

high-low Yeah

[Programming an ATtiny w/ Arduino 0022](#)

This tutorial shows you how to program an ATtiny45 or ATtiny85 microcontroller using the Arduino software and hardware. The ATtiny45 and ATtiny85 are small (8-leg), cheap (\$2-3) microcontrollers that are convenient for running simple programs.

This tutorial is obsolete! See the [updated version of this tutorial](#) (with support for Arduino 1.0).

They are almost identical, except that the ATtiny85 has twice the memory of the ATtiny45 and can therefore hold more complex programs. We like to use both of them with paper circuits and other craft electronics. To program them, we'll use a port of the Arduino core libraries created by [Alessandro Saporetti](#) and slightly modified by HLT.

Materials and Tools

For this tutorial, you'll need:

- [Arduino Uno or Duemilanove](#) (w/ an ATmega328, not an older board with an ATmega168)
- [ATtiny45](#) or [ATtiny85](#) (8-pin DIP package)
- a 10 uF capacitor (e.g. [from Sparkfun](#) or [from Digi-Key](#))
- a breadboard
- jumper wires

For more information, see our [list of materials and parts](#) and our [list of prototyping supplies](#).

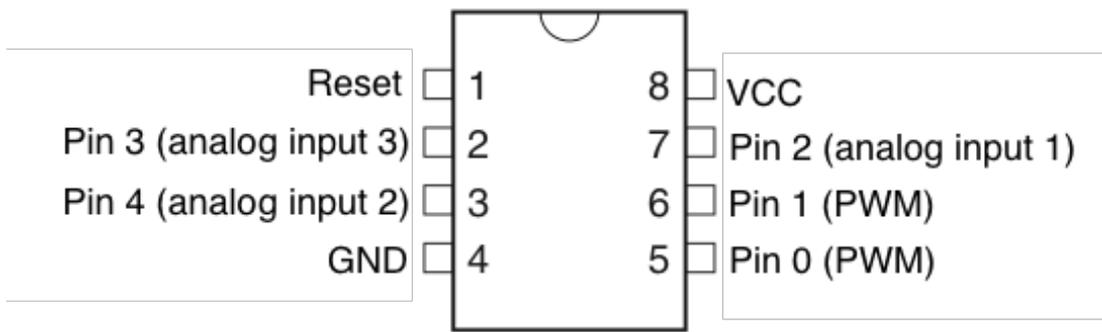
Software

You'll need the Arduino software (version 0022) and a plugin to add support for the ATtiny45/85. Installation instructions for the Arduino software are available [for Windows](#) and [for Mac OS X](#).

Download: [Arduino software, attiny45_85.zip](#)

The ATtiny45 / ATtiny85 Microcontroller

Here's the pinout of the ATtiny45 and ATtiny85:



Installing ATtiny45 / ATtiny85 support in Arduino

- Download: [attiny45_85.zip](#)
- Locate your Arduino sketchbook folder (you can find its location in the preferences dialog in the Arduino software)
- Create a new sub-folder called “hardware” in the sketchbook folder.
- Copy the attiny45_85 folder from the attiny45_85.zip to the hardware folder.
- Restart the Arduino development environment.

