Software setup

- *Arduino:*
  - Install the Arduino IDE from: http://arduino.cc
  - Add ATtiny support following the directions from http://highlowtech.org/?p=1695
- *GCC/Makefiles*
  - Mac
    - Download the AvrMacPack from http://www.obdev.at/products/crosspack/index.html
    - Or use MacPorts to run `sudo port install avr-gcc avrdude avr-binutils avr-libc` (similar options available for Homebrew)
  - Windows
    - Install the FabISP drivers from https://learn.adafruit.com/usbtinyisp/drivers
    - Install WinAVR from http://sourceforge.net/projects/winavr/
    - Login then logout to apply the changes
  - Linux (Ubuntu)
    - `sudo apt-get install gcc-avr avrdude avrprog avr-libc binutils-avr build-essential`

Flashing programs

- Start with a working (programmed) FabISP or AVRISP programmer
- Connect power to the target board (and verify it doesn't start smoking)
- Connect the ISP header from your programmer to the target board, verifying cable orientation
- *Arduino:*
  - Select the appropriate board and timing source
  - Select the appropriate programmer (USBtiny is the same as FabISP)
  - Select "Burn bootloader" to set the fuses and get timing right (be careful! If you set the wrong timing source, you can leave your chip unusable. Starting with an internal clock is safer, even if there is a crystal on the board.)
  - Select "Upload using programmer" to upload your program (Plain "Upload" only works if you have an Arduino clone with a bootloader installed and hardware serial is properly connected)
- *GCC/Makefiles*
  - Start with a working Makefile and make sure the PROJECT, MMCU, F_CPU, and fuse values are set correctly. If you are starting, it's easiest to start with a Makefile that matches your chip and board design (crystal or no crystal and its speed). Then you don't need to worry about changing fuse values.
  - Set your board's fuses (this is only required if they need changing, for instance if you're using an external time source), maybe with a command like:
    `make -f hello.ftdi.44.echo.c.make program-usbtiny-fuses`
  - Flash your program, maybe with a command like:
    `make -f hello.ftdi.44.echo.c.make program-usbtiny`

Common problems

- `avrdude: initialization failed, rc=-1  Double check connections and try again`
  (or)
  `avrdude: Device signature = 0x000000 avrdude: Yikes!  Invalid device signature`

  Your target board is dead/malfunctioning/not powered, or the connection is broken. Double check for power and electrical problems on the target board. Check your programming cable for damage and make sure it's oriented correctly. Connectors are often where bad solder joints hide.

- `avrdude: usbdev_open(): did not find any USB device "usb"`

  avrdude can't find your programmer. Did you tell it to look for an AVRISP instead of a USBtiny, or vice versa? Are you sure your FabISP works? Check the AVRDUDE line in your Makefile or the Programmer option in Arduino, or try another programmer.

- `avrdude: Expected signature for ATtiny44 is 1E 92 07  Double check chip`

  avrdude thinks you should have a different chip than you do. Check the DEVICE line in your Makefile or the Board option in Arduino. Check the chip you actually put on your board. Be careful about "p" or "v" variants.

- My board is running really slowly or really quickly (My LED is blinking 8X too slowly).

  Double check your board timing options and the fuses you've set. In your Makefile, check the F_CPU and the FUSE values, consulting here: http://www.ladyada.net/learn/avr/fuses.html. In Arduino, make sure you have the correct board variant. For both, ensure you've set your fuses (Something like "make fuse" or "Burn bootloader").

- My board programmed successfully, but what I want to see isn't happening (my LED isn't blinking, etc.).

  This could be many problems, but first check your pin assignments, especially with Arduino. The smaller boards have pin mappings that are different from the big boards, and Arduino pin numberings don't always match the physical order on the chip (see David Mellis's HLT page, linked above). After that, check your schematic, hardware, and overall program logic. Try to start with the simplest possible examples and go from there, or with any existing piece of code that is known to work.

See also:
http://fab.cba.mit.edu/classes/863.14/tutorials/electronics_production/fab_isp.html
https://learn.adafruit.com/usbtinyisp/help