high-low tech

- PROJECTS
- PEOPLE
- PUBLICATIONS

- TUTORIALS

  Paper Lamps with Codeable Objects
  Paper circuits w/ copper tape
  Using the Codeable Objects Processing Library
  Simple Arduino audio samples
  Using the TinyProgrammer
  Arduino board as ATtiny programmer
  Programming an ATtiny w/ Arduino 1.0.1
  Microcontroller circuit with copper tape
  Simple circuit with copper tape
  Electronic Origami Flapping Crane
  LED Dragon Kite
  Light-up Postcard
  Paper Speakers
  Salt and Vinegar Etching
  Electronics and Textiles (e-textiles)
  Programming an ATtiny w/ Arduino
  Painted Circuits
  Data Logging with uLog Module

- MATERIALS
- NEWS & EVENTS
- WORKSHOPS
- MIT CLASSES
- ADMISSIONS

NEWSLETTER

Programming an ATtiny w/ Arduino 1.0.1

This tutorial shows you how to program an ATtiny45, ATtiny85, ATtiny44 or ATtiny84 microcontroller using the Arduino software (version 1.0.1). 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*.

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:
    - A 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
    - the TinyProgrammer, a board that we've developed to make it as easy as possible to program the ATtiny45 and 85.
- 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.

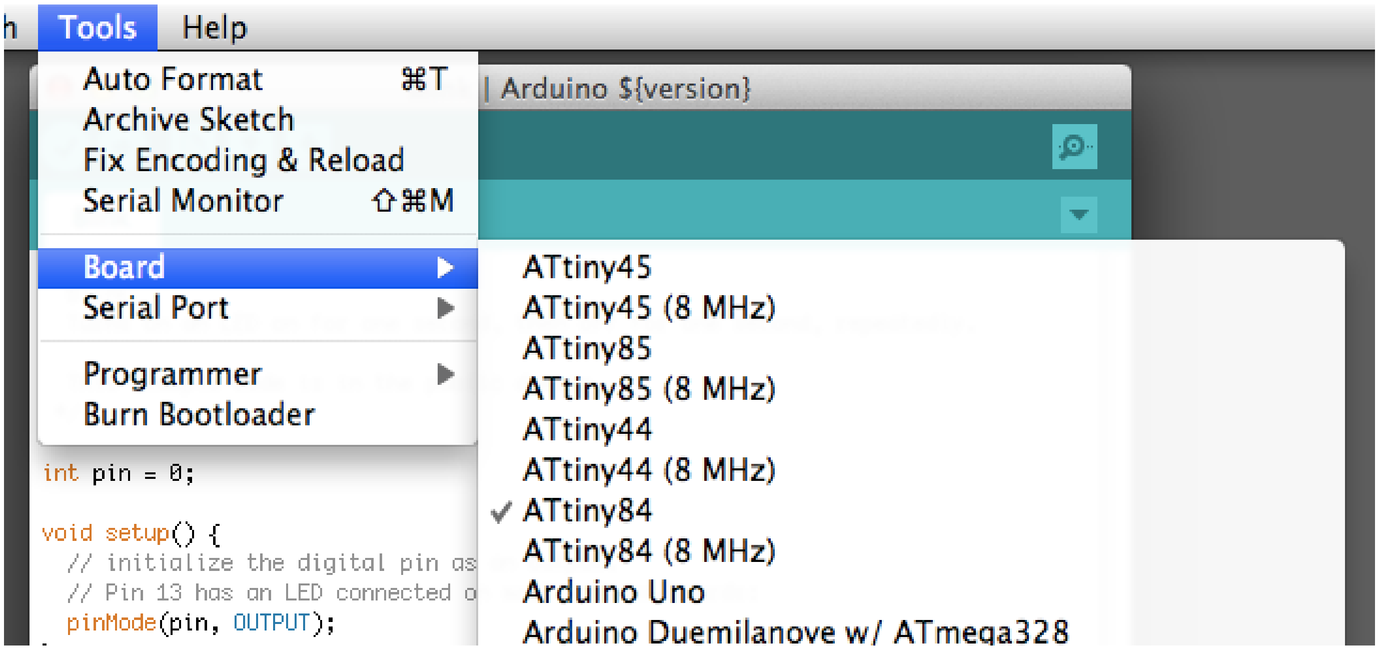
Software

You'll need the Arduino software, version 1.0.1. (Instructions for Arduino 0022 are also available.) You can download Arduino 1.0.1 from the Arduino site. Installation instructions are available for Windows and for Mac OS X.

Download: Arduino 1.0.1 and the ATtiny zip (from this GitHub repository)

Installing ATtiny support in Arduino

- Download: ATtiny (from this GitHub repository)
- 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 attiny folder from inside the .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.