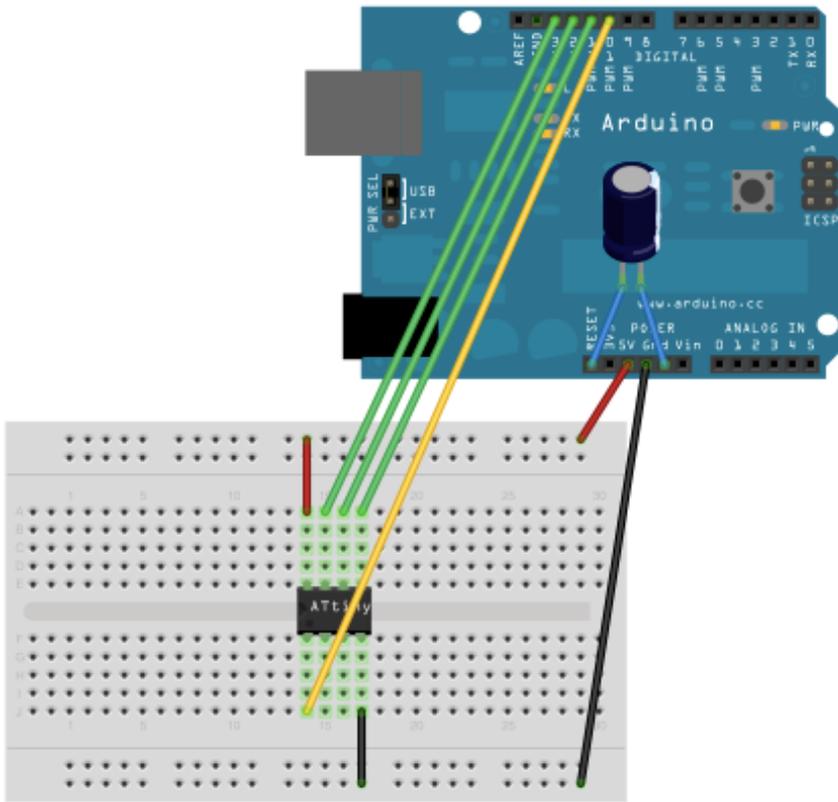
Turning the Arduino board into a programmer

We’re going to use the Arduino board to program the ATtiny45 or ATtiny85. First, we’ll need to turn the Arduino board into an “in-system programmer” (ISP). To this:

- Run the Arduino development environment.
- Open the ArduinoISP sketch from the examples menu.
- Select the board and serial port that correspond to your Arduino board.
- Upload the ArduinoISP sketch.

Connecting the Arduino board and the ATtiny

We’ll connect the Arduino board to the ATtiny as shown in the following diagram. Use the dot in the corner of the ATtiny to orient it properly. We’ll also connect a 10 uF capacitor between reset and ground on the Arduino board as shown in the diagram (the stripe on the capacitor that’s marked with a negative sign (-) should go to ground). The capacitor prevents the Arduino board from resetting (which starts the bootloader), thus ensuring that the Arduino IDE talks to the ArduinoISP (not the bootloader) during the upload of sketches. (The capacitor is needed if you’re using an Arduino Uno, and might also be necessary for an Arduino Duemilanove.)



Pin connections:

- ATtiny Pin 2 to Arduino Pin 13 (or SCK of another programmer)
- ATtiny Pin 1 to Arduino Pin 12 (or MISO of another programmer)
- ATtiny Pin 0 to Arduino Pin 11 (or MOSI of another programmer)
- ATtiny Reset Pin to Arduino Pin 10 (or RESET of another programmer)

Programming the ATtiny45 or ATtiny85

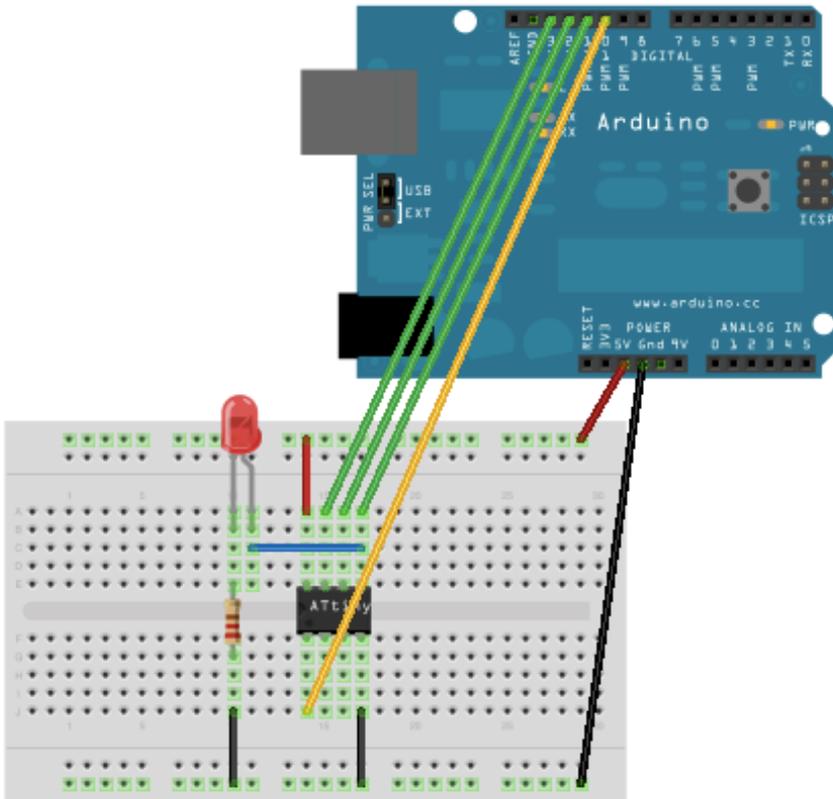
Next, we can use the Arduino as an ISP to upload a program to the ATtiny45/85:

