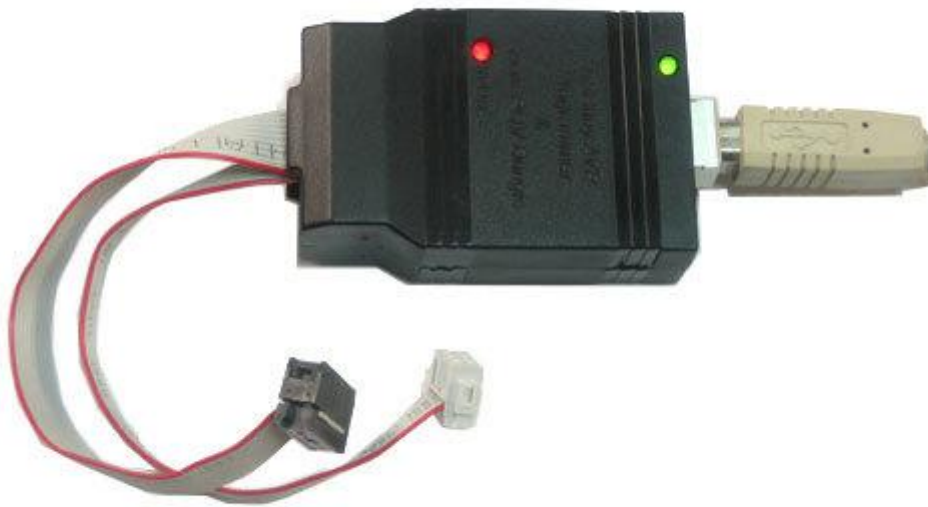




USBtinyISP

Created by lady ada



<https://learn.adafruit.com/usbtinyisp>

Last updated on 2022-12-01 01:58:34 PM EST

Table of Contents

Overview	5
<ul style="list-style-type: none">• AVR programmer & SPI interface• Introduction• What's so great about it?• Easy to make• Easy to use• Easy to power• Easy to extend	
F.A.Q.	7
Make It!	9
<ul style="list-style-type: none">• Make Make Make• Step-by-step	
Preparation	9
<ul style="list-style-type: none">• Get ready to solder, sports-racers!• Tutorials• Tools	
Parts list	13
<ul style="list-style-type: none">• Kit parts list• Schematic (v2.0)	
Solder it!	23
<ul style="list-style-type: none">• Ouch, hot!• Make 6-pin cable (Old kits, without pre-made 6-pin cables)• Case	
Parts (v1.0)	45
<ul style="list-style-type: none">• Kit parts list• This is the list for very old kits, its very unlikely you have a v1.0 but we're keeping it around for historical record	
Solder (v1.0)	54
<ul style="list-style-type: none">• V1.0 instructions!• Its very unlikely you have a v1.0 but we're keeping it around for historical record• Solder it!• Make 6-pin cable• Case	
Use It!	70
<ul style="list-style-type: none">• How to use it!• Indicator LEDs• Programming Cables• Jumper JP3 (USB power to target)• Using it as an SPI interface	
User Manual	72
<ul style="list-style-type: none">• How to use it!• Indicator LEDs• Programming Cables	

- Jumper JP3 (USB power to target)
- Using it as an SPI interface

Drivers 74

- AVR programmer & SPI interface
- Windows 7, 8 & XP
- Mac OS X & Linux

AVRDUDE 77

- Using the programmer with AVRDUDE
- For Windows
- For Mac OS X
- For Linux

AVRStudio 81

- AVRISP/STK500v2 compatibility bridge
- Notes
- Step 1. COM bridge
- Step 2. Install AVRStudio
- Step 3. Download USBtiny500

Download 91

- Windows Drivers
- AVRDUDE
- Hardware/Firmware Files for v2.0
- Hardware/Firmware Files for v1.0
- USBtiny500 compatibility bridge for AVR Studio

HELP!!! 93

- Frequently Asked Questions
- I'm running avrdude and I get "Initialization failed, rc=1"
- Check:
- It doesn't work with a USB 3 port
- I'm running avrdude and I get "error: usbtiny_transmit: initialization failed, rc=-1"
- It's not working!
- I'm running avrdude and I get "USB read error: expected 4, got -1" (or something similar)
- My 64 bit computer doesn't seem to work!
- I'm having trouble compiling/burning the chip for this project...
- I'm running avrdude and I get "Could not find USB device 0x1781/0xc9f"
- I'm having trouble building this project from scratch...
- I'm running avrdude and I get "error at avrdude.conf:370 unrecognized character: 'u'"
- I'm running avrdude and I get 'avrdude: Can't find programmer id "usbtiny"

Overview

AVR programmer & SPI interface



Introduction

This is documentation for a simple open-source USB AVR programmer and SPI interface. It is [low cost \(\)](#), [easy to make \(\)](#), works great with [avrdude \(\)](#), is [AVRStudio-compatible \(\)](#) and tested under Windows, Linux and MacOS X. Perfect for students and beginners, or as a backup programmer.

The project is based off of the [USBtiny code & design \(\)](#). The main improvements are: adjusting the code to allow it to act as a [SpokePOV \(\)](#) interface, adding lowlevel bitbang commands, and addition of a "USB good" LED. Other changes are new VID/PID (to make it official), removing some of the commands, and moving around the pins a bit.

You can build this design using the [schematic and firmware \(\)](#), or buy a kit from the [Adafruit webshop \(\)](#). Having a full kit available solves the "chicken & egg" problem of purchasing or building a USB programmer that then needs a programmer of some sort to 'kick start'. (See [USBasp \(\)](#), [AVRdoper \(\)](#), [USBprog \(\)](#))

All the firmware code is distributed under the GPL, the hardware design layout files are [CC 2.5 Attrib./Share-alike \(\)](#).

What's so great about it?

Easy to make

- Ultra low cost: programmer is \$16 in parts, less than half the price of the AVRISP v2 ! (Kits are \$22 and [available from the adafruit shop \(\)](#).)
- Kit comes with both 6-pin and 10-pin AVR-standard connectors and cables. Almost no programmers that are not from Atmel have both! (Including the AVRISP v2)
- Easy to build: All through-hole parts, all common and available from large distributors.

Easy to use

- AVRdude compatible - [support for usbtiny added in v5.5! \(\)](#)
- [USB drivers available for Windows \(\)](#) using libusb, no drivers needed for Mac OS X or Linux.
- Durable off-the-shelf enclosure
- High speed! Max clock rate is 400KHz. Write speed:1Kb/s, read speed: 2Kb/s. (Atmega8 takes 8s to write, 4s to read/verify)
- 2 LEDs to indicate "USB/Power good" and "Busy"
- I/O is buffered to allow programming of 2V-6V targets (v2)
- Works with any AVR ISP chip with 64K of flash (or less) - does not work with Atmega1281/1280/2561/2560

Easy to power

- Powered off of 5V USB bus at less than 100mA to allow it to be used with unpowered USB hubs
- Easily accessible jumper to power target project off of USB (target must be 5V tolerant, of course)
- Remove the jumper and it will self-power but buffer the I/O to match the target device. (v2)

Easy to extend

- Easily interfaced with libusb
 - Existing firmware allows for fast SPI interfacing using USB
 - Bit-bang commands provide 8 bits of I/O control (including LED) for open-ended project ideas
-

F.A.Q.

Is this sold as an assembled programmer?

Not yet, it is only available as a kit.

How hard is it to assemble?

There are very clear instructions available in the "[Make it! \(\)](#)" link. It's a simple kit and should be fairly easy for anyone with proper tools even if it's their first soldering project.

Does this work with linux?

Yes. We have tested it with linux (Ubuntu 7.04) and it didn't require anything strange so it should work with any distribution. If you're having problems make sure you are running as root to have permissions on the device.

Why is there no Serial/COM/port (or /dev/ttyXX device) ?

USBtiny is not a USB-Serial device, it is its own USB protocol which is understood by Avrdude. You will not see a COM port or Serial port created when you plug it in.

Can I send serial messages using the USBtiny as well as programming, like an Arduino?

No, the USBtiny does not create a serial port and cannot do that. It programs chips directly, using the ISP connection, not Serial. Arduinos are not AVR programmers, they are an AVR with a bootloader that runs over a serial port.

What chips can be programmed?

Any AVR that uses the ISP interface for programming and has 64K or less of flash can be programmed.

Chips such as the Atmega1280/1281 and Atmega2560/2561 have more than 64K and cannot be programmed.

Chips that use TPI interface, such as Attiny4/5/9/10 cannot be programmed.

Some very old chips such as the AT90S1200 and similar cannot be programmed

Can I program a bootloader (like an Arduino one) with USBtinyISP?

Yes, this is what an AVR programmer can do. We suggest using the 'built in' bootloader-burner in the IDE to do it.

How do I program a bootloader onto an Arduino?

1. Put a fresh AVR chip (such as an Atmega328) into the Arduino in the correct orientation
 2. Remove the jumper from the USBtinyISP
 3. Plug in the USBtiny to USB
 4. Plug the Arduino into DC or USB so it is powered
 5. Plug the 6 pin cable from the USBtinyISP into the Arduino so that pin 1 mark is lined up with the red wire on the cable
 6. Start up Arduino IDE
 7. Select the chip/Arduino you are using in the Tools->Board menu
 8. Do not select a COM/Serial port
 9. Select Tools->Burn Bootloader->w/USBtinyISP
 10. The USBtinyISP red LED should light up. It will take a minute or two to program the chip
 11. When it is done, the IDE will tell you it has completed and the red LED will be off.
-

I need help getting this working

Check the [HELP! page \(\)](#).

Does this work with the 8051-core (AT89) series chips?

The USBtinyISP design as-is only works with the AVR core chips (ATtiny/ATmega/ etc). [However Lucas Chiesa and his peers have done an excellent job porting this version to support 8051-core chips. \(\)](#)

What is "Self Program"?

The original USBtinyISP could be programmed by another programmer by jumpering a pin. This is not true anymore now that there is a buffer. You should ignore the jumper.

Also, you cannot program the usbtiny with itself.

Make It!

Make Make Make

Step-by-step

Making a USBtinyISP from kit is easy, just follow these steps:

1. [Preparation and tutorials \(\)](#)
 2. [Parts list check \(\)](#) (v2.0) or if you have an older version, [parts list for v1.0 \(\)](#)
 3. [Solder it together! \(\)](#) (v2.0) or if you have an older version, [assembly for v1.0 \(\)](#)
-

Preparation

Get ready to solder, sports-racers!

Tutorials

[Learn how to solder with tons of tutorials! \(\)](#)

[Don't forget to learn how to use your multimeter too! \(\)](#)

Tools

There are a few tools that are required for assembly. None of these tools are included. If you don't have them, now would be a good time to borrow or purchase them. They are very very handy whenever assembling/fixing/modifying electronic

devices! I provide links to buy them, but of course, you should get them wherever is most convenient/inexpensive. Many of these parts are available in a place like Radio Shack or other (higher quality) DIY electronics stores.

Soldering iron



Any entry level 'all-in-one' soldering iron that you might find at your local hardware store should work. As with most things in life, you get what you pay for.

Upgrading to a higher end soldering iron setup, like the [Hakko FX-888 that we stock in our store \(http://adafru.it/180\)](http://adafru.it/180), will make soldering fun and easy.



Do not use a "ColdHeat" soldering iron! They are not suitable for delicate electronics work and can damage the kit ([see here \(\)](#)).

[Click here to buy our entry level adjustable 30W 110V soldering iron \(http://adafru.it/180\)](http://adafru.it/180).

[Click here to upgrade to a Genuine Hakko FX-888 adjustable temperature soldering iron. \(http://adafru.it/303\)](http://adafru.it/303)

Solder

You will want rosin core, 60/40 solder. Good solder is a good thing. Bad solder leads to bridging and cold solder joints which can be tough to find.



[Click here to buy a spool of leaded solder \(recommended for beginners\) \(http://adafru.it/145\)](http://adafru.it/145).

[Click here to buy a spool of lead-free solder \(http://adafru.it/734\)](http://adafru.it/734).



Multimeter

You will need a good quality basic multimeter that can measure voltage and continuity.

[Click here to buy a basic multimeter. \(http://adafru.it/71\)](http://adafru.it/71)

[Click here to buy a top of the line multimeter. \(http://adafru.it/308\)](http://adafru.it/308)

[Click here to buy a pocket multimeter. \(http://adafru.it/850\)](http://adafru.it/850)





Flush Diagonal Cutters

You will need flush diagonal cutters to trim the wires and leads off of components once you have soldered them in place.

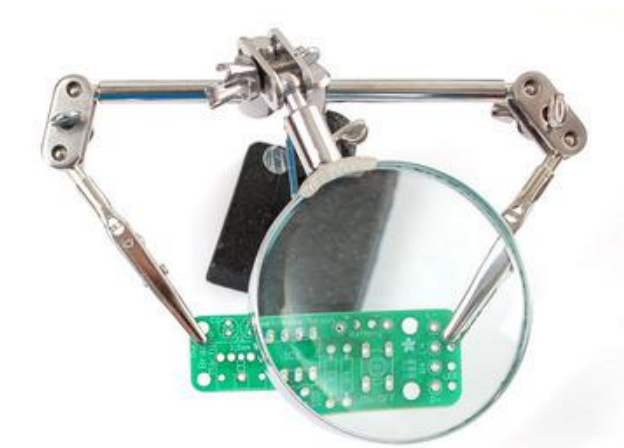
[Click here to buy our favorite cutters \(http://adafru.it/152\).](http://adafru.it/152)



Solder Sucker

Strangely enough, that's the technical term for this desoldering vacuum tool. Useful in cleaning up mistakes, every electrical engineer has one of these on their desk.

[Click here to buy a one \(http://adafru.it/148\).](http://adafru.it/148)



Helping Third Hand With Magnifier

Not absolutely necessary but will make things go much much faster, and it will make soldering much easier.

[Pick one up here \(http://adafru.it/291\).](http://adafru.it/291)

Good light. More important than you think.

Parts list





Kit parts list

Check to make sure your kit comes with the following parts. Sometimes we make mistakes so double check everything and email support@adafruit.com if you need replacements!

