, this is probably something like **/dev/tty.usbserial-1B1** (for a USB board), or **/dev/tty.USA19QW1b1P1.1** (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably **COM1** or **COM2** (for a serial board) or **COM4**, **COM5**, **COM7**, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager.

*Burn Bootloader*

The items in this menu allow you to burn a bootloader onto your board with a variety of programmers. This is not required for normal use of an Arduino board, but may be useful if you purchase additional ATmega's or are building a board yourself. Ensure that you've selected the correct board from the Boards menu beforehand. To burn a bootloader with the AVR ISP, you need to select the item corresponding to your programmer from the Serial Port menu. Instructions are available for building a parallel programmer.

## Preferences

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preference files.

## Guide.Guide History

Show minor edits - Show changes to markup

October 22, 2006, at 12:19 PM by David A. Mellis -
Deleted lines 0-30:

## Guide to Arduino

This guide tell you everything you need to get started with Arduino. It consists of the following pages:

- **Introduction**: this page, which explains what it is and why you'd want to use it
- How To: step-by-step instructions on getting your first Arduino program working
- Board: description of the Arduino board and its parts
- Environment: description of the Arduino development environment and its parts
- References: pointers to other sources of information, on this site and elsewhere, for learning more

### What is Arduino?

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

### Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino

- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Restore
October 22, 2006, at 12:18 PM by David A. Mellis -
Changed lines 3-4 from:

This guide tell you everything you need to get started with Arduino. It explains what it is and why you'd want to use it. Then, it offers step-by-step instructions on getting your first Arduino program working, followed by more details on the Arduino hardware and software. Finally, it points to other sources of information, on this site and elsewhere, for learning more.

to:

This guide tell you everything you need to get started with Arduino. It consists of the following pages:

- **Introduction**: this page, which explains what it is and why you'd want to use it
- How To: step-by-step instructions on getting your first Arduino program working
- Board: description of the Arduino board and its parts
- Environment: description of the Arduino development environment and its parts
- References: pointers to other sources of information, on this site and elsewhere, for learning more

Restore
October 22, 2006, at 12:02 PM by David A. Mellis -
Changed lines 7-12 from:

Arduino is an open-source physical computing platform based on a simple i/o board, and a development environment for writing Arduino software. The Arduino programming language is an implementation of Wiring, itself built on Processing.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand? or purchased? preassembled; the open-source IDE can be downloaded? for free.

Arduino received an Honory Mention in the Digital Communities section of the 2006 Ars Electronica Prix. Credits?

to:

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

Restore
October 22, 2006, at 12:01 PM by David A. Mellis - tom's what is arduino and why would you want to use it
Added lines 1-25:

## Guide to Arduino

This guide tell you everything you need to get started with Arduino. It explains what it is and why you'd want to use it. Then, it offers step-by-step instructions on getting your first Arduino program working, followed by more details on the Arduino hardware and software. Finally, it points to other sources of information, on this site and elsewhere, for learning more.

### What is Arduino?

Arduino is an open-source physical computing platform based on a simple i/o board, and a development environment for writing Arduino software. The Arduino programming language is an implementation of Wiring, itself built on Processing.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand? or purchased? preassembled; the open-source IDE can be downloaded? for free.

Arduino received an Honory Mention in the Digital Communities section of the 2006 Ars Electronica Prix. Credits?

## Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino
- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Restore

## Guide.Guide History

Hide minor edits - Show changes to output

October 22, 2006, at 12:19 PM by David A. Mellis -
Deleted lines 0-30:
!!Guide to Arduino

This guide tell you everything you need to get started with Arduino. It consists of the following pages:

* [[Guide/Guide | Introduction]]: this page, which explains what it is and why you'd want to use it
* [[Guide/Howto | How To]]: step-by-step instructions on getting your first Arduino program working
* [[Guide/Board]]: description of the Arduino board and its parts
* [[Guide/Environment]]: description of the Arduino development environment and its parts
* [[Guide/References]]: pointers to other sources of information, on this site and elsewhere, for learning more

!!!What is Arduino?

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

!!!Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

* Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

* Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

* Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino

* Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.

* Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The

plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.
Restore
October 22, 2006, at 12:18 PM by David A. Mellis -
Changed lines 3-4 from:
This guide tell you everything you need to get started with Arduino. It explains what it is and why you'd want to use it. Then, it offers step-by-step instructions on getting your first Arduino program working, followed by more details on the Arduino hardware and software. Finally, it points to other sources of information, on this site and elsewhere, for learning more.
to:
This guide tell you everything you need to get started with Arduino. It consists of the following pages:

* [[Guide/Guide | Introduction]]: this page, which explains what it is and why you'd want to use it
* [[Guide/Howto | How To]]: step-by-step instructions on getting your first Arduino program working
* [[Guide/Board]]: description of the Arduino board and its parts
* [[Guide/Environment]]: description of the Arduino development environment and its parts
* [[Guide/References]]: pointers to other sources of information, on this site and elsewhere, for learning more
Restore
October 22, 2006, at 12:02 PM by David A. Mellis -
Changed lines 7-12 from:
Arduino is an open-source physical computing platform based on a simple i/o board, and a development environment for writing Arduino software. The Arduino programming language is an implementation of [[http://wiring.org.co/ | Wiring]], itself built on [[http://www.processing.org/ | Processing]].

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be [[USBAssembly | assembled by hand]] or [[buy | purchased]] preassembled; the open-source IDE can be [[software | downloaded]] for free.

Arduino received an Honory Mention in the Digital Communities section of the 2006 Ars Electronica Prix. [[Credits]]
to:
Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.
Restore
October 22, 2006, at 12:01 PM by David A. Mellis - tom's what is arduino and why would you want to use it
Added lines 1-25:
!!Guide to Arduino

This guide tell you everything you need to get started with Arduino. It explains what it is and why you'd want to use it. Then, it offers step-by-step instructions on getting your first Arduino program working, followed by more details on the Arduino hardware and software. Finally, it points to other sources of information, on this site and elsewhere, for learning more.

!!!What is Arduino?

Arduino is an open-source physical computing platform based on a simple i/o board, and a development environment for writing Arduino software. The Arduino programming language is an implementation of [[http://wiring.org.co/ | Wiring]], itself built on [[http://www.processing.org/ | Processing]].

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be [[USBAssembly | assembled by hand]] or [[buy | purchased]] preassembled; the open-source IDE can be [[software | downloaded]] for free.

Arduino received an Honory Mention in the Digital Communities section of the 2006 Ars Electronica Prix. [[Credits]]

!!!Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

* Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

* Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

* Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino

* Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you want to.

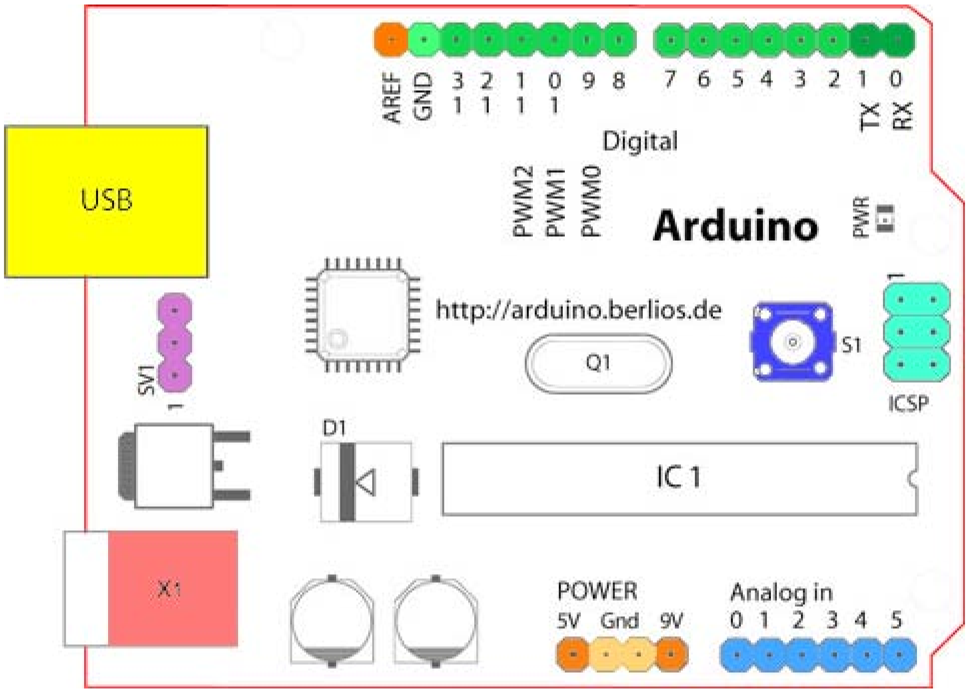
* Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.
Restore

*(redirected from* Guide.Board*)*

**Reference** Language (extended) | Libraries | Comparison | **Board**

## Introduction to the Arduino Board

Looking at the board from the top down, this is an outline of what you will see (parts of the board you might interact with in the course of normal use are highlighted):



Starting clockwise from the top center:

- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - *These pins cannot be used for digital i/o (**digitalRead** and **digitalWrite**) if you are also using serial communication (e.g. **Serial.begin**).*
- Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green)
- Analog In Pins 0-5 (light blue)
- Power and Ground Pins (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC) - X1 (pink)
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