- Open the Blink sketch from the examples menu.
- Change the pin numbers from 13 to 0.
- Select “ATtiny45 (w/ Arduino as ISP)” or “ATtiny85 (w/ Arduino as ISP)” from the Tools > Board menu (leave the serial port set to that of your Arduino board).
- Upload the sketch.

You’ll probably get the following message, but don’t worry, the upload should have worked anyway:

```
avrdude: please define PAGEL and BS2 signals in the configuration file for part ATtiny85
avrdude: please define PAGEL and BS2 signals in the configuration file for part ATtiny85
```

Now, connect an LED and see if it blinks.



Reference

The following Arduino commands should be supported:

- [pinMode\(\)](#)
- [digitalWrite\(\)](#)
- [digitalRead\(\)](#)
- [analogRead\(\)](#)
- [analogWrite\(\)](#)
- [shiftOut\(\)](#)
- [pulseIn\(\)](#)
- [millis\(\)](#)
- [micros\(\)](#)
- [delay\(\)](#)
- [delayMicroseconds\(\)](#)

high-low tech

[Arduino board as ATtiny programmer](#)

This tutorial shows you how to wire up an Arduino used as an ISP (programmer) to an ATtiny45 or ATtiny85. To find out how to write code for the microcontroller, see the the tutorial on [programming an ATtiny using the Arduino 1.0.1 software](#) (or the [older version for Arduino 0022](#)).

Materials and Tools

For this tutorial, you'll need:

- [Arduino Uno or Duemilanove](#) (w/ an ATmega328, not an older board with an ATmega168)
- [ATtiny45](#) or [ATtiny85](#) (8-pin DIP package)
- a 10 uF capacitor (e.g. [from Sparkfun](#) or [from Digi-Key](#))
- a breadboard
- jumper wires

For more information, see our [list of materials and parts](#) and our [list of prototyping supplies](#).

Download and Install the Arduino Software

You'll need the [Arduino software](#) (version 1.0.1 or 0022). Installation instructions for the Arduino software are available [for Windows](#) and [for Mac OS X](#).