Image	Name	Description	Part #
		Preprogrammed when purchased in a kit)	ATTINY2313-20PU
	IC2	Buffer chip (New in v2)	74AHC125

	XTL1	12.00 MHz ceramic oscillator Make sure it says 12.00 on it	ZTT-12.00MT
---	------	--	-----------------------------

	C1	Bypass 0.1uF capacitor (104) Might be blue	Generic
	C2	Bypass 100uF/6.3V electrolytic capacitor (photo shows 10V but 6.3V is fine) New in v2.0	Generic
	R10	10K 1/4W 5% resistor (brown, black orange gold)	

	R3, R4, R5, R6, R7	1.5K 1/4W 5% resistor (brown green red gold)	
	R1, R2	27-68 ohm 1/4W 5% resistor	
			
		LED1 Red 3mm LED Generic	
		LED2 Green 3mm LED	



D2 3.6V Zener diode

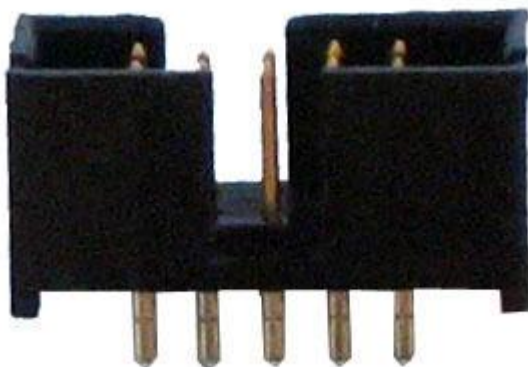
[1N5227B](#)



X1

USB type B male jack




[Generic](#)



JP2

10 pin box header

[3M 30310-6002HB](#)

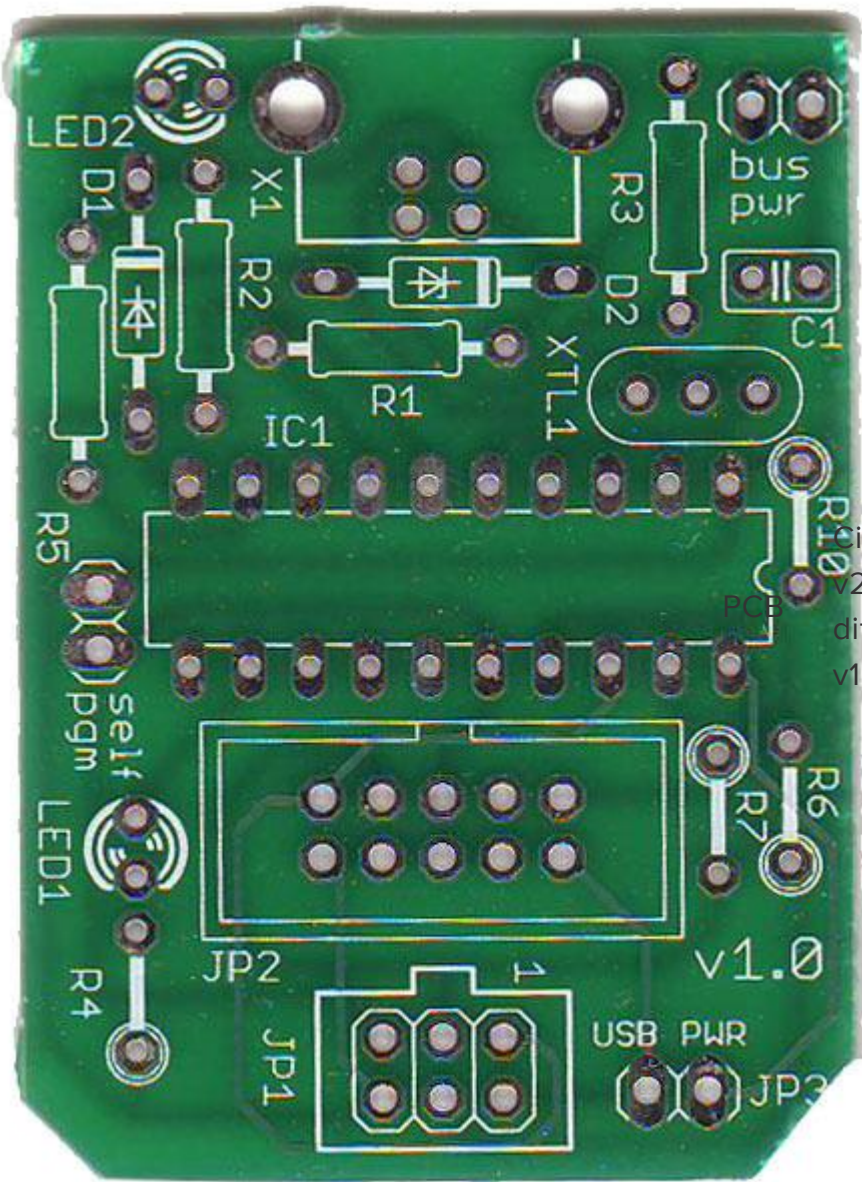
	JP1	6 pin straight header (0.1" x 0.1")	Molex 10-89-7062
	JP3	2 pin right angle header	Tyco 640453-2
	JP3'	Jumper/Shunt	Generic



10 pin IDC cable [Generic Cable](#)



6 pin IDC cable [Generic Cable](#)

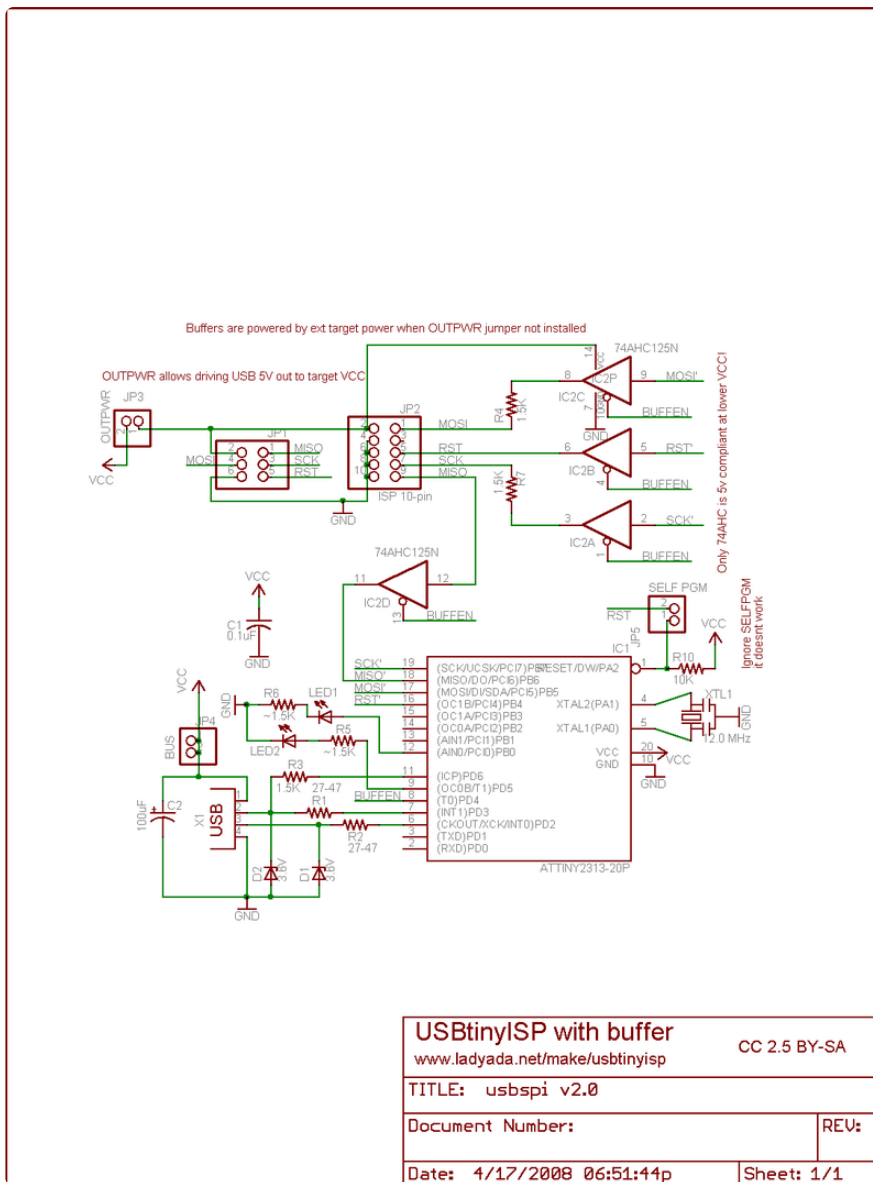


Circuit board v2.0 looks different than v1.0



Schematic (v2.0)

[Click to enlarge...](#)



Solder it!

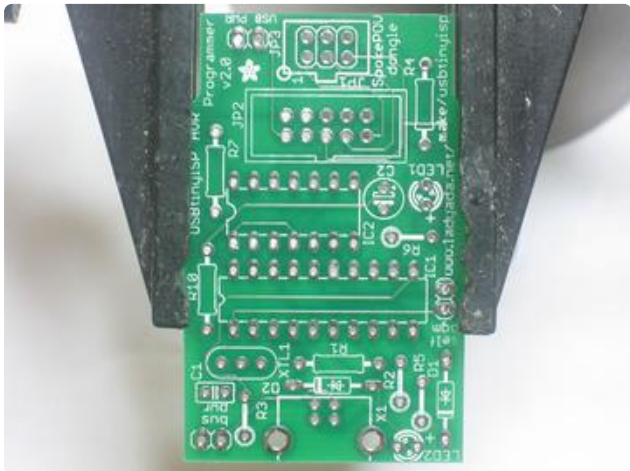
Ouch, hot!

The first step is to solder the kit together. If you've never soldered before, check the [P reparation page for tutorials and more](#) ().

Check the kit to verify you have all the parts necessary, [read the parts page](#) () for a list of parts you should have in your kit.

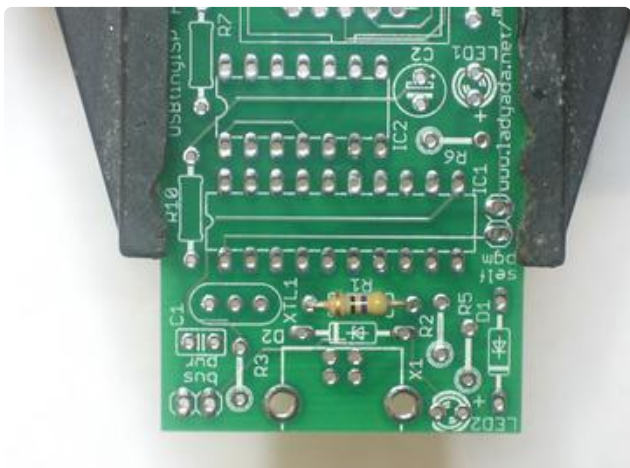


Get your tools ready! A board vise, soldering iron & solder, diagonal cutters, and a solder sucker (desoldering tool) if you have one.

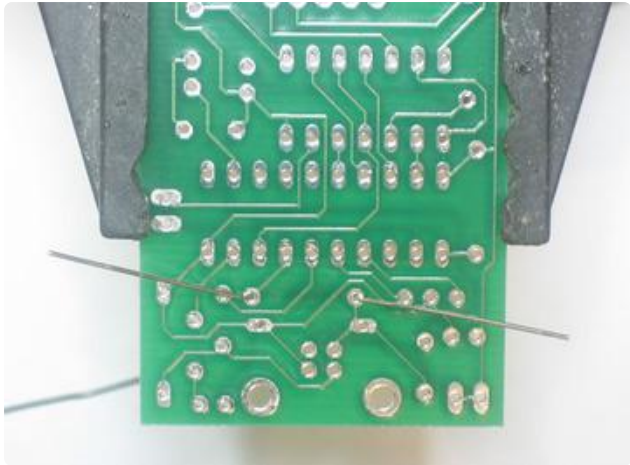


Next, get ready by placing the PCB in a vise so that you can easily place and solder the parts in!

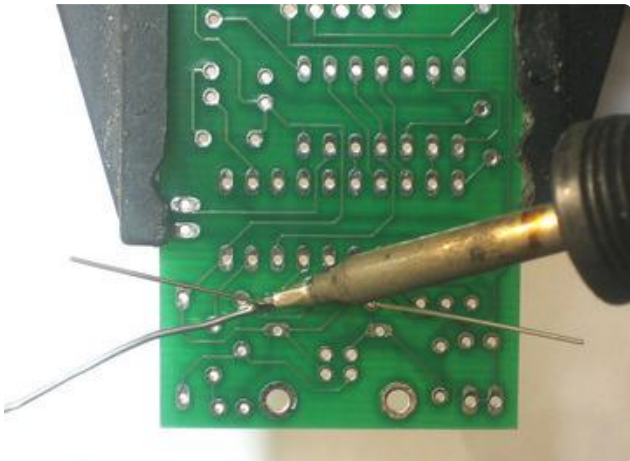
Check also that you have all the tools you need, & warm up your soldering iron to 650-700degF.



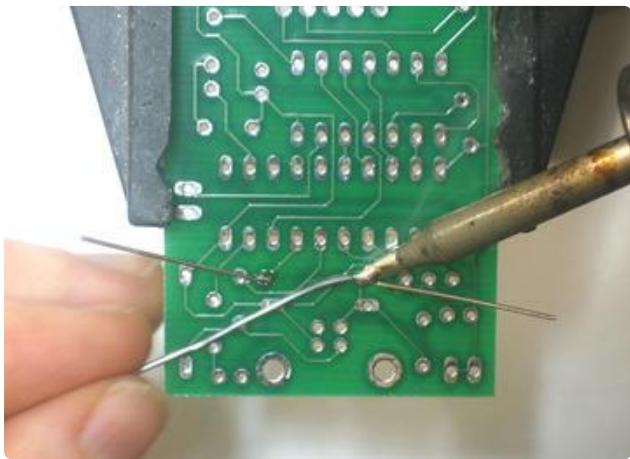
The first part to solder is a resistor, R1. This resistor has a 47ohm value, check the parts list to make sure you have the right one. Bend the two legs of the resistor so that its staple-shaped. Then slide the resistor into the PCB so that the outline matches the image on the silkscreen. Resistors are bi-directional so you don't have to worry about putting it in the wrong way.



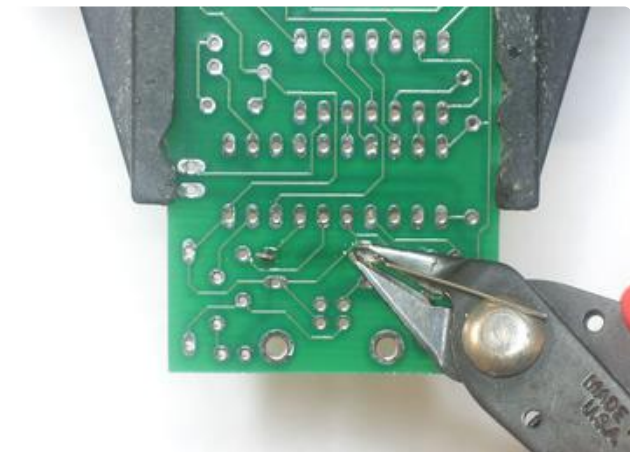
Bend the wire legs so that when the board is flipped over, it won't fall out.



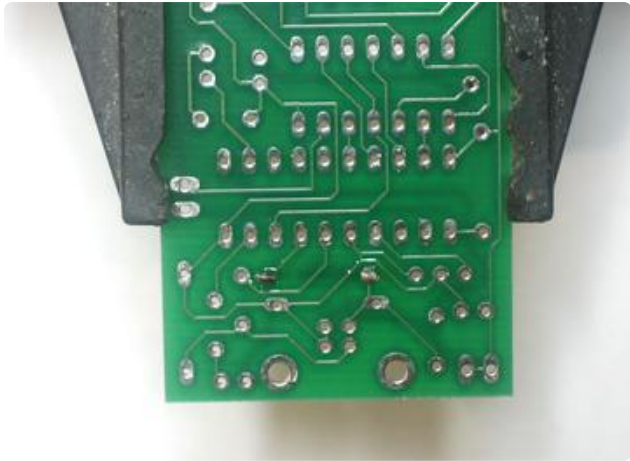
Next, with your soldering iron, solder each of the resistor legs. Place the tip of the iron against both the pad (ring) and lead (leg) and after a few counts, touch the solder in, to make a nice joint.



