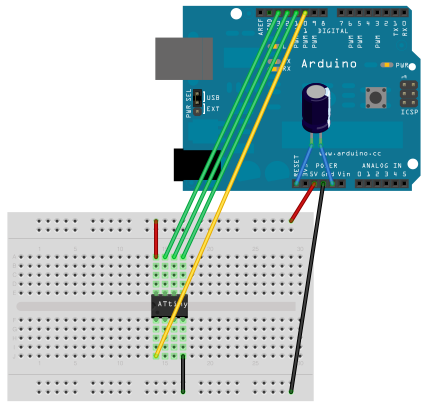
Connecting the ATtiny

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:

[[](http://highlowtech.org/?p=1706)  
connecting an Arduino (as ISP) to an ATtiny.](http://highlowtech.org/?p=1706)

[[](http://highlowtech.org/?p=1801)](http://highlowtech.org/?p=1801)

## Programming the ATtiny

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

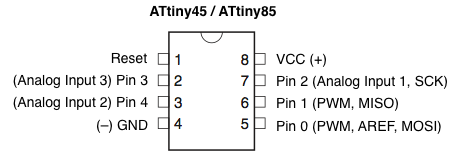
* Open the Blink sketch from the examples menu.
* Change the pin numbers from 13 to 0.
* Select “ATtiny” from the Tools > Board menu and the particular ATtiny you’re using from the Tools > Processor menu. (In Arduino 1.0.x, these options are combined in just the Tools > Board menu.)
* Select the appropriate 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.

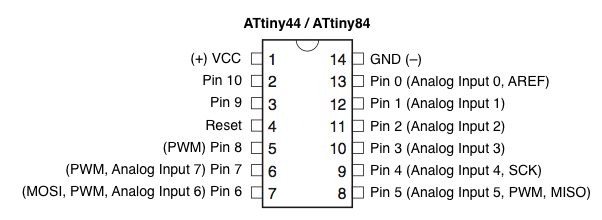
You should see “Done uploading.” in the Arduino software and no error messages. If you then connect an LED between pin 0 and ground, you should see it blink on and off. Note that you may need to disconnect the LED before uploading a new program.

## Configuring the ATtiny to run at 8 MHz

By default, the ATtiny’s run at 1 MHz. You can configure them to run at 8 MHz instead, which is useful for faster baud rates with the SoftwareSerial library or for faster computation in general. To do so, once you have the microcontroller connected, select “8 MHz (Internal)” from the Tools > Clock menu. (In Arduino 1.0.x, select the appropriate 8 MHz clock option from the main Tools > Board menu.) **Warning: make sure you select “internal” not “external” or your microcontroller will stop working (until you connect an external clock to it).** 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





Reference

The following Arduino commands should be supported:

* [pinMode()](http://arduino.cc/en/Reference/PinMode)
* [digitalWrite()](http://arduino.cc/en/Reference/DigitalWrite)
* [digitalRead()](http://arduino.cc/en/Reference/DigitalRead)
* [analogRead()](http://arduino.cc/en/Reference/AnalogRead)
* [analogWrite()](http://arduino.cc/en/Reference/AnalogWrite)
* [shiftOut()](http://arduino.cc/en/Reference/ShiftOut)
* [pulseIn()](http://arduino.cc/en/Reference/PulseIn)
* [millis()](http://arduino.cc/en/Reference/Millis)
* [micros()](http://arduino.cc/en/Reference/Micros)
* [delay()](http://arduino.cc/en/Reference/Delay)
* [delayMicroseconds()](http://arduino.cc/en/Reference/DelayMicroseconds)
* [SoftwareSerial](http://arduino.cc/en/Reference/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](https://github.com/damellis/attiny/issues).

References

* [arduino-tiny](http://code.google.com/p/arduino-tiny/): alternative approach to ATtiny support for Arduino
* [TinyWireM & TinyWireS](http://arduino.cc/playground/Code/USIi2c): Wire (I2C / TWI) library for the ATtiny85 (using USI)
* [Servo8Bit](https://github.com/fri000/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](http://arduino.cc/en/Tutorial/ArduinoToBreadboard). 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.