Turning the Arduino board into a programmer

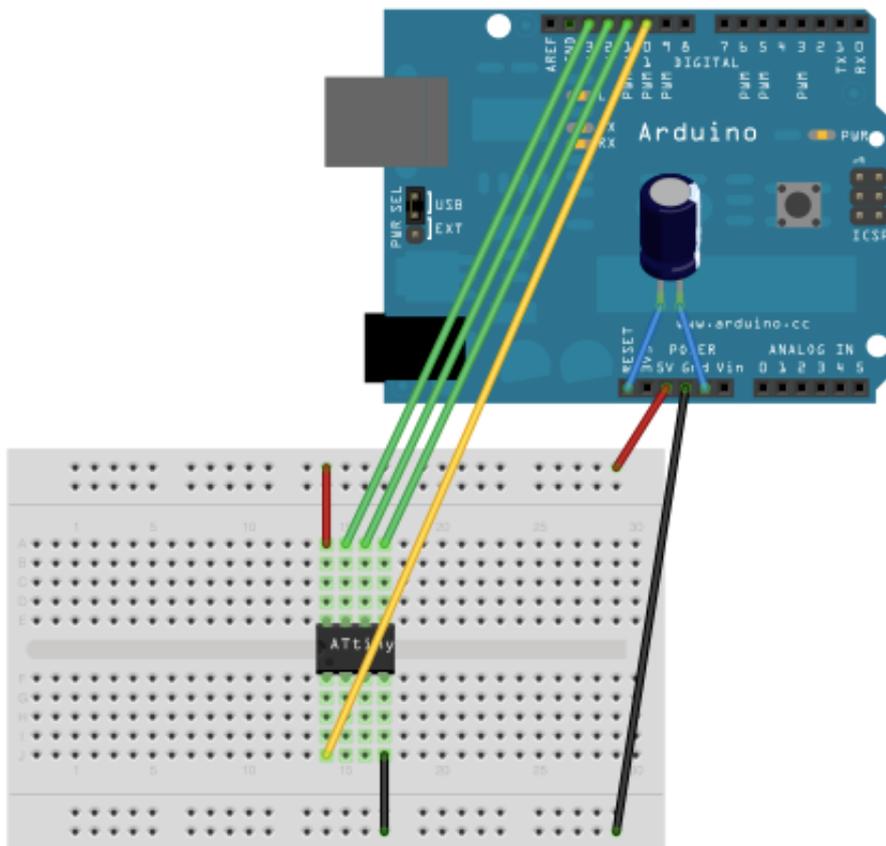
We're going to use the Arduino board to program the ATtiny. First, we'll need to turn the Arduino board into an "in-system programmer" (ISP). To this:

- Run the Arduino development environment.
- Open the ArduinoISP sketch from the examples menu.
- **Note for Arduino 1.0:** you need to make a small change to the ArduinoISP sketch before uploading it. Find the line in the heartbeat() function that says "delay(40);" and change it to "delay(20);".
- Select the board and serial port that correspond to your Arduino board.
- Upload the ArduinoISP sketch.

Connecting the Arduino board and the ATtiny

We'll connect the Arduino board to the ATtiny as shown in the following diagram. Use the dot in the corner of the ATtiny to orient it properly. We'll also connect a 10 uF capacitor between reset and

ground on the Arduino board as shown in the diagram (the stripe on the capacitor that's marked with a negative sign (-) should go to ground). The capacitor prevents the Arduino board from resetting (which starts the bootloader), thus ensuring that the Arduino IDE talks to the ArduinoISP (not the bootloader) during the upload of sketches. (The capacitor is needed if you're using an Arduino Uno, and might also be necessary for an Arduino Duemilanove.)



Pin connections:

- ATtiny Pin 2 to Arduino Pin 13 (or SCK of another programmer)
- ATtiny Pin 1 to Arduino Pin 12 (or MISO of another programmer)
- ATtiny Pin 0 to Arduino Pin 11 (or MOSI of another programmer)
- ATtiny Reset Pin to Arduino Pin 10 (or RESET of another programmer)

Links

- [Programming an ATtiny using the Arduino 1.0 software.](#)
- [Older version for Arduino 0022.](#)

high-low Yeah

[Programming an ATtiny w/ Arduino 1.0](#)

This tutorial shows you how to program an ATtiny45, ATtiny85, ATtiny44 or ATtiny84 microcontroller using the Arduino software. These are small, cheap (\$2-3) microcontrollers that are convenient for running simple programs. The ATtiny45 and ATtiny85 have eight legs and are almost identical, except that the ATtiny85 has twice the memory of the ATtiny45 and can therefore hold more complex programs. The ATtiny44 and ATtiny84 have 14-legs and more inputs and outputs. *Thanks to Mark Sproul for his work on making the Arduino core portable across processors.*

ATtiny45/85 vs. an Arduino Board

The ATtiny45 or 85 is a great option for running simple Arduino programs: it's small, cheap and relatively easy to use. It does, however, have some limitations relative to the ATmega328P on an Arduino Uno. There are fewer pins, meaning you can't connect as many components. There's less flash memory (4KB or 8KB instead of 32KB), meaning your programs can't be as big. There's less RAM (256 or 512 bytes instead of 2KB), meaning you can't store as much data. And there's no hardware serial port or I2C port (Wire library), making communication trickier. (There are workarounds, like the SoftwareSerial library or the TinyWire library, but they're not as robust and flexible.)