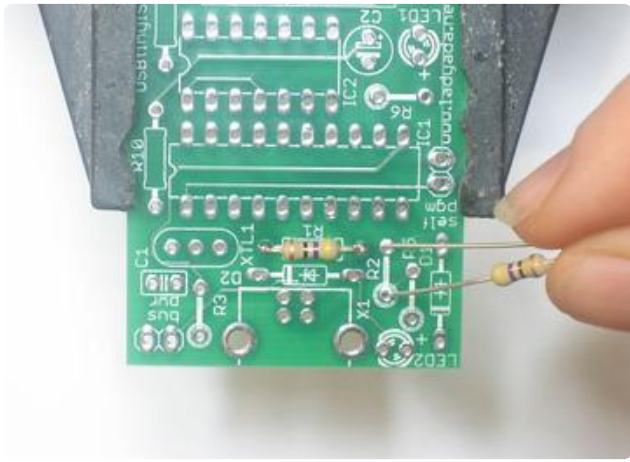
Repeat for the other joint.



Next, clip the excess leads using the diagonal cutters. Clip right above the top of the solder joint.

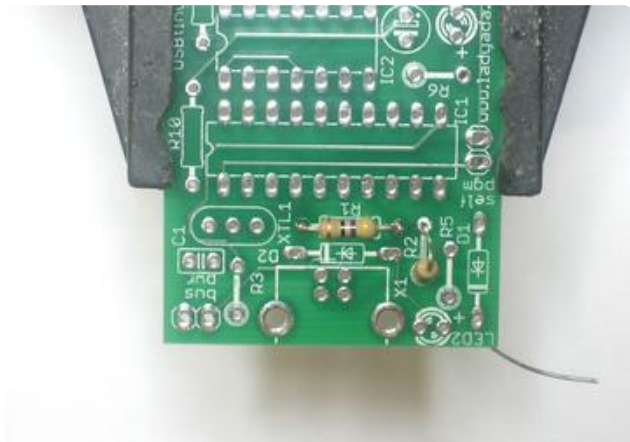


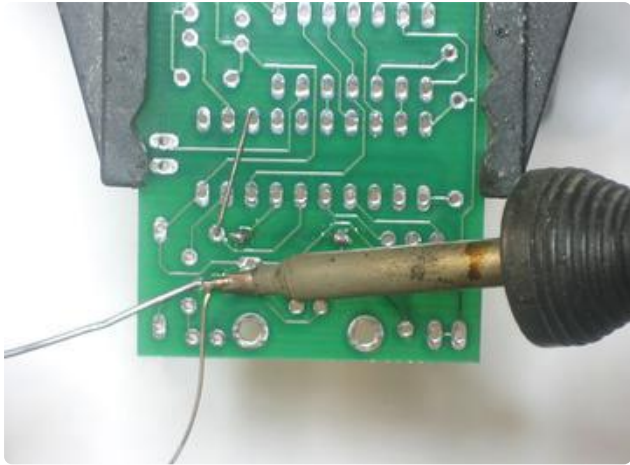
When you are done, it should look like this. If you have some sticky stuff on the solder joints, that's OK, that's the rosin inside the solder that protects the joints from oxidation. It isn't necessary to clean it off.



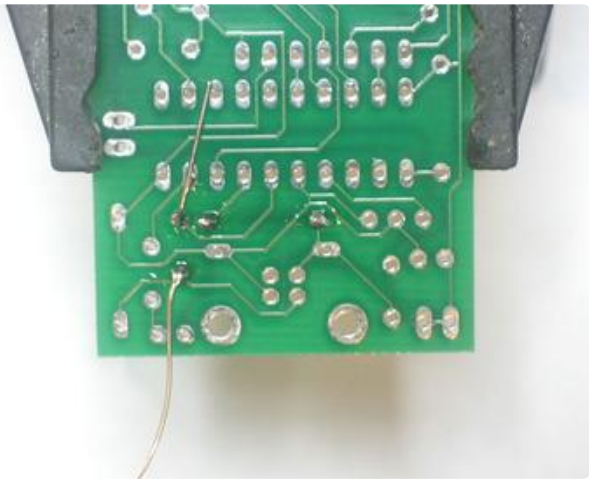
Next is the other 47 ohm resistor, R2

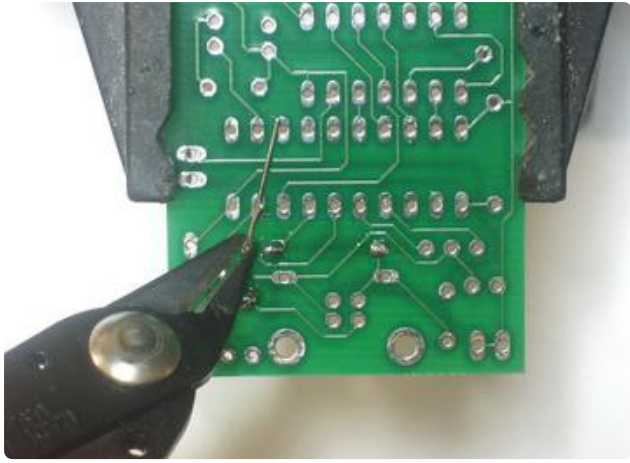
This one doesn't sit flat like R1, so bend it over as shown. Again it doesn't matter which end goes where since resistors work both ways.



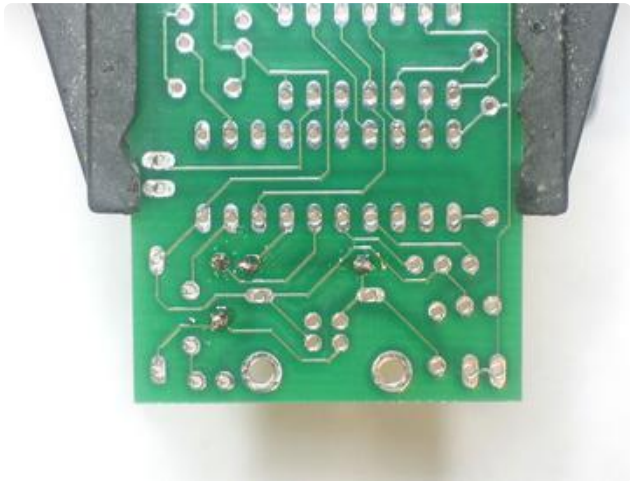


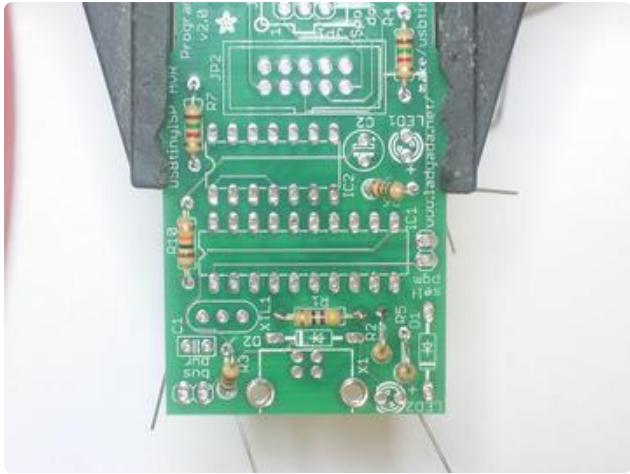
Solder the resistor just like you did with R1.





Then clip the excess wire off.



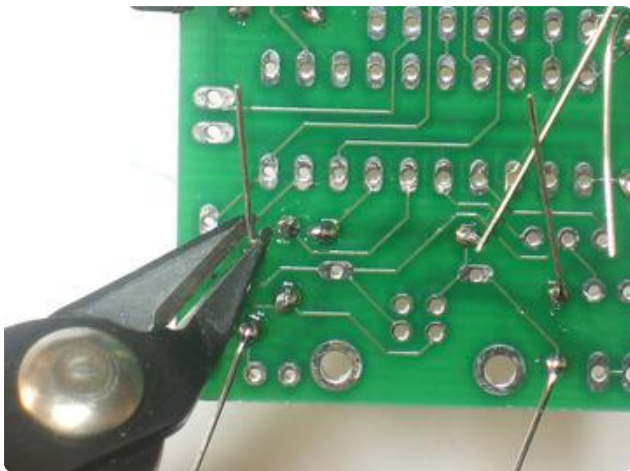
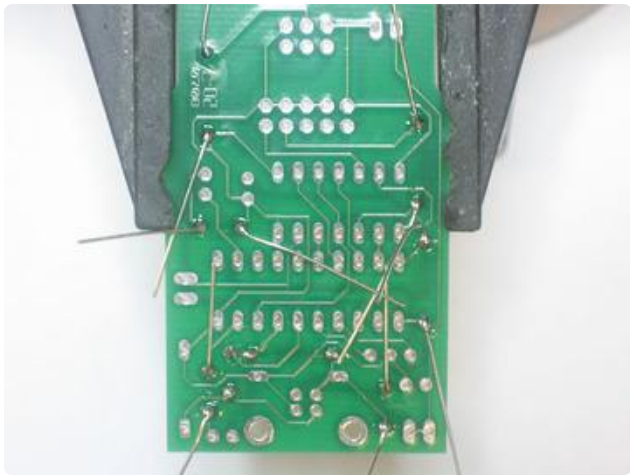
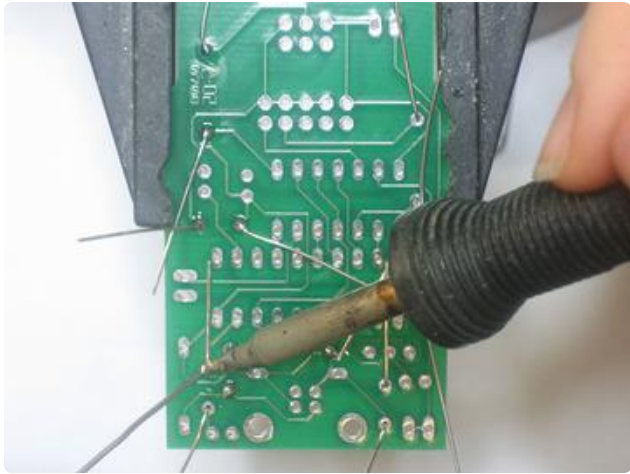


Now that you have a lot of practice with resistors, you can do the remaining 5 all at once. Place R10 (10K pullup resistor), R3, R5 and R6 (1.5K resistors for the USB connection, LEDs and output buffer).

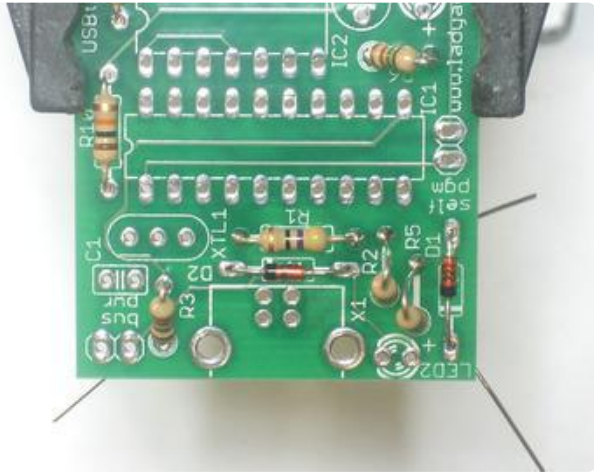
If you are using the UsbtinyISP with a SpokePOV kit, install R4 and R7 (1.5K) as well. If not you may want to switch these resistors for jumpers (see the second photo for a 'finished' shot) as it will mean that target boards with loaded pins can be programmed.

Note: sometimes the 74AHC125 is a bit larger than the silkscreen so you may want to put R7 in later, once the chip is in place.





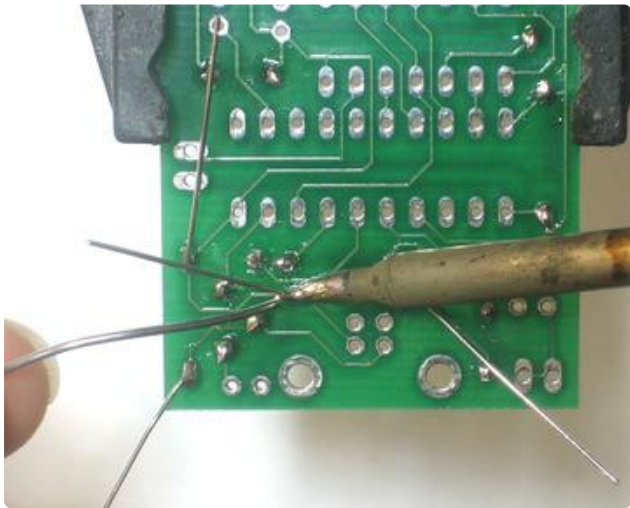
Solder and clip all of the leads.



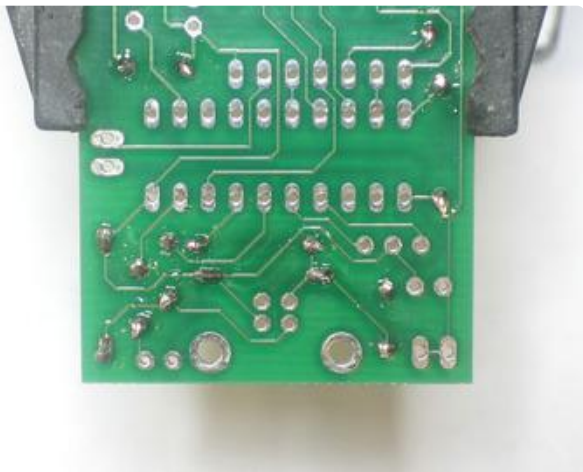
Next are the two 3.6V zener diodes, D1 and D2.

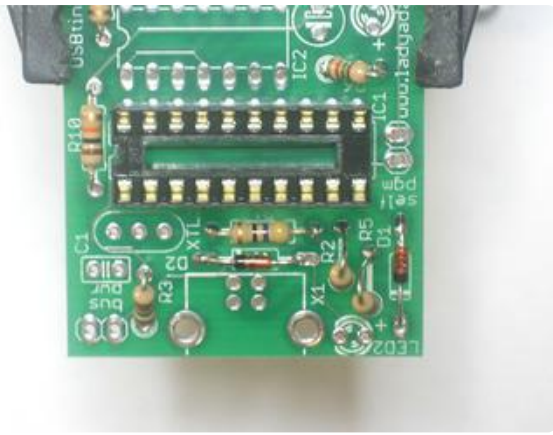
These diodes help convert the voltage from the microcontroller down to 3.3V, safe for the USB connection.

Diodes, unlike resistors, have to be placed a certain way or they won't work at all. Each diode has a small black line at one end. Make sure that this end matches with the white line on the silkscreen image. (See left)

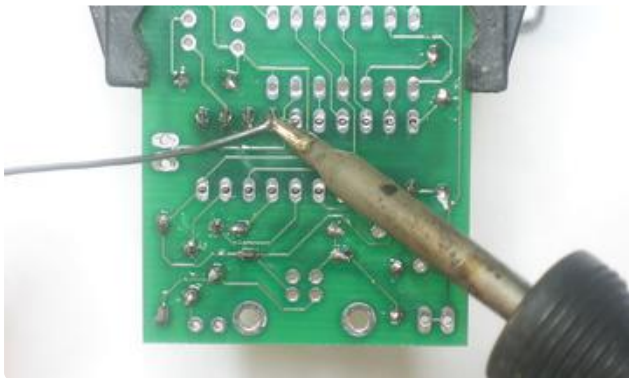


Solder and clip the diode leads.