### Microcontrollers

*ATmega168* (used on most Arduino boards)

| | |
|---|---|
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| Analog Input Pins | 6 (DIP) or 8 (SMD) |

*ATmega8* (used on some older board)

| | |
|---|---|
| Digital I/O Pins | 14 (of which 3 provide PWM output) |
| Analog Input Pins | 6 |

| | | | |
|---|---|---|---|
| DC Current per I/O Pin | 40 mA | DC Current per I/O Pin | 40 mA |
| Flash Memory | 16 KB | Flash Memory | 8 KB |
| SRAM | 1 KB | SRAM | 1 KB |
| EEPROM | 512 bytes | EEPROM | 512 bytes |
| (datasheet) | | (datasheet) | |

## Digital Pins

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pinMode(), digitalRead(), and digitalWrite() commands. Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40 mA.

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. On the Arduino Diecimila, these pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip. On the Arduino BT, they are connected to the corresponding pins of the WT11 Bluetooth module. On the Arduino Mini and LilyPad Arduino, they are intended for use with an external TTL serial module (e.g. the Mini-USB Adapter).
- **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. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.
- **BT Reset: 7.** (Arduino BT-only) Connected to the reset line of the bluetooth module.
- **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.** On the Diecimila and LilyPad, 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.

## Analog Pins

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analogRead() function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

- **I$^2$C: 4 (SDA) and 5 (SCL).** Support I$^2$C (TWI) communication using the Wire library (documentation on the Wiring website).

## Power Pins

- **VIN** (sometimes labelled "9V"). The input voltage to the Arduino 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. Note that different boards accept different input voltages ranges, please see the documentation for your board. Also note that the LilyPad has no VIN pin and accepts only a regulated input.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** (Diecimila-only) A 3.3 volt supply generated by the on-board FTDI chip.
- **GND.** Ground pins.

## Other Pins

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

Reference Home

*Corrections, suggestions, and new documentation should be posted to the Forum.*

## Other Resources

### On the Arduino Website

- Tutorials: code examples and circuits for performing many tasks.
- Reference: documentation of the Arduino programming language and functions.
- Hardware: descriptions of the various Arduino boards and other hardware, with schematics, PCB layout files, assembly instructions, etc.
- Language comparison: compares the Arduino/Wiring language (based on C/C++) with Processing (based on Java).
- FAQ: frequently asked questions about Arduino.
- Discussion forums: for questions about anything Arduino related.
- Playground: a publicly editable wiki collecting community documentation (register here).

### External Resources on Arduino

- Video lectures: Tom Igoe introduces Arduino. (thanks to Pollie Barden).
- Course guides: Longer documents introducing Arduino: class 1 (getting started), class 2 (input and sensors), class 3 (communication, servos, and pwm), class 4 (piezo sound & sensors, arduino+processing, stand-alone operation).

### Electronics and Physical Computing

- Wiring electronics reference: circuit diagrams for connecting a variety of basic electronic components.
- Schematics to circuits: from Wiring, a guide to transforming circuit diagrams into physical circuits.
- Open Circuits: a public wiki collecting electronic circuits, components, and techniques.
- Tom Igoe's Physical Computing Site: lots of information on electronics, microcontrollers, sensors, actuators, books, etc.

### Related Sites

- Instant Soup is an introduction to electronics through a series of beautifully-documented fun projects.
- Make magazine has some great links in its electronics archive.
- hack a day has links to interesting hacks and how-to articles on various topics.

## Login to Arduino

| | |
|---|---|
| Username: | |
| Password: | |
| Keep me logged in: | ✓ |

Login

# Arduino

search

## Login to Arduino

| | |
|---|---|
| Username: | |
| Password: | |
| Keep me logged in: | ✓ |

Login

# Arduino

search

## Login to Arduino

|  |  |
|---|---|
| Username: |  |
| Password: |  |
| Keep me logged in: | ✓ |

Login

# Arduino

search

## Login to Arduino

| | |
|---|---|
| Username: | |
| Password: | |
| Keep me logged in: | ✓ |

Login

# Arduino

search

## Login to Arduino

|  |  |
|---|---|
| Username: |  |
| Password: |  |
| Keep me logged in: | ✓ |

Login

# Arduino

search

## Login to Arduino

|  |  |
|---|---|
| Username: |  |
| Password: |  |
| Keep me logged in: | ✓ |

Login

## Login to Arduino

Username:

Password:

Keep me logged in: ✓

Login

# Arduino

search

## Login to Arduino

Username:

Password:

Keep me logged in: ✓

Login