In short, then, if your project requires only a few simple inputs and/or outputs, you're probably fine using an ATtiny. If you're trying to hook up more components or do more complex communication or data processing, though, you're probably better off with something like the ATmega328P on an Arduino Uno. If you want something smaller and cheaper than a full Arduino board, you might try using an [ATmega328P on a breadboard](#) instead.

Materials and Tools

For this tutorial, you'll need:

- An in-system programmer (ISP), a piece of hardware used to load programs onto the ATtiny. Options include:
 - The [Tiny AVR Programmer](#), a board we developed with SparkFun specifically for programming ATtiny45/85s.
 - Another commercial programmer like the [AVRISP mkII](#) or [USBtinyISP](#).
 - a Arduino Uno or Duemilanove (w/ an ATmega328, not an older board with an ATmega168). See this tutorial [for using an Arduino board as a programmer](#)
- [ATtiny45](#) or [ATtiny85](#) (8-pin DIP package) or an ATtiny44 or ATtiny84.
- a solderless breadboard and jumper wires (unless you're using the TinyProgrammer w/ the ATtiny45 or 85)

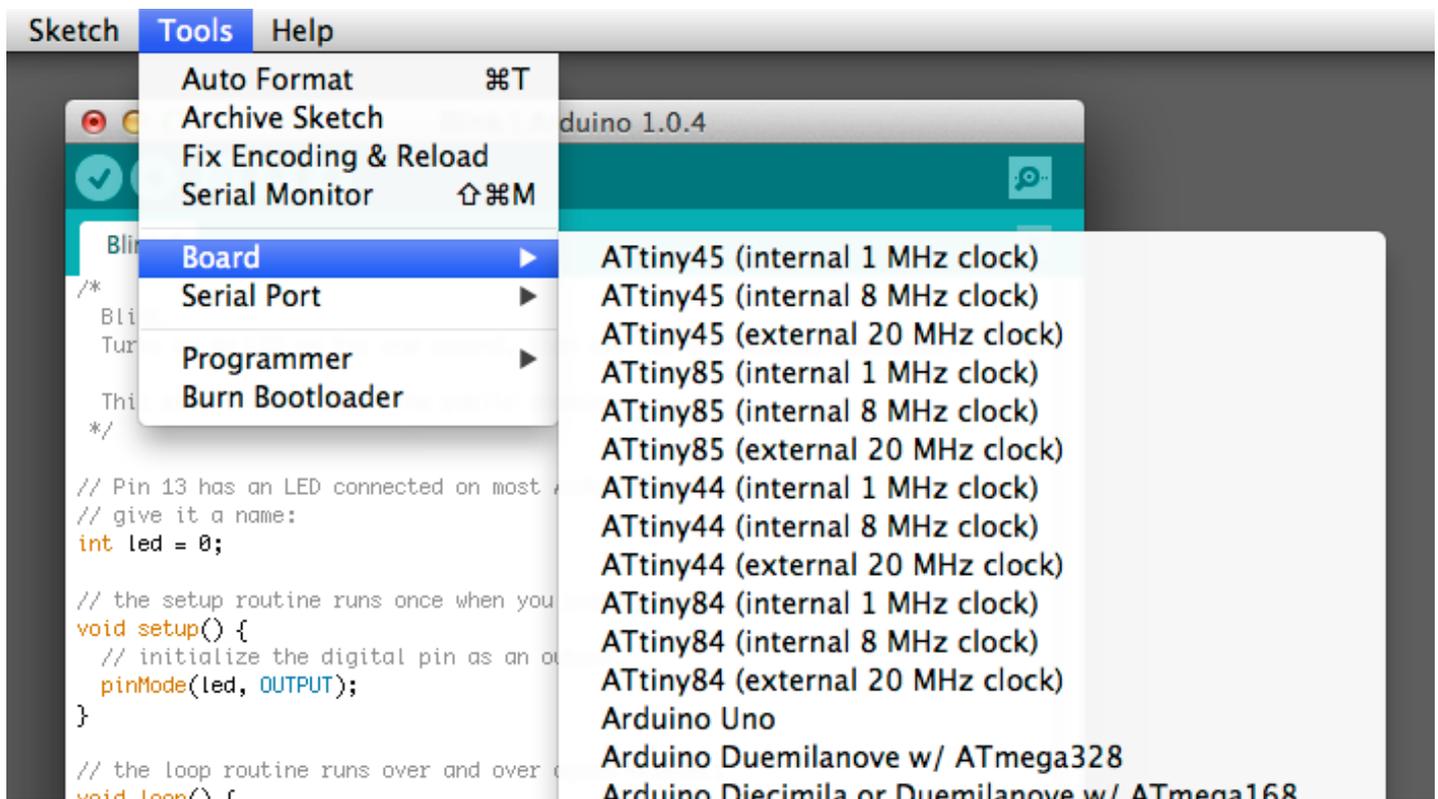
For more information, see our [list of materials and parts](#) and our [list of prototyping supplies](#).