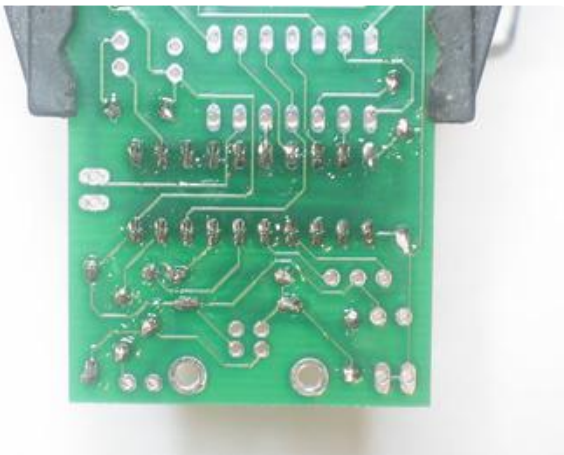




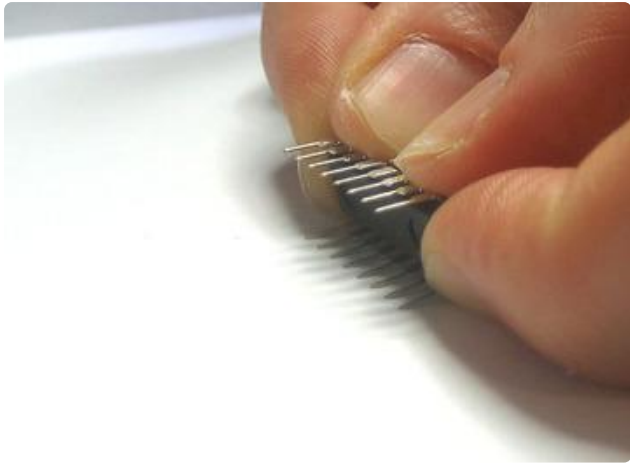
Next is the socket for the microcontroller that does all the hard work. A socket is useful because you can replace the chip in case of upgrade or damage. Sockets have a little notch in them to indicate which way to put in the chip. This notch should match the notch in the silkscreen image, in this picture, the notch is on the left-hand side.



Tack two opposite corners of the socket, to keep it in place, and then solder all the pins.



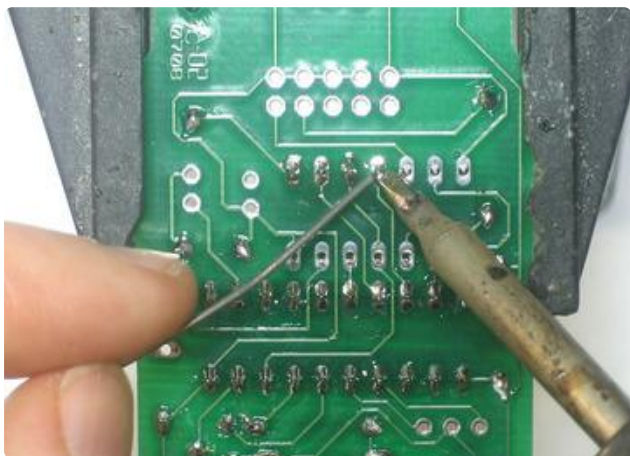
No clipping is needed as the socket pins are quite short already.



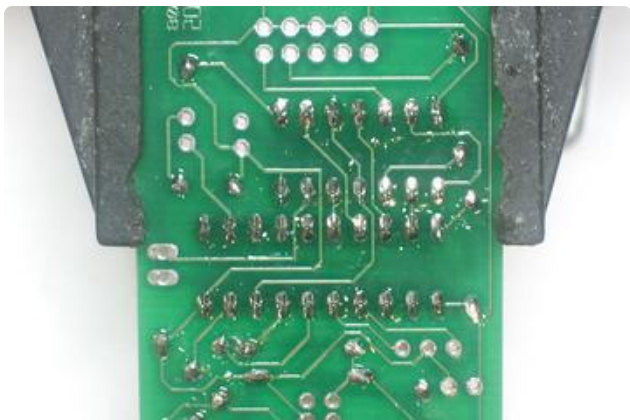
Next to be placed is the 74AHC125 buffer. This chip does a level-shifting conversion on signals from the USBtiny microcontroller to the device being programmed. This way you can safely program chips that vary from 1.8V to 5.5V voltage.

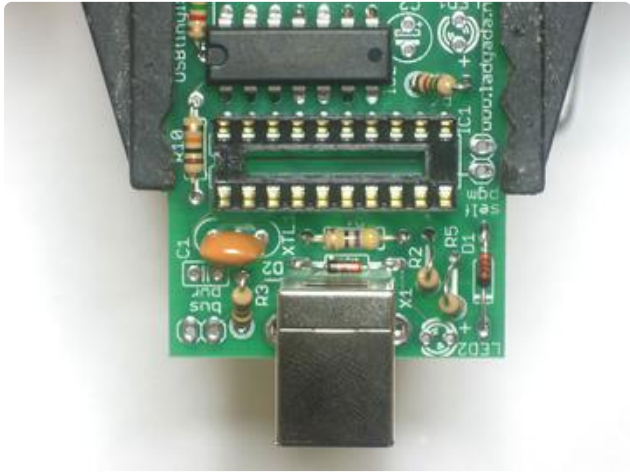
Integrated circuits must be placed correctly, check that the notch in the end of the chip matches the notch in the silkscreen image.

When ICs come from the factory, the legs are angled out somewhat which makes it difficult to insert them into the PCB. Prepare them for soldering by gently bending the legs against a flat tabletop so that they are perfectly straight.



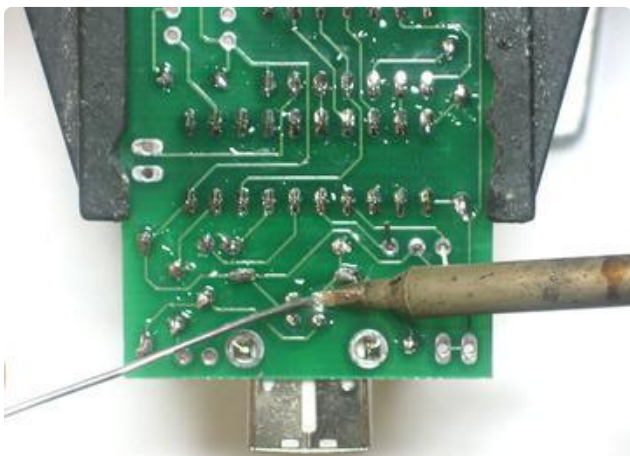
Solder each pin of the buffer, you won't need to clip the leads as they are quite short already.



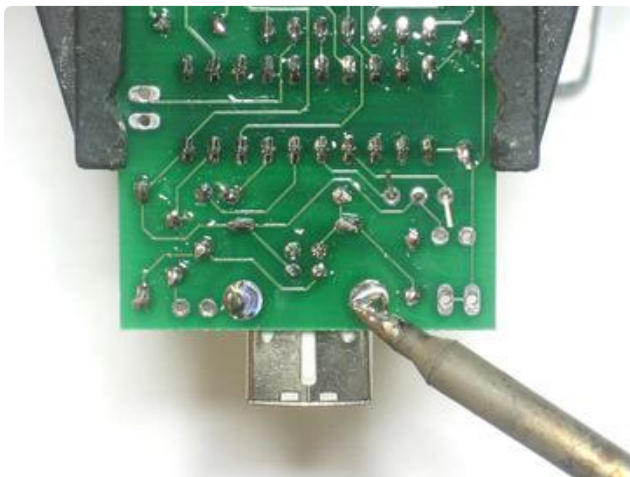


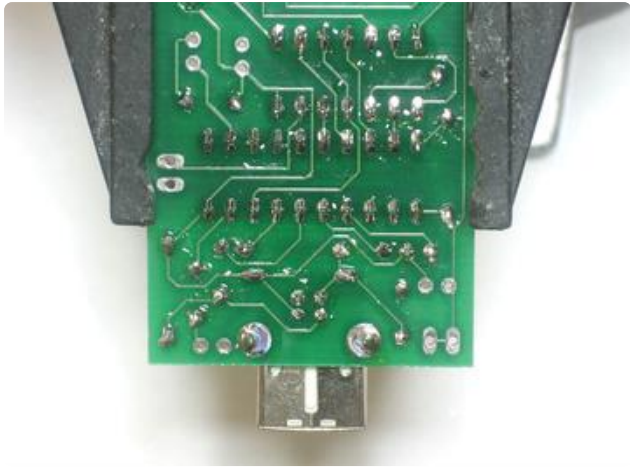
Next is the USB connector and the 12.00MHz ceramic oscillator. The USB connector is what we use to plug into the computer, the oscillator makes sure the USBtiny microcontroller runs at the precise rate necessary to communicate at the very picky USB protocol rates.

The oscillator can go in 'either way', they're made to be symmetric. The USB connector should snap in easily.

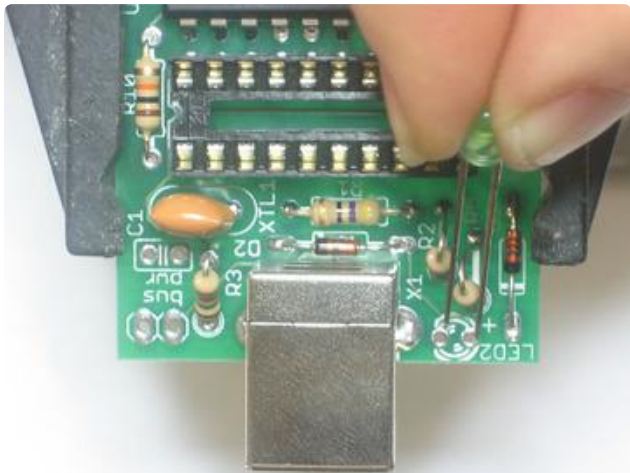


Solder in the three-pins of the oscillator and all 6 pins of the USB connector. Make sure to not bridge any of the square-pins and put plenty of solder on the mechanical tabs. These provide the resistance when you plug in a cable so it's important that they are soldered well, as shown here.

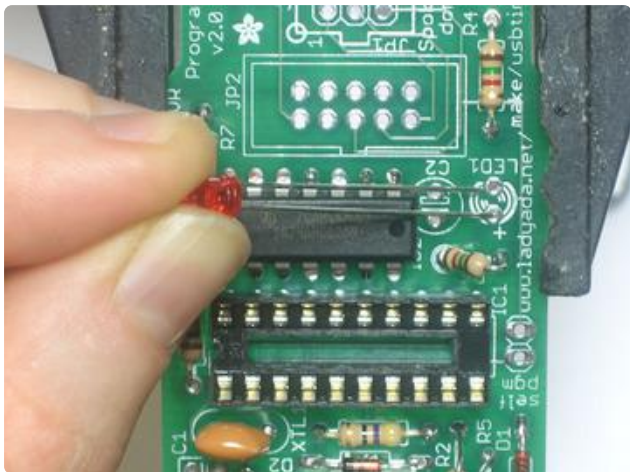




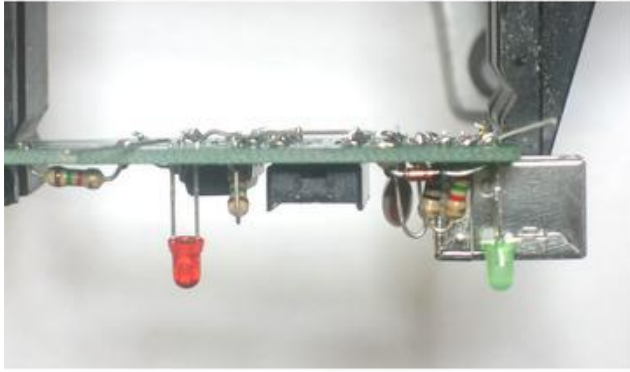
Although they are short, you should clip the pins to the oscillator if possible, to make sure they don't bend over and touch another component.



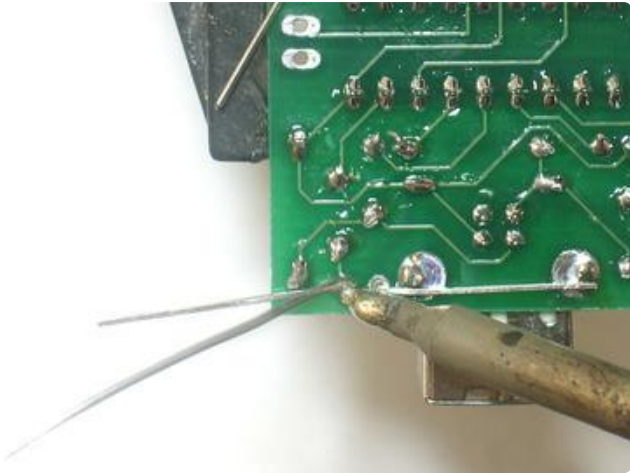
Next are the two indicator LEDs, green LED2 and red LED1. These LEDs let you know that the USB device connected successfully, and is in the process of programming the target device.



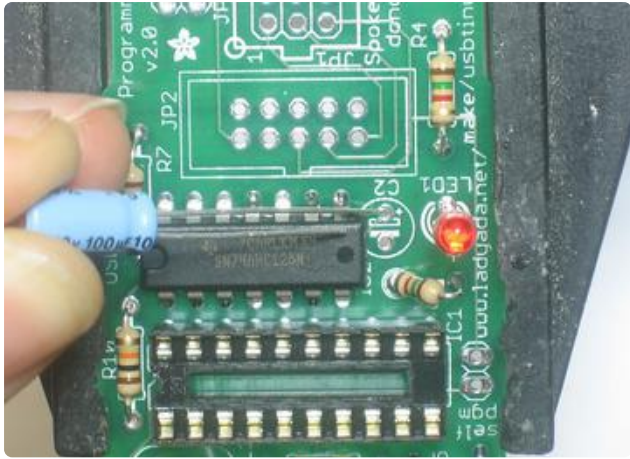
LEDs are diodes, and must be placed correctly or they won't light up, which is very confusing. Make sure the longer (positive) lead of the LED goes into the hole marked with a +. See the images to the left.



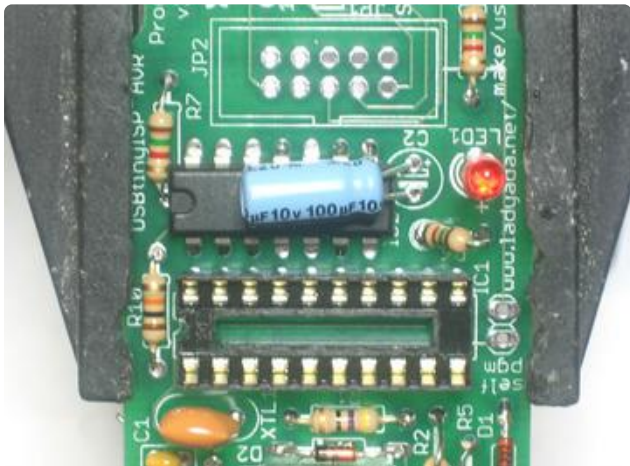
The LEDs are also supposed to be very close to the top of the enclosure, so that you can see the light through the drilled holes, when you bend the leads, make sure the LEDs stick out about 1/2" above the PCB.