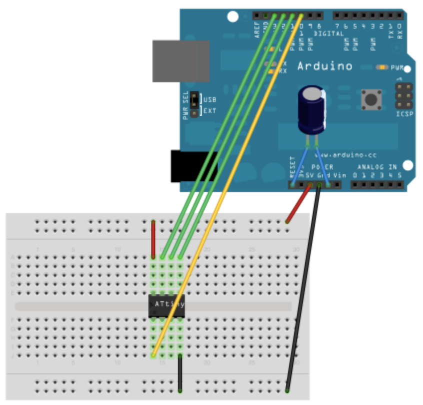


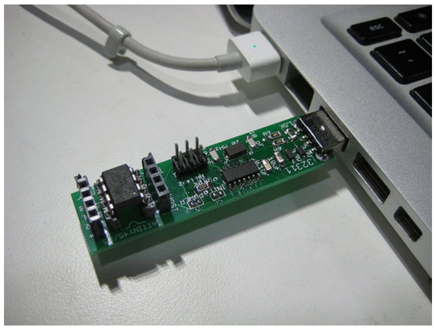
```
int pin = 0;

void setup() {
  // initialize the digital pin as
  // Pin 13 has an LED connected o
  pinMode(pin, OUTPUT);
```

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:


connecting an Arduino (as ISP) to an ATtiny.


using the TinyProgrammer

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.)

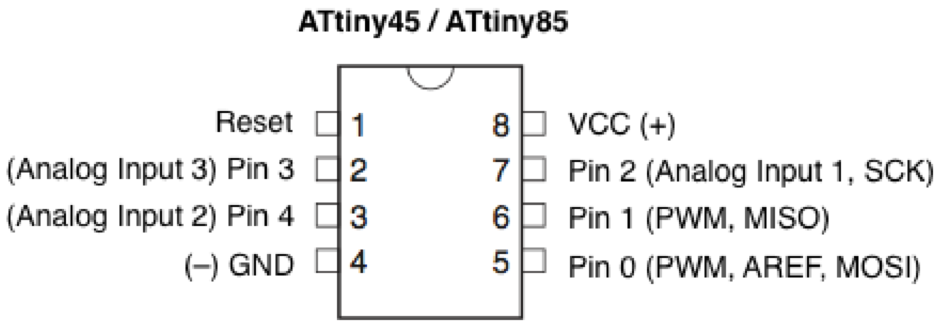
Programming the ATtiny

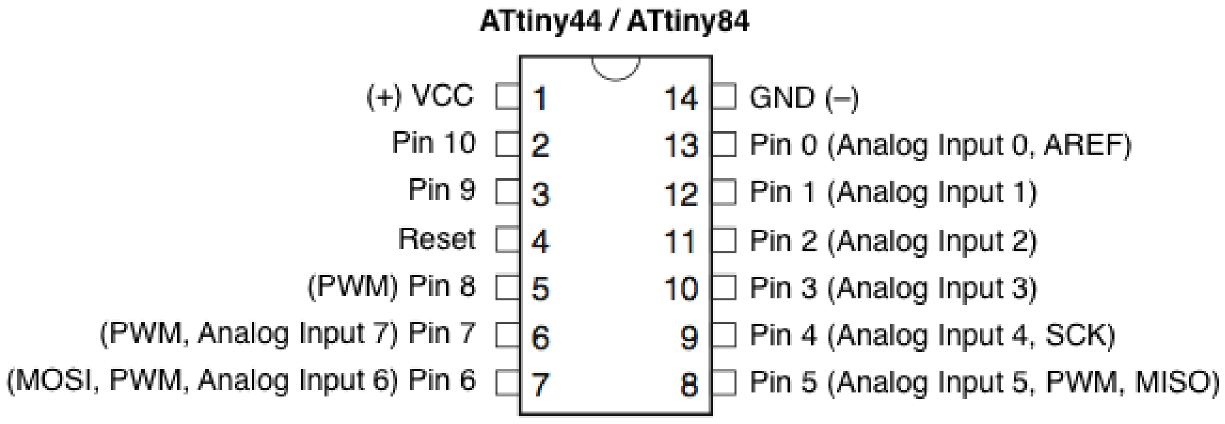
Next, we can use the Arduino as an ISP to upload a program to the ATtiny:

- 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.

ATtiny Microcontroller Pin-Outs




ATtiny44 / ATtiny84
(+) VCC 1
Pin 10 2
Pin 9 3
Reset 4
(PWM) Pin 8 5
(PWM, Analog Input 7) Pin 7 6
(MOSI, PWM, Analog Input 6) Pin 6 7
14 GND (–)
13 Pin 0 (Analog Input 0, AREF)
12 Pin 1 (Analog Input 1)
11 Pin 2 (Analog Input 2)
10 Pin 3 (Analog Input 3)
9 Pin 4 (Analog Input 4, SCK)
8 Pin 5 (Analog Input 5, PWM, MISO)

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)

References

- arduino-tiny: alternative approach to ATtiny support for Arduino
- TinyWireM & TinyWireS: Wire (I2C / TWI) library for the ATtiny85 (using USI)

High-Low Tech Group :: MIT Media Lab