Download

- [ATtiny master.zip](#) (hosted by GitHub)

Installing ATtiny support in Arduino

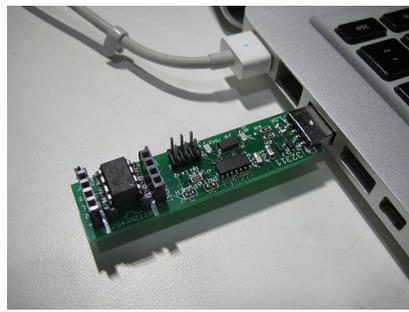
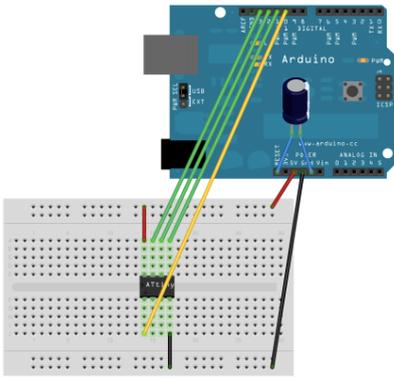
- If you haven't already, [download the Arduino software](#), version 1.0.4 (1.0.3 and 1.0.1 should work too, but *not* 1.0.2). Install the Arduino software, following the instructions [for Windows](#) or [for Mac OS X](#).
- Download the ATtiny master.zip file from the link above.
- Unzip the attiny master.zip file. It should contain an “attiny-master” folder that contains an “attiny” folder.
- Locate your Arduino sketchbook folder (you can find its location in the preferences dialog in the Arduino software)
- Create a new sub-folder called “hardware” in the sketchbook folder, if it doesn't exist already.
- Copy the “attiny” folder (not the attiny-master folder) from the unzipped ATtiny master.zip to the “hardware” folder. You should end up with folder structure like **Documents > Arduino > hardware > attiny** that contains the file **boards.txt** and another folder called **variants**.
- Restart the Arduino development environment.
- You should see ATtiny entries in the Tools > Board menu.



Connecting the ATtiny

You'll need to provide power to the ATtiny and connect it to your programmer. That is, connecting MISO, MOSI, SCK, RESET, VCC, and GND of the programmer to the corresponding pins on the ATtiny. (Or, if you're using an circuit w/ an ATtiny, simply connect the programmer to the ISP header on the board – you may also need to power the board separately.)

Instructions and diagrams are available for:



[using the TinyProgrammer](#)

[connecting an Arduino \(as ISP\) to an ATtiny.](#)

Programming the ATtiny

Next, we can use the Arduino as an ISP to upload a program to the ATtiny:

- Open the Blink sketch from the examples menu.
- Change the pin numbers from 13 to 0.
- Select the appropriate item from the Tools > Board menu (leave the serial port set to that of your Arduino board).
- Select the appropriate item from the Tools > Programmer menu (e.g. “Arduino as ISP” if you’re using an Arduino board as the programmer, USBtinyISP for the USBtinyISP, FabISP, or TinyProgrammer, etc).
- Upload the sketch.