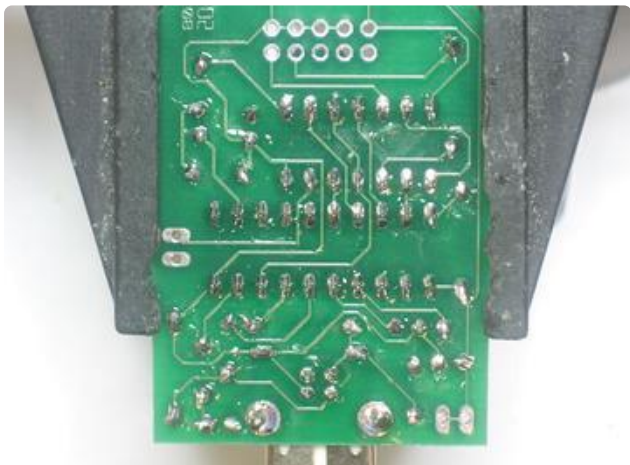
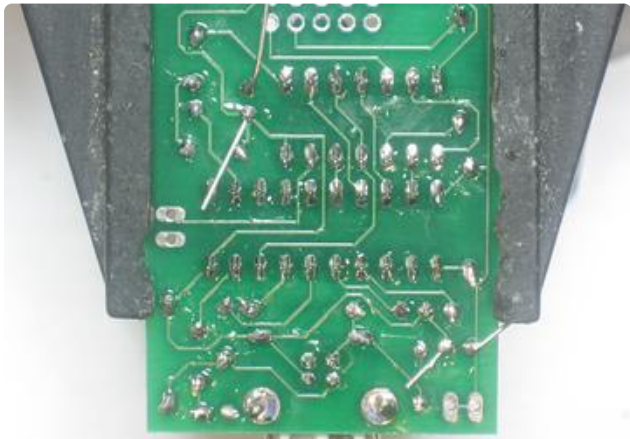
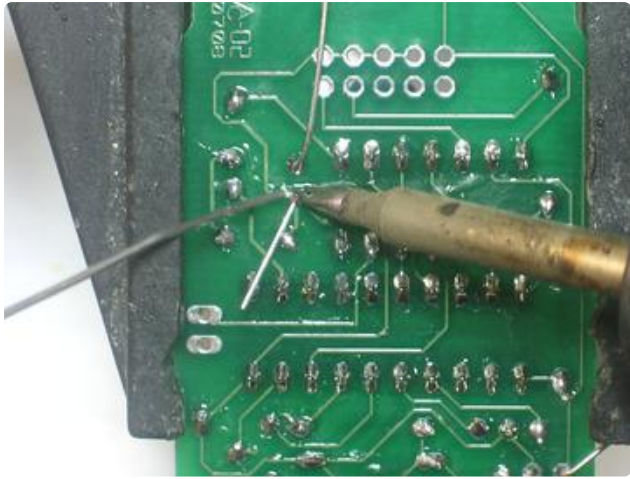
Solder and clip the two LEDs.



Next are the two capacitors, C1 and C2. These provide some power supply filtering so that the USBtinyISP is less flakey. C1 goes in the corner next to the USB connector. It is a non-polar ceramic capacitor so it can go in either way.



C2 is a polarized electrolytic. It must go in only one way. Make sure the longer leg of the capacitor goes into the hole with a +. Bend the capacitor so that it lies on top of the buffer chip.



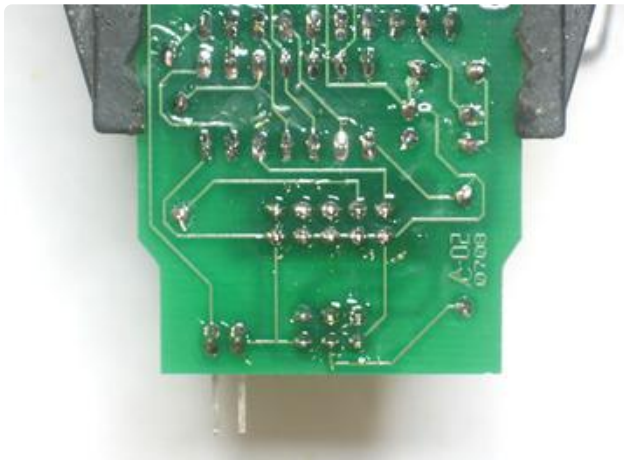
Solder and clip both capacitor's leads.



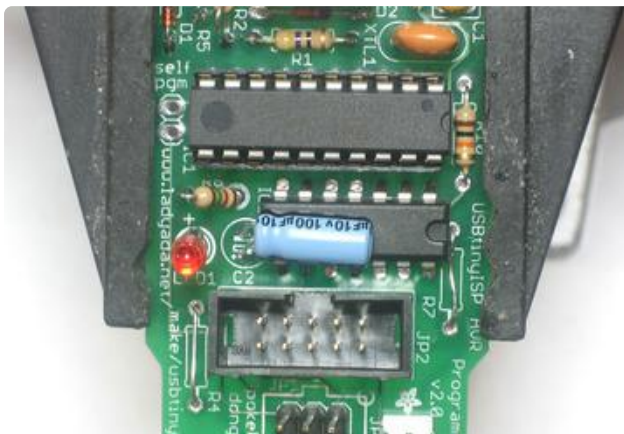
Almost done! The last set of parts are the headers for the cables, and the jumper header. The 10-pin box header has a notch in it, make sure it matches up with the silkscreen, as shown.

The 6-pin header goes in with the long pins sticking up.

The 2 pin jumper has the long pins pointing out.

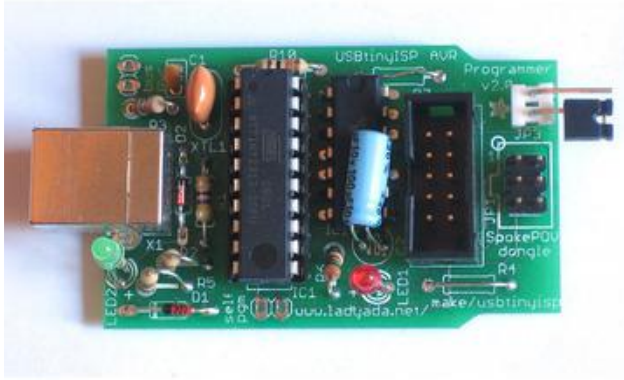


Solder in all the pins of the headers. You won't need to clip them as they are already quite short.



Finally, straighten the pins of the microcontroller and place it, so that the notch in the chip matches the notch in the socket (and the silkscreen) as shown.

Now go on to make the cables and put the PCB in the case.

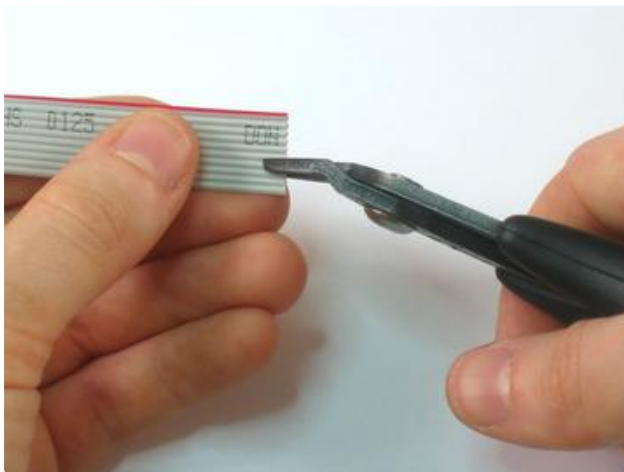


If you're not using the USBtiny to talk to a SpokePOV kit, and if you are using a target that has some load on the SCK and MOSI pins, you may need to replace R4 and R7 with jumpers as the 1.5K resistors will have trouble driving the load!

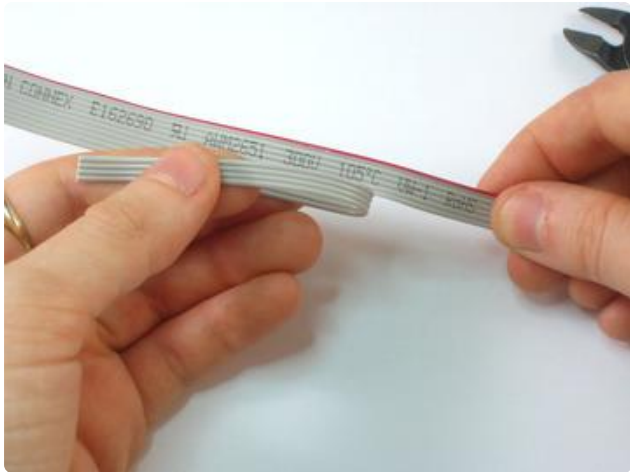
Make 6-pin cable (Old kits, without pre-made 6-pin cables)

There are two standards for AVR programming, 6-pin and the 10-pin headers. Therefore, it's important that an AVR programmer have both types of cables. The 10-pin cables are easy to come by, but the 6-pin ones must be custom made. However, making a cable is super easy, just follow these steps!

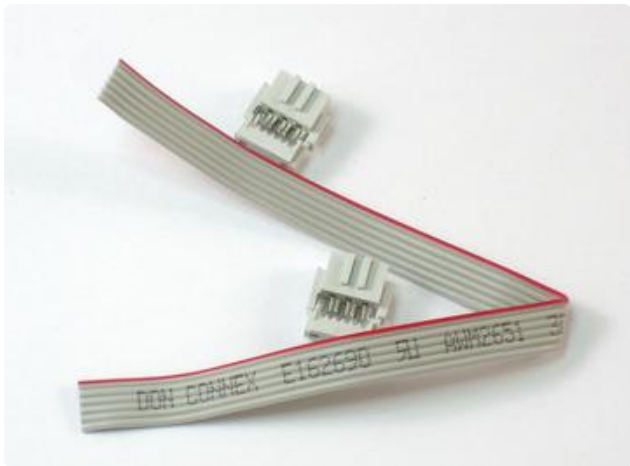
If you're using the adaptor for a spokepov or don't need the 6pin cable, you can just skip this part.



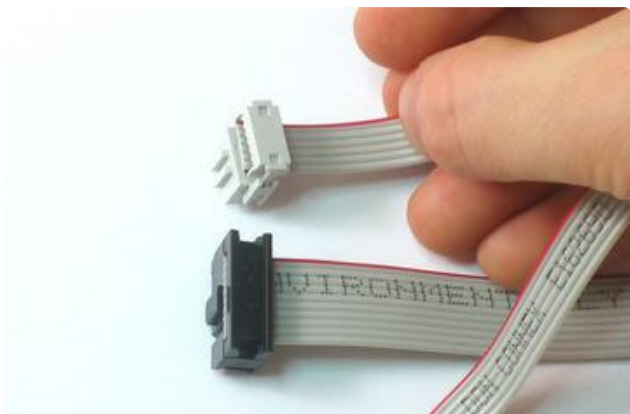
It's hard to find 6-conductor ribbon cable so you may end up with 10-conductor wire. (The kit ships with 6-conductor) If so, just use your diagonal cutters (or a knife) to cut a notch so that the red stripe is on the 6-conductor side.



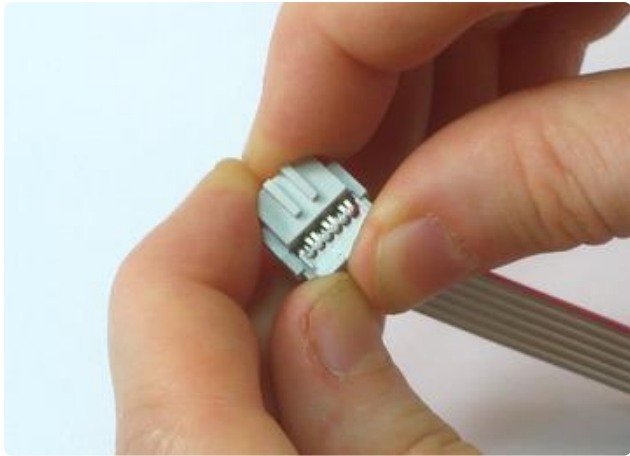
Tear the cable, it should come apart cleanly.



You are now ready to assemble the cable.



It's important that the key (the bump in the connector) and the red stripe line up. Match the image on the left, just poke the conductor in with a mm or two past the edge.

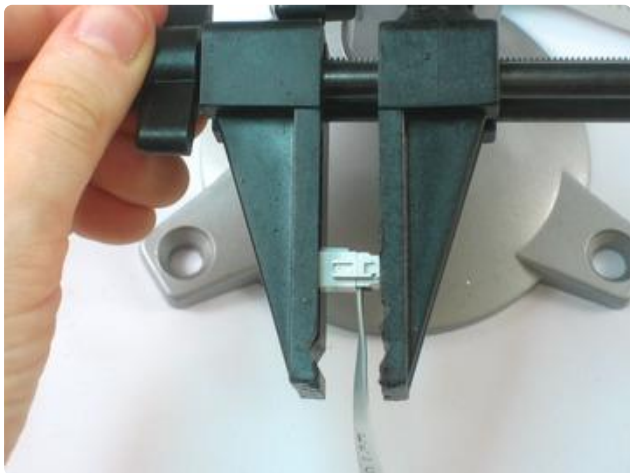


Get it started by just squeezing it with your fingers to make sure the wires are aligned properly. You won't be able to finish the cable this way so dont try!

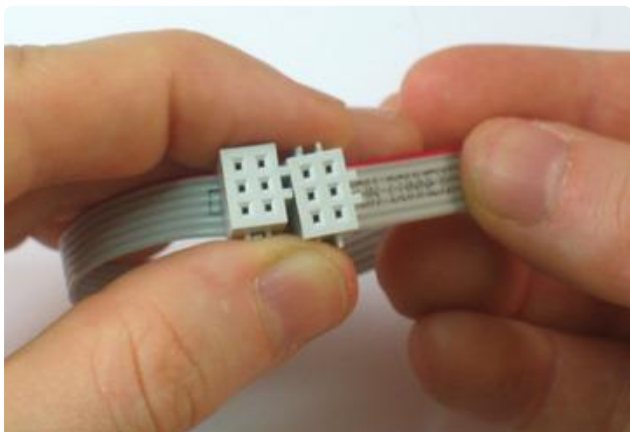


Do not use needlenose pliers to try to press the pieces together. You have to have very flat pressure from both sides.

For example, use the flat side of a tool to press against a table top.



Or better yet, a vice! Slowly squeeze the two sides together until they lock.



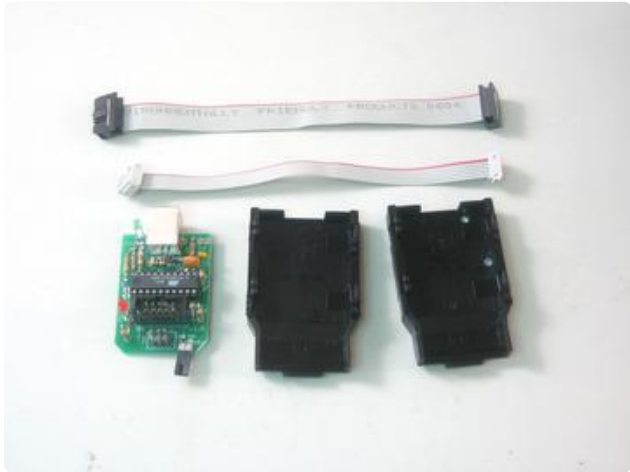
Do the other end, keeping track of the key and red line.



Yay! You've got two cables!

Case

Finally it's time to put the programmer in the case for use.



