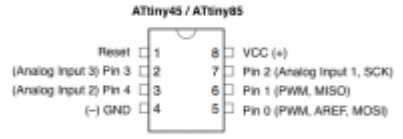
# Programming an ATtiny with Arduino ISP




OfficineArduinoTO
25/06/2014

1. Introduction
2. ATtiny45/85 vs. an Arduino Board
3. Materials and Tools
4. Installing ATtiny support in Arduino
5. Connecting the ATtiny
6. Programming the ATtiny
7. Configuring the ATtiny to run at 8 MHz (for SoftwareSerial support)
8. ATtiny Microcontroller Pin-Outs
9. Reference

## 1. Introduction

This tutorial shows you how to program with the new Arduino ISP 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.*

## 2. 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.

## 3. Materials and Tools

For this tutorial, you'll need:

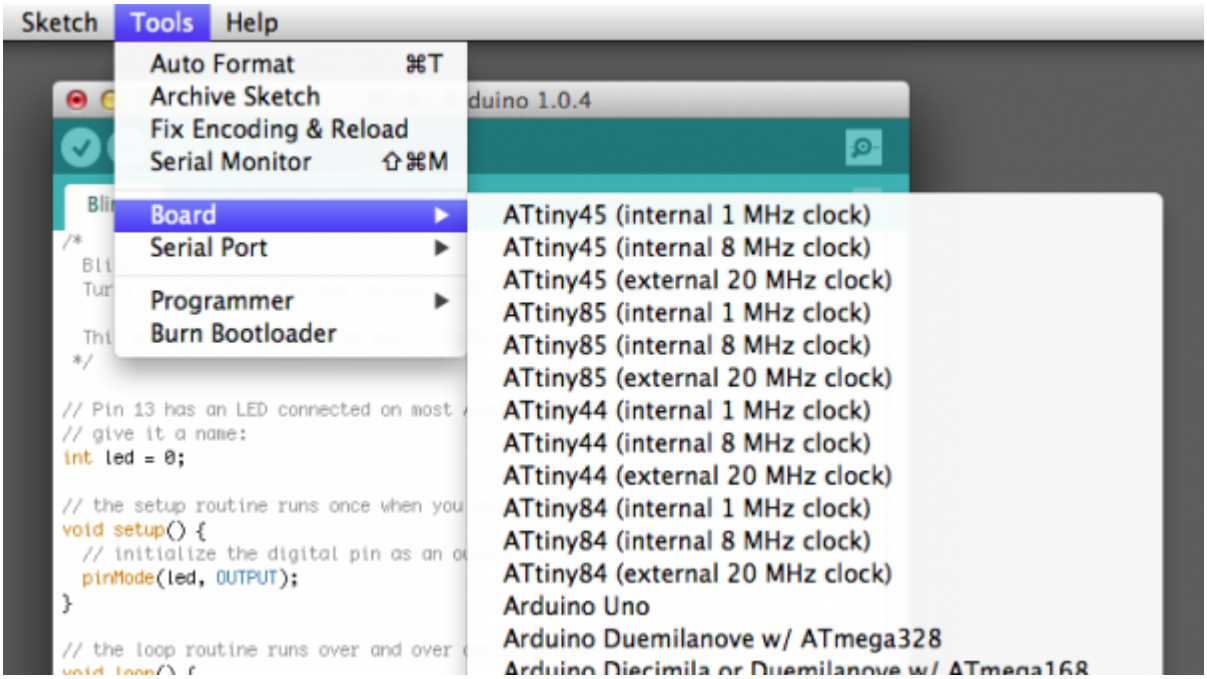
- An in-system programmer (ISP), a piece of hardware used to load programs onto the ATtiny. Options include:
  - the new Arduino ISP,
  - a Arduino Uno or Duemilanove (w/ an ATmega328, not an older board with an ATmega168). See this tutorial for using an Arduino board as a programmer
  - Another commercial programmer
- ATtiny45 or ATtiny85 (8-pin DIP package) or an ATtiny44 or ATtiny84.
- a solderless breadboard and jumper wires

## 4. Installing ATtiny support in Arduino

- If you haven't already, download the Arduino software, version 1.0.5 (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, for Mac OS X or for Linux.
- Download the ATiny support for Arduino IDE
- 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 > 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.