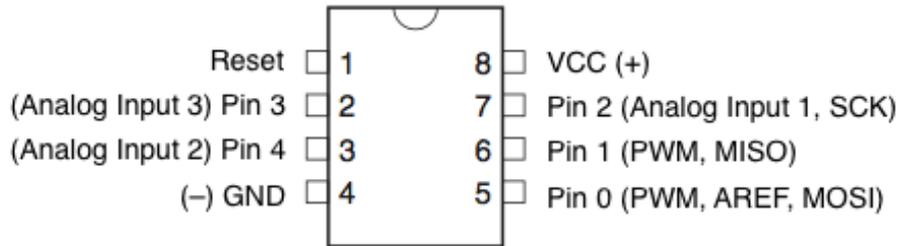
You should see “Done uploading.” in the Arduino software and no error messages. If you then connect an LED between pin 0 and ground, you should see it blink on and off. Note that you may need to disconnect the LED before uploading a new program.

Configuring the ATtiny to run at 8 MHz (for SoftwareSerial support)

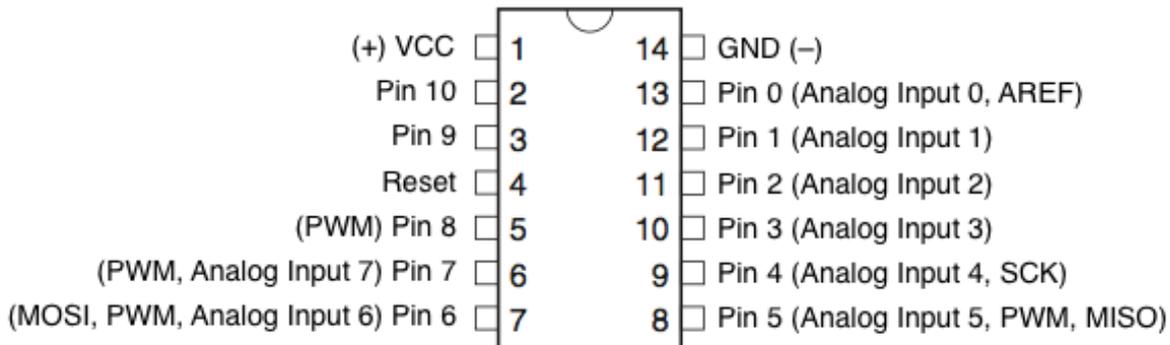
By default, the ATtiny’s run at 1 MHz (the setting used by the unmodified “ATtiny45”, etc. board menu items). You need to do an extra step to configure the microcontroller to run at 8 MHz – necessary for use of the SoftwareSerial library. Once you have the microcontroller connected, select the appropriate item from the Boards menu (e.g. “ATtiny45 (8 MHz)”). Then, run the “Burn Bootloader” command from the Tools menu. This configures the fuse bits of the microcontroller so it runs at 8 MHz. Note that the fuse bits keep their value until you explicitly change them, so you’ll only need to do this step once for each microcontroller. (Note this doesn’t actually burn a bootloader onto the board; you’ll still need to upload new programs using an external programmer.)

ATtiny Microcontroller Pin-Outs

ATtiny45 / ATtiny85



ATtiny44 / ATtiny84



Reference

The following Arduino commands should be supported:

- [pinMode\(\)](#)
- [digitalWrite\(\)](#)
- [digitalRead\(\)](#)
- [analogRead\(\)](#)
- [analogWrite\(\)](#)
- [shiftOut\(\)](#)
- [pulseIn\(\)](#)
- [millis\(\)](#)
- [micros\(\)](#)
- [delay\(\)](#)
- [delayMicroseconds\(\)](#)
- [SoftwareSerial](#) (has been updated in Arduino 1.0)

Suggestions and Bug Reports

To report problems or suggest changes to the ATtiny support, please use the [issues list on GitHub](#).

References

- [arduino-tiny](#): alternative approach to ATtiny support for Arduino
- [TinyWireM & TinyWireS](#): Wire (I2C / TWI) library for the ATtiny85 (using USI)
- [Servo8Bit](#): Servo library for the ATtiny45/85.

Alternative: ATmega328P on a Breadboard

If the ATtiny isn't quite powerful enough but you still want to use a bare microcontroller instead of a full Arduino board, see this tutorial on [using an ATmega328P on a breadboard](#). It allows you to use all the same functions and libraries as the Arduino Uno, but with just a microcontroller and a few small components.