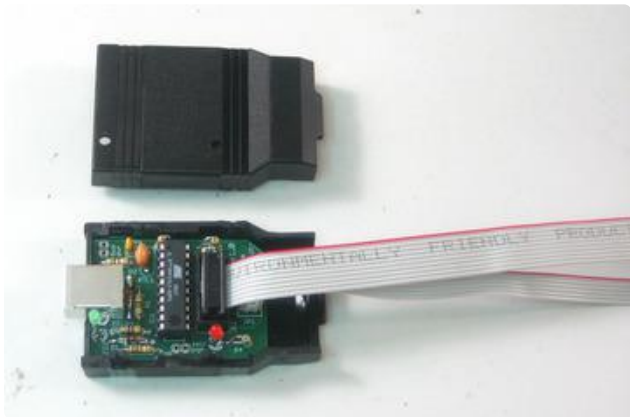
Take the PCB, two case halves and the cables you've made.



Plug in the two cables as shown, the red stripes on top and so that the cables don't bend over the plug (the case won't fit).

Put the PCB into the bottom case half.

The 6-pin cable may have strain reliefs that can clip on. You don't really need them but if you do want strain relief, put it on the one that goes to the target: the cable won't fit in the case if the strain-relief bit is on.



Line up the LEDs and snap the top on. You're done!



Next up, [read the usage manual \(\)](#).





Can't get it working? Don't worry, help is available in the [forums \(\)](#)!

Parts (v1.0)

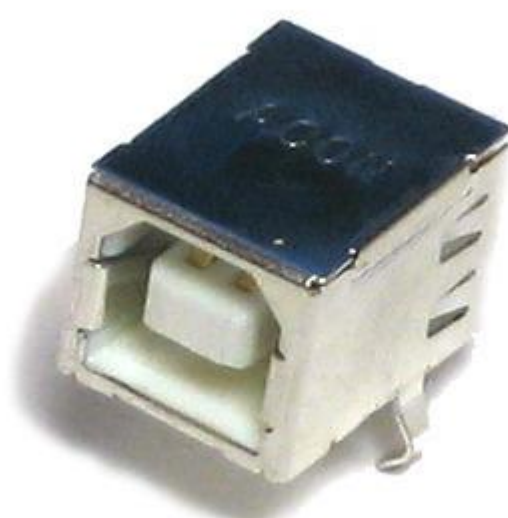
Kit parts list

This is the list for very old kits, its very unlikely you have a v1.0 but we're keeping it around for historical record

Image	Name	Description	Distributor
 A black 8-pin DIP microcontroller chip. The top surface is printed with the ATMEL logo, the part number "0441", and "ATTINY2313-20P1". A small triangle on the left side indicates the pin 1 orientation.		ATMEL 0441 ATTINY2313-20P1 Pre-programmed when purchased in a kit)	ATMNY2313-20PU Digikey & Mouser
 A small, orange, rectangular ceramic oscillator component with two metal leads extending from the bottom.	XTL1	12.00 MHz ceramic oscillator	ZTT-12.00MT

	C1	Bypass 104 capacitor (0.1uF) Might be blue	BC1160CT-ND
	R10	10K 1/4W 5% resistor (brown, black orange gold)	10KQBK-ND
 <p data-bbox="178 1765 539 1803">BrownGreenRedGold.gif</p>	R3, R4, R5, R6, R7	1.5K 1/4W 5% resistor (brown green red gold)	1.5KQBK-ND
	R1, R2	27-68 ohm 1/4W 5% resistor	47QBK-ND

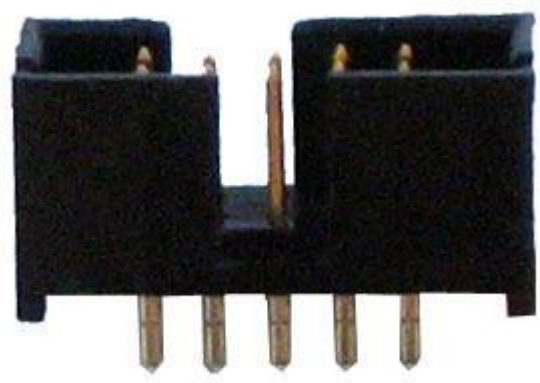




X1

USB type B
male jack

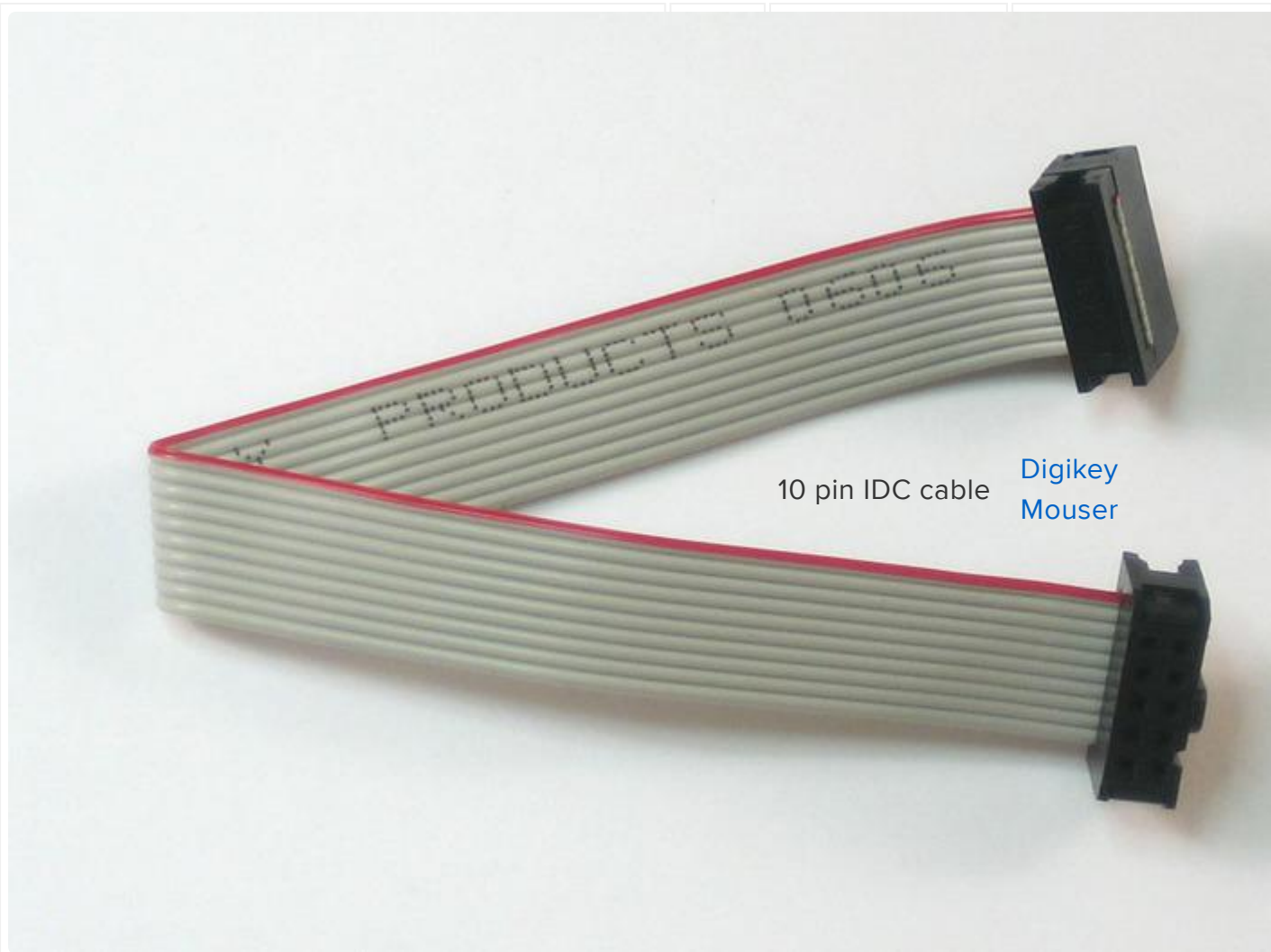
61729-0010BLF



JP2

10 pin box
header

	JP1	6 pin straight header	Molex 10-89-7062
	JP3	2 pin right angle header	Mouser (640453-2)
	JP3'	Jumper/Shunt	Mouser (71363-102LF)



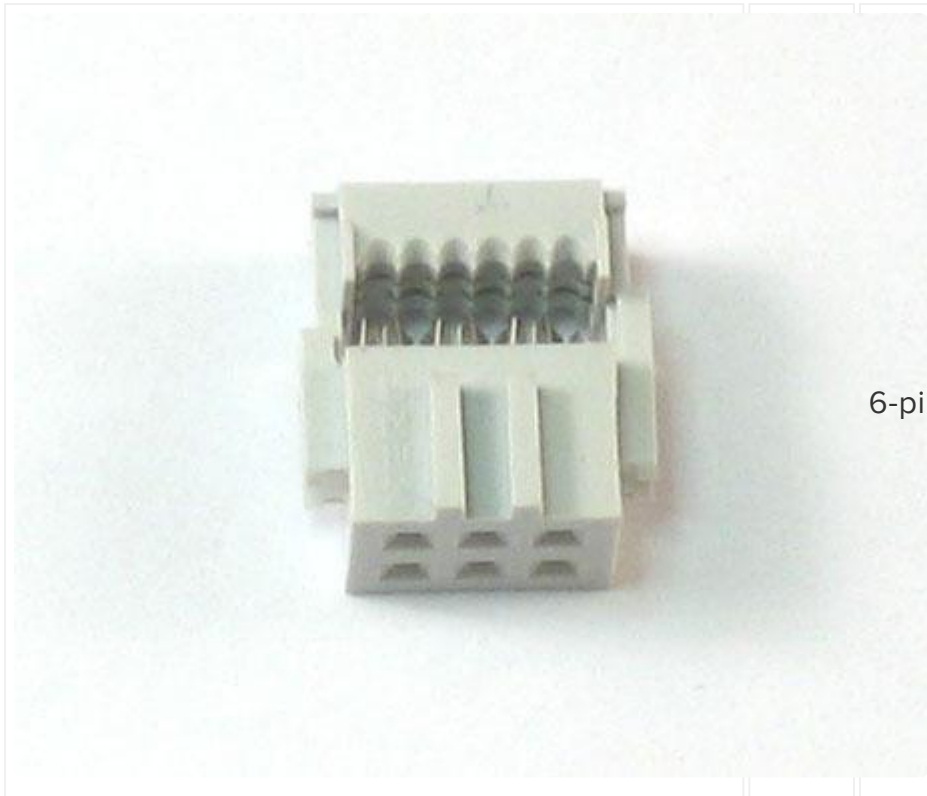
10 pin IDC cable

[Digikey](#)
[Mouser](#)



6" wire ribbon
cable (6
conductors)

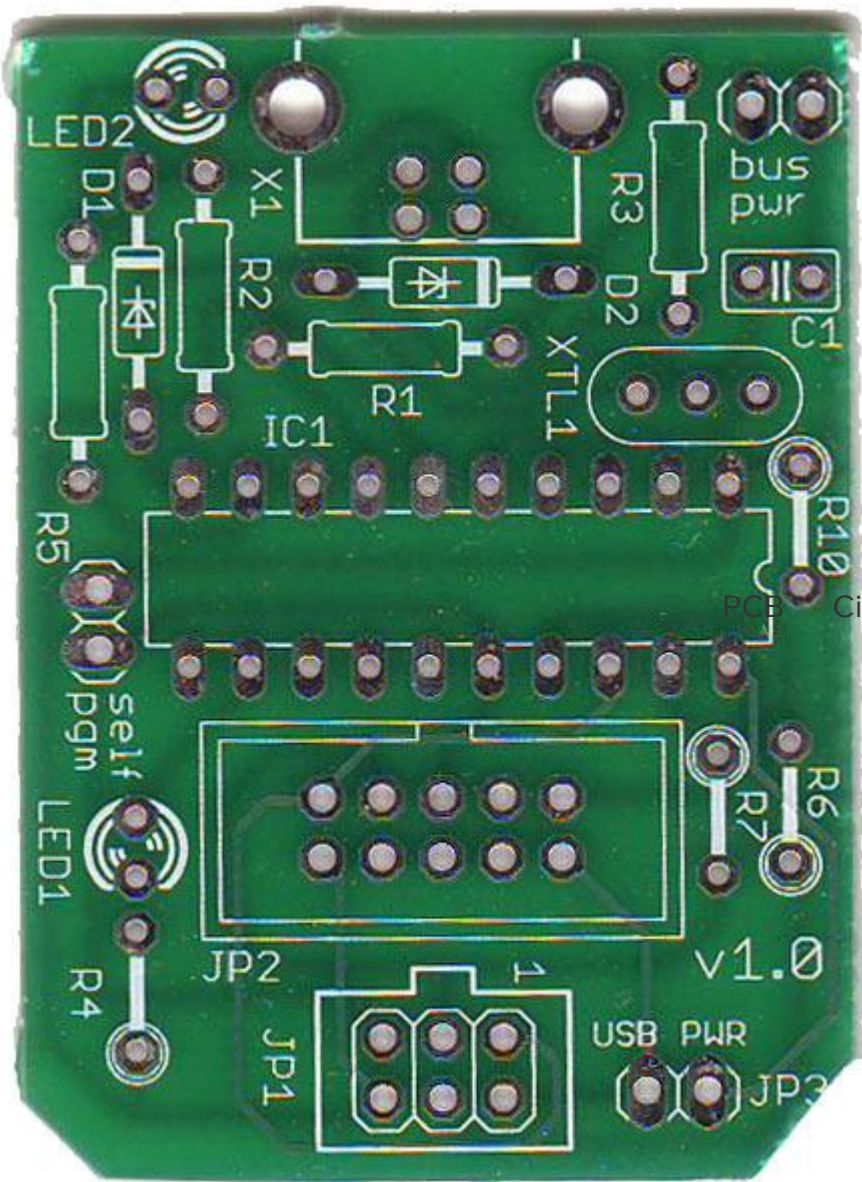
[Digikey](#)



6-pin IDC plug

FCI 71600-006LF

[Mouser](#)



PCB Circuit board

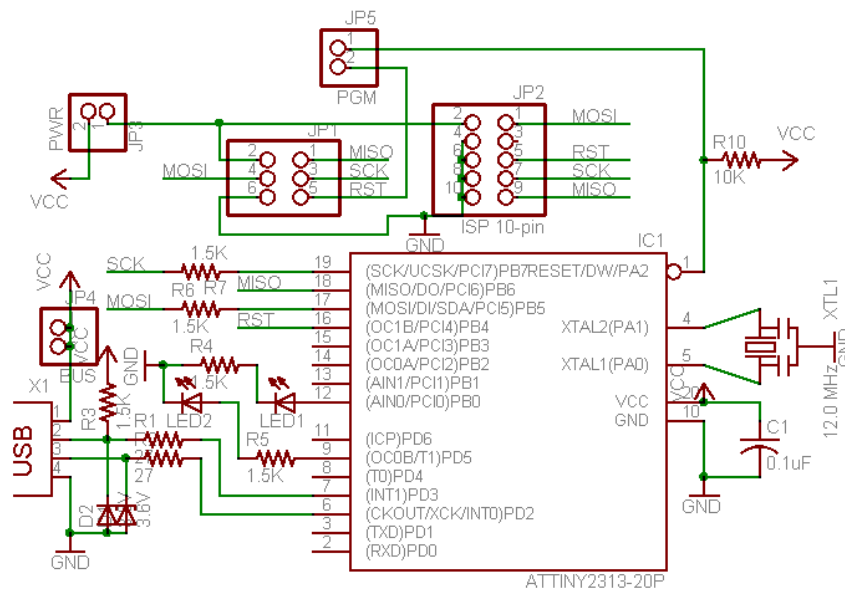
Adafruit Industries



Part no CNS-0407

Enclosure

Mouser
Pactec



Solder (v1.0)

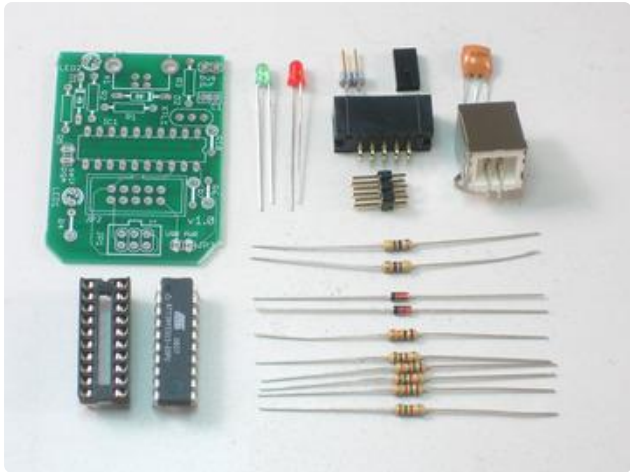
V1.0 instructions!

These are instructions for v1.0 USBtinyISP. If your PCB looks a bit different, [you probably have a v2.0 and should go here for the instructions \(\)](#).

Its very unlikely you have a v1.0 but we're keeping it around for historical record

Solder it!

The first step is to solder the kit together. If you've never soldered before, check the [P reparation page for tutorials and more \(\)](#).



Check the kit to verify you have all the parts necessary (the 0.1uF capacitor is missing, oops!).



Get your tools ready! A board vise, soldering iron & solder, diagonal cutters, and a solder sucker (desoldering tool) if you have one.



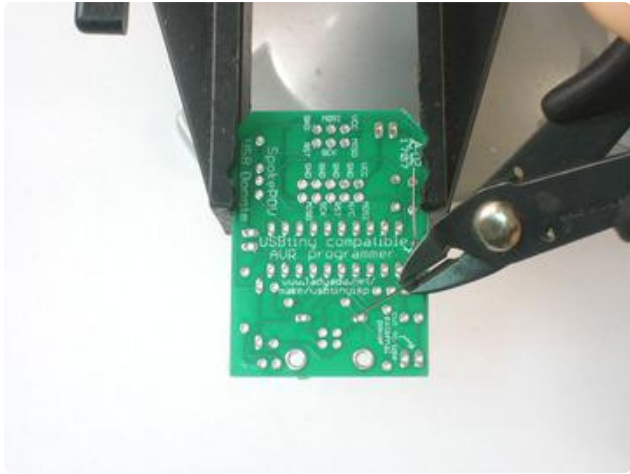
Put the PCB in the vise & heat up the soldering iron so you are ready to go!



Place the first component: the 10K resistor as shown. Resistors are non-directional, so you can put them in 'either' way and they'll work fine. When you put the legs through the PCB, bend them out so when the PCB is flipped it won't fall out.

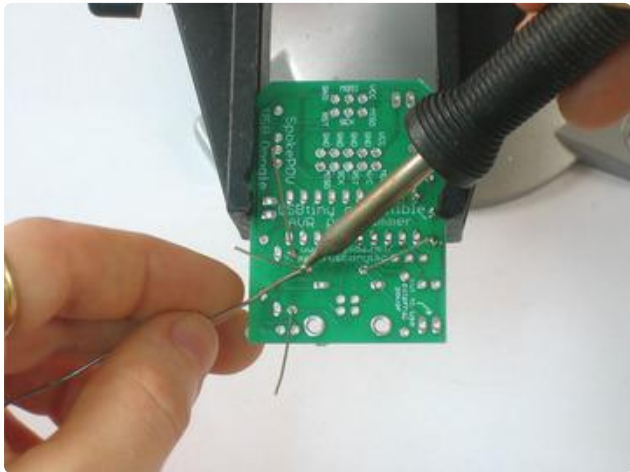


Flip the PCB over.

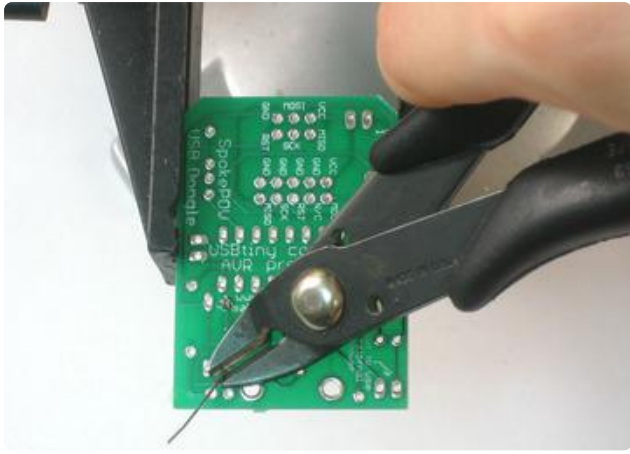


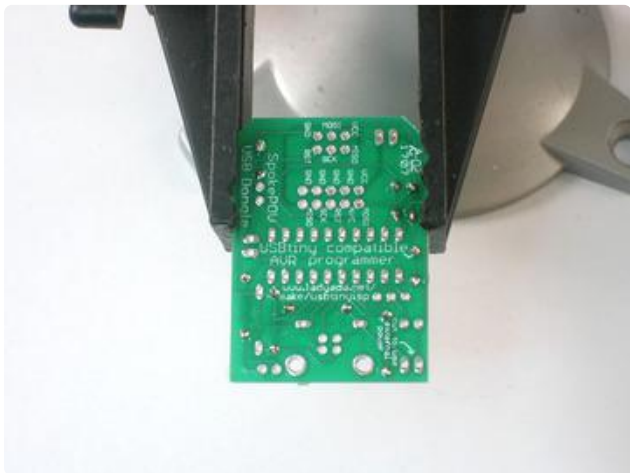
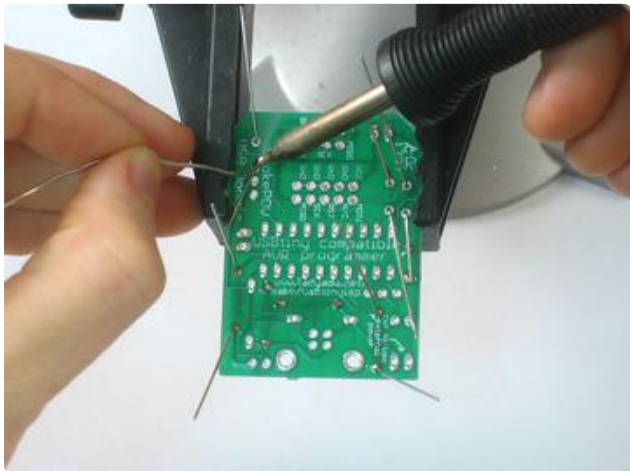
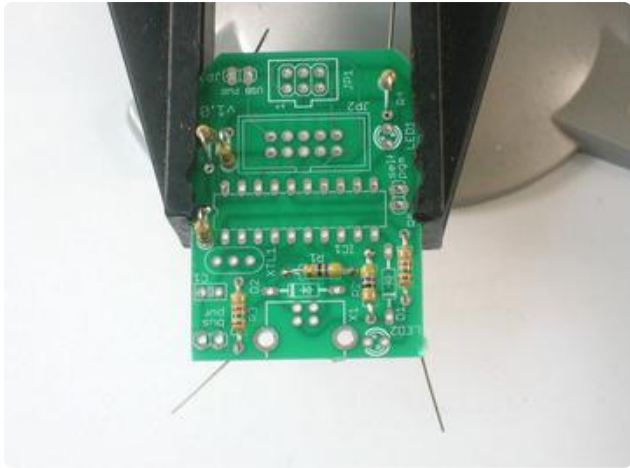
Use the diagonal cutters to clip the legs off just above the solder joint.



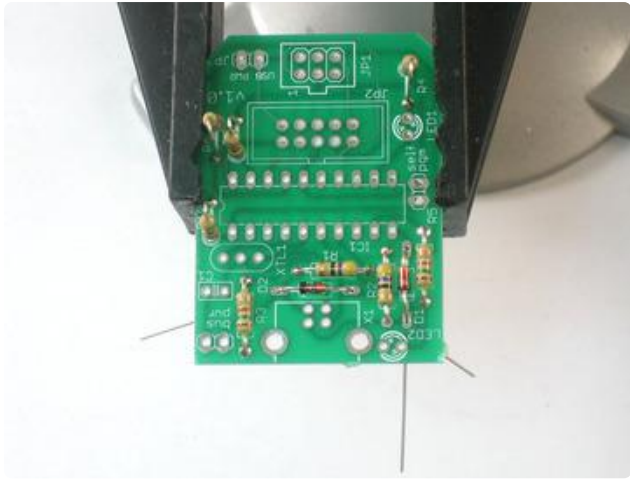


Next up are the 2 47 ohm resistors. Place them as shown. Bend the legs out, flip the PCB, solder the 4 joints and clip them.

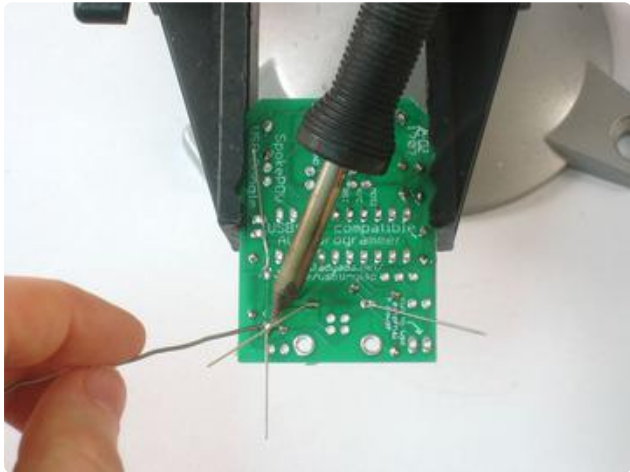




Next are the 5 1.5K resistors. Place them as shown, then solder them in and clip the leads off.



Next are the zener diodes. Diodes are directional so make sure to put them in as shown. There's a white stripe on the PCB drawing that matches the black stripe on the glass diode.





The next parts are the USB connector (big silver part), the 12.00MHz ceramic oscillator (the three pin part) and the ceramic capacitor (the small yellow part).



Both the capacitor and oscillator are nondirectional so they can go in either way. The USB connector can only go in one way, and snaps into place.



When you solder in the parts, make sure that you put plenty of solder on the two prongs that hold the USB connector in. These are mechanical connections: the solder actually acts as a 'glue' here, keeping the part fixed in place!



Next are the headers and the microcontroller socket. The 6-pin header has no direction so just put it in either way.

The 10-pin box header has a notch which should match the notch in the PCB graphic. (here it's closest to the microcontroller socket).



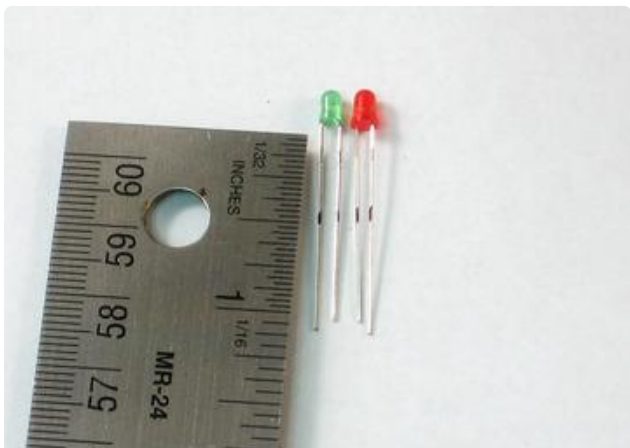
The right angle header JP3 should go in as shown, with the two prongs sticking out over the PCB.

The microcontroller socket also has to go so that the notch in the end matches the drawing. Here it's on the right. If you mess it up it's not the end of the world, just remember to put the microcontroller in the right way!



When you solder it in, it might be tough to keep the parts in place. you can try 'tacking' the parts in place by holding it in with a finger and soldering one or two corners as shown.

Then go back and solder each and every connection



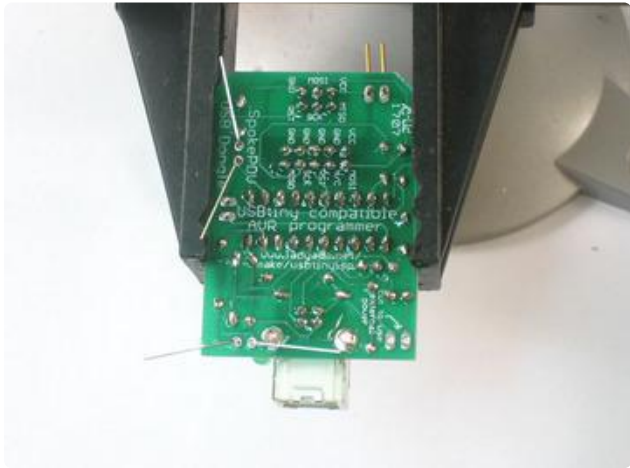
The two LEDs are next. They should stand-off a bit from the PCB so make a marking about 1/2" (1cm) from where the colored plastic ends.



Place the LEDs as shown, the red one near the 10-pin header and the green one near the USB plug.



LEDs are directional and if you put them in backwards they won't light up. To figure out which way is right, look on the PCB, at the image of the LED. One side of the image is slightly flattened. That indicates the negative side. The LEDs have one lead that is shorter than the other. The short lead is also negative.



In this image, the negative side for the green LED is on the left. The negative side for the red LED is towards the top.



Instead of making the LEDs sit flat against the PCB, bend the leads at the 1/2" mark you made so they stick out. Solder them in place.



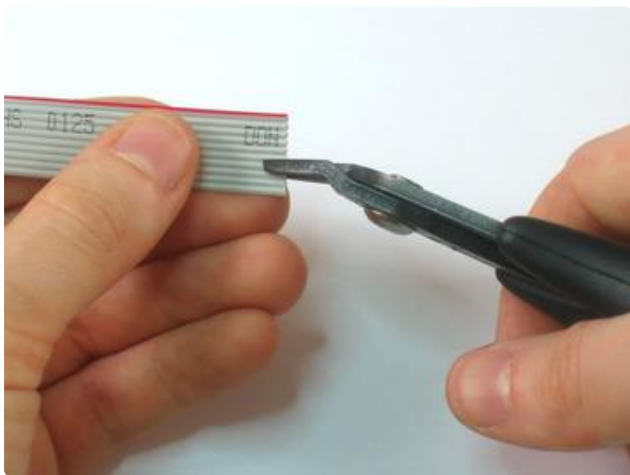
Soldering's done. Next up, insert the microcontroller. You can do this by gently bending the legs in using your fingertips or the tabletop.

Make sure it's inserted as shown, and press it in so it's seated well into the socket.

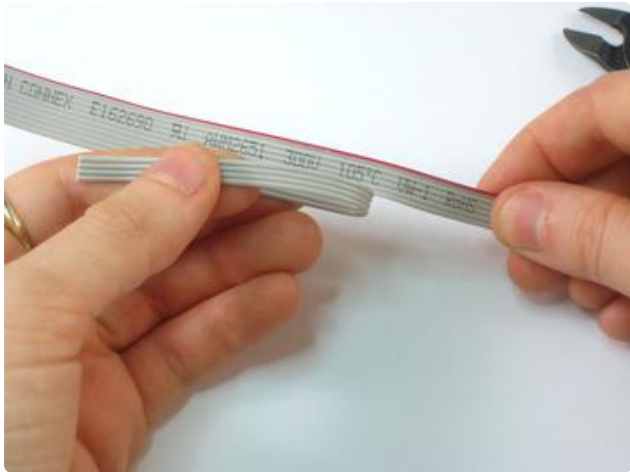
Make 6-pin cable

There are two standards for AVR programming, 6-pin and the 10-pin headers. Therefore, it's important that an AVR programmer have both types of cables. The 10-pin cables are easy to come by, but the 6-pin ones must be custom made. However, making a cable is super easy, just follow these steps!

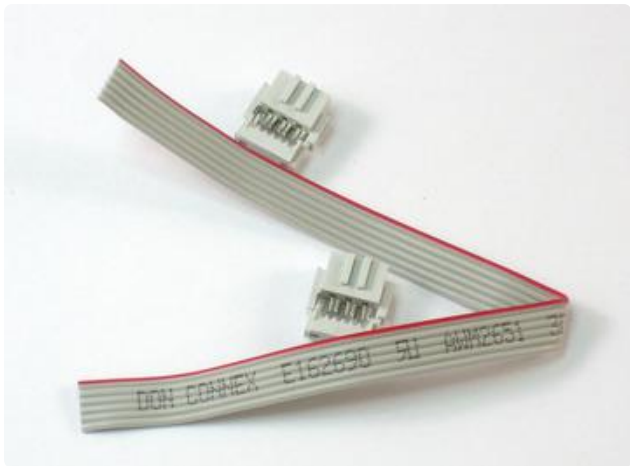
If you're using the adaptor for a spokepov or don't need the 6pin cable, you can just skip this part.



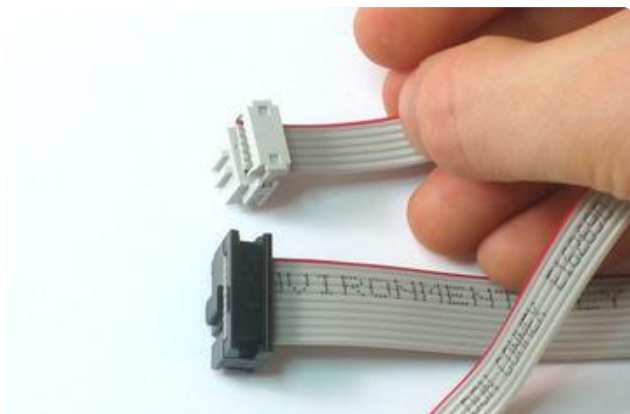
It's hard to find 6-conductor ribbon cable so you may end up with 10-conductor wire. (The kit ships with 6-conductor) If so, just use your diagonal cutters (or a knife) to cut a notch so that the red stripe is on the 6-conductor side.



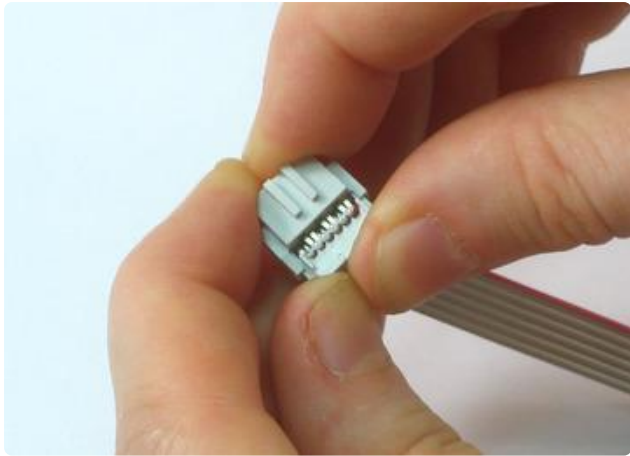
Tear the cable, it should come apart cleanly.



You are now ready to assemble the cable.



It's important that the key (the bump in the connector) and the red stripe line up. Match the image on the left, just poke the conductor in with a mm or two past the edge.

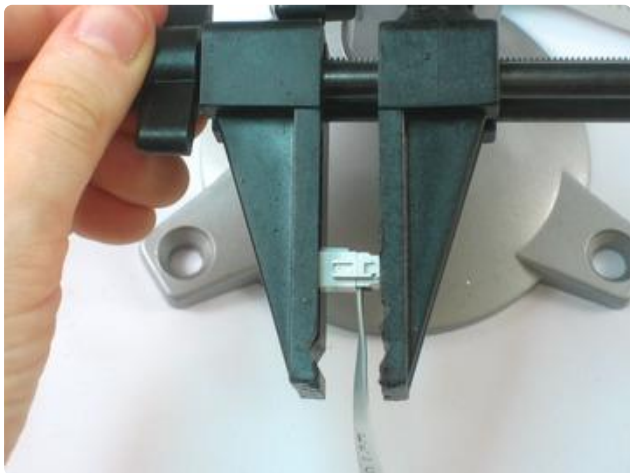


Get it started by just squeezing it with your fingers to make sure the wires are aligned properly. You won't be able to finish the cable this way so don't try!

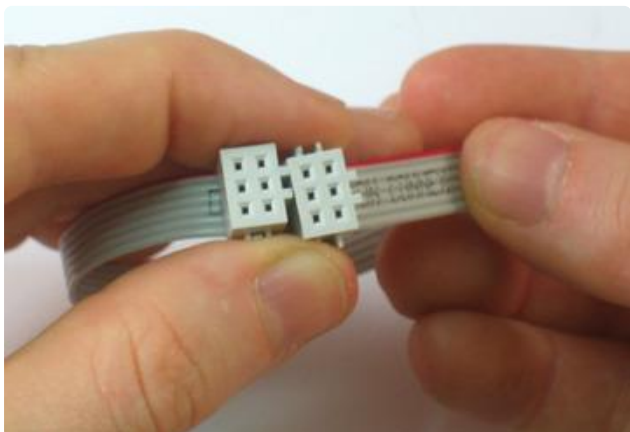


Do not use needlenose pliers to try to press the pieces together. You have to have very flat pressure from both sides.

For example, use the flat side of a tool to press against a table top.



Or better yet, a vice! Slowly squeeze the two sides together until they lock.



Do the other end, keeping track of the key and red line.



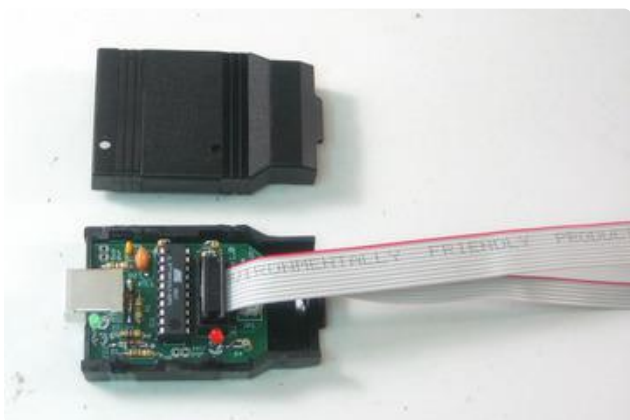
Yay! You've got two cables!

Case

Finally it's time to put the programmer in the case for use.



Take the PCB, two case halves and the cables you've made.



Plug in the two cables as shown, the red stripes on top and so that the cables don't bend over the plug (the case won't fit).

Put the PCB into the bottom case half.

The 6-pin cable may have strain reliefs that can clip on. You don't really need them but if you do want strain relief, put it on the one that goes to the target: the cable won't fit in the case if the strain-relief bit is on.



Line up the LEDs and snap the top on.
You're done!

Next up, [read the usage manual \(\)](#).

Can't get it working? Don't worry, help is available in [the forums \(\)](#)!

Use It!

How to use it!

The USBtinyISP is pretty easy to use, but here are a couple hints:



Indicator LEDs

There are two LEDs, a green one near the USB port and a red one near the cables.

The green LED indicates that the USB connection was successful. If the green LED doesn't ever light and you're sure it's in right, there was a problem with enumeration. If you're using a Windows or Linux machine and the green LED does not light up when you plug it in, there's a problem. If you have a newer Mac OS machine, try sending it commands via avrdude - the LED should light up then (strange but true!)

The red LED indicates that the USBtinyISP is 'busy' programming. You probably don't want to unplug it or the device being programmed while it's lit. However, if there's a software crash the LED may remain on even though it's not doing anything.

Programming Cables

There are two cables for programming: a 10-pin ISP cable and a 6-pin ISP cable. They are the two prevailing standards for in-circuit AVR programming. This programmer doesn't do JTAG programming

Jumper JP3 (USB power to target)

There's a jumper sticking out near the cables, JP3. When the jumper is in place (connecting the two wires) then that means that the USBtinyISP is providing 5V power to the device being programmed. If you don't want to power the device then just take the jumper out or make sure it's only on one of the wires.

The USBtinyISP can only provide 5V, up to about 100mA to the device. If you need more power then you should remove the jumper and power the device separately. (Alternately, if you're feeling adventurous you can reprogram the USBtinyISP to require 500mA from the USB port instead of 100mA but if you don't know how to do this I'd suggest not.)

Version 1.0 of USBtinyISP sends data to the device at 5V level no matter whether it's powering the device or not so make sure it's 5V compliant! (Note that there are 2 1.5K resistors in series with the data lines for protection)

Version 2.0 which is almost certainly what you've got, uses a level shifter so that if the jumper is not in place, it will use whatever the target voltage is, a lot better for your low-voltage devices!

So, if you have a device that needs to run at 3.3V, don't have the jumper in place!

Using it as an SPI interface

USBtinyISP can be used as a 'generic' SPI device. The best place to look for examples of how to use this is download the avrdude source code and read `usbtiny.c`

[Here is a submitted example of using it under Linux, in c++ \(\)](#). Thanks Matt D!

User Manual

How to use it!

The USBtinyISP is pretty easy to use, but here are a couple hints:



Indicator LEDs

There are two LEDs, a green one near the USB port and a red one near the cables.

The green LED indicates that the USB connection was successful. If the green LED doesn't ever light and you're sure it's in right, there was a problem with enumeration. If you're using a Windows or Linux machine and the green LED does not light up when you plug it in, there's a problem. If you have a newer Mac OS machine, try sending it commands via avrdude - the LED should light up then (strange but true!)

The red LED indicates that the USBtinyISP is 'busy' programming. You probably don't want to unplug it or the device being programmed while it's lit. However, if there's a software crash the LED may remain on even though it's not doing anything.

Programming Cables

There are two cables for programming: a 10-pin ISP cable and a 6-pin ISP cable. They are the two prevailing standards for in-circuit AVR programming. This programmer doesn't do JTAG programming

Jumper JP3 (USB power to target)

There's a jumper sticking out near the cables, JP3. When the jumper is in place (connecting the two wires) then that means that the USBtinyISP is providing 5V power to the device being programmed. If you don't want to power the device then just take the jumper out or make sure it's only on one of the wires.

The USBtinyISP can only provide 5V, up to about 100mA to the device. If you need more power then you should remove the jumper and power the device separately. (Alternately, if you're feeling adventurous you can reprogram the USBtinyISP to require 500mA from the USB port instead of 100mA but if you don't know how to do this I'd suggest not.)

Version 1.0 of USBtinyISP sends data to the device at 5V level no matter whether it's powering the device or not so make sure it's 5V compliant! (Note that there are 2 1.5K resistors in series with the data lines for protection)

Version 2.0 which is almost certainly what you've got, uses a level shifter so that if the jumper is not in place, it will use whatever the target voltage is, a lot better for your low-voltage devices!

So, if you have a device that needs to run at 3.3V, don't have the jumper in place!

Using it as an SPI interface

USBtinyISP can be used as a 'generic' SPI device. The best place to look for examples of how to use this is download the avrdude source code and read usbtiny.c

[Here is a submitted example of using it under Linux, in c++ \(\)](#). Thanks Matt D!

Drivers

AVR programmer & SPI interface

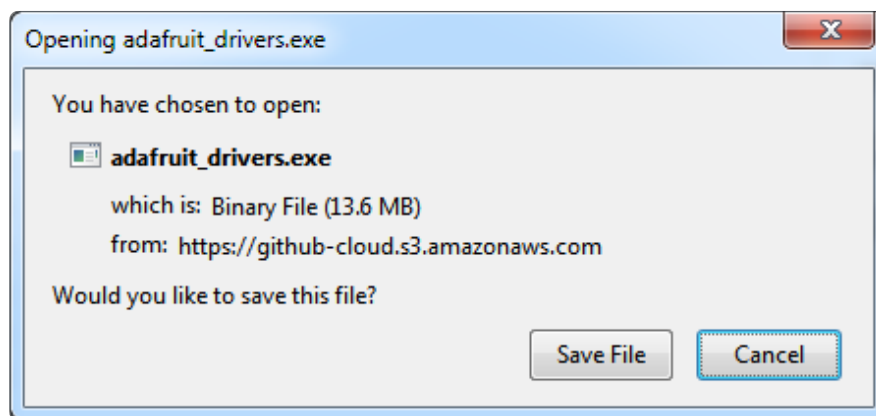
Windows 7, 8 & XP

Before you plug in your board, you'll need to possibly install a driver!

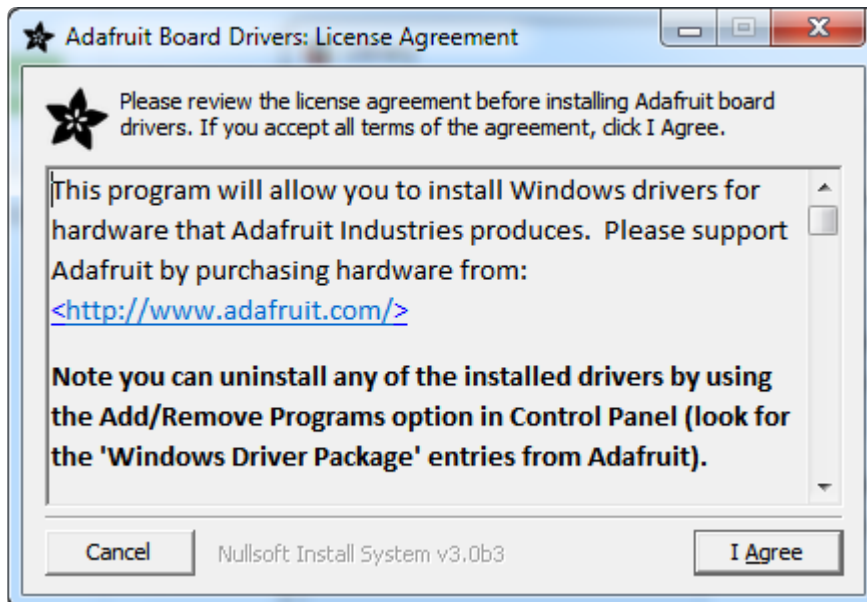
Click below to download our Driver Installer

[Download Adafruit Driver Installer](#)