## 5. 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.

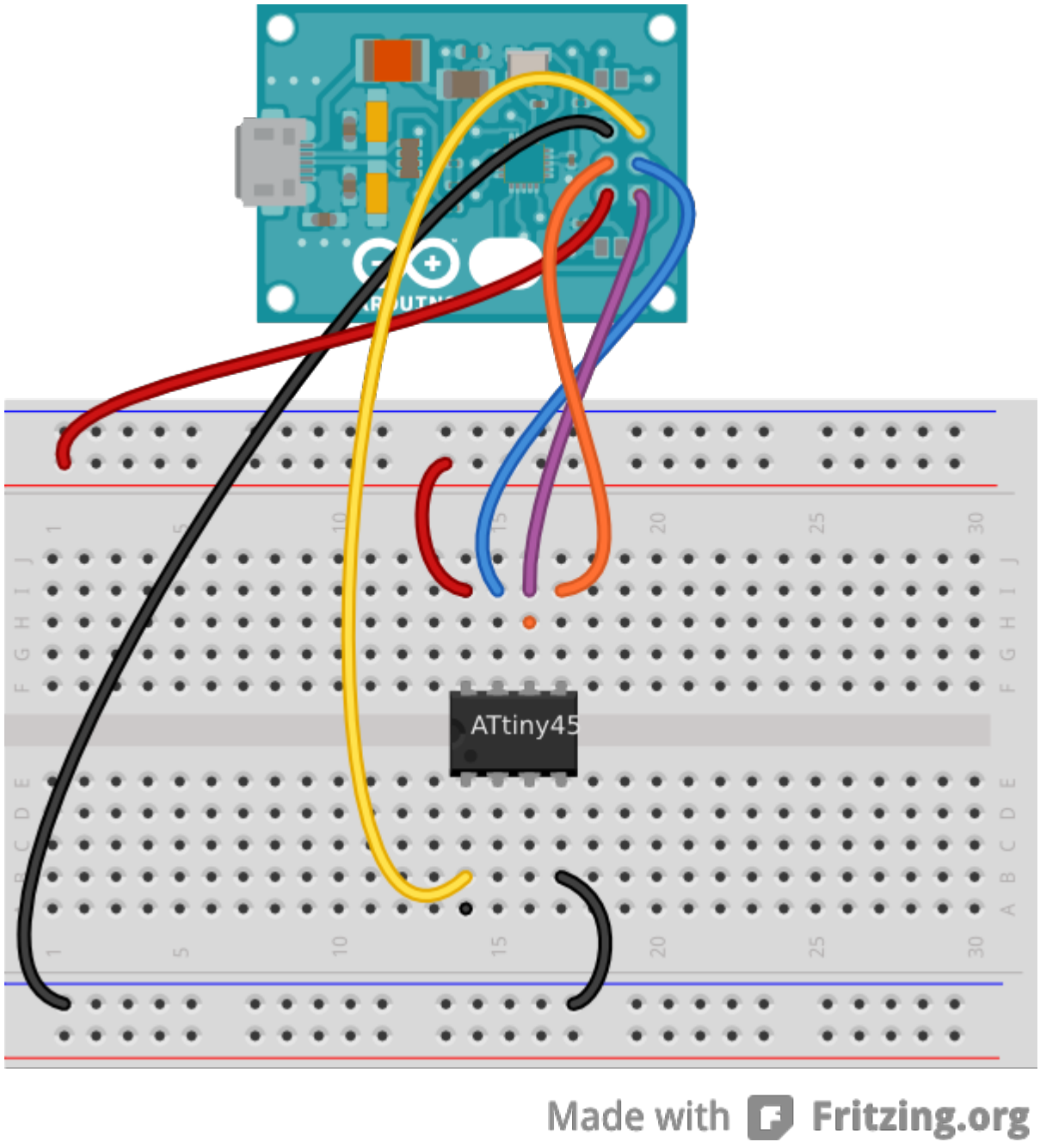
## 6. Programming the ATtiny

Next, we can use the Arduino 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 ISP).
- Select Arduino ISP from the Tools

- 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.

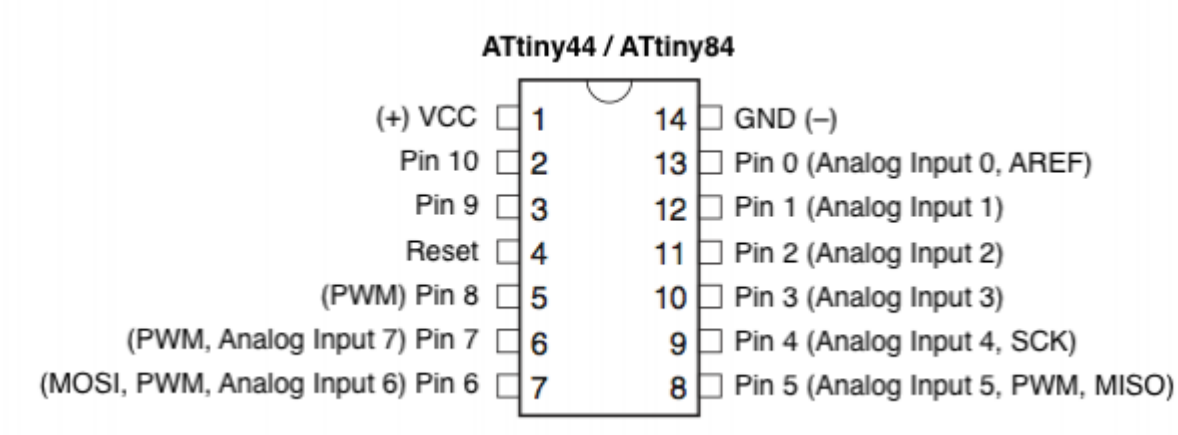


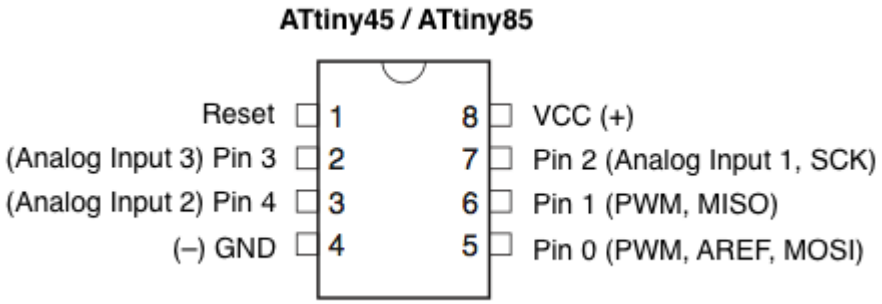
## 7. Configuring the ATtiny to run at 8 MHz (for SoftwareSerial support)

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).

## 8. ATtiny Microcontroller Pin-Outs

ATtiny 44/84 and 45/85 pin-outs






## 9. 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)

This tutorial originally written by David Mellis: http://highlowtech.org/?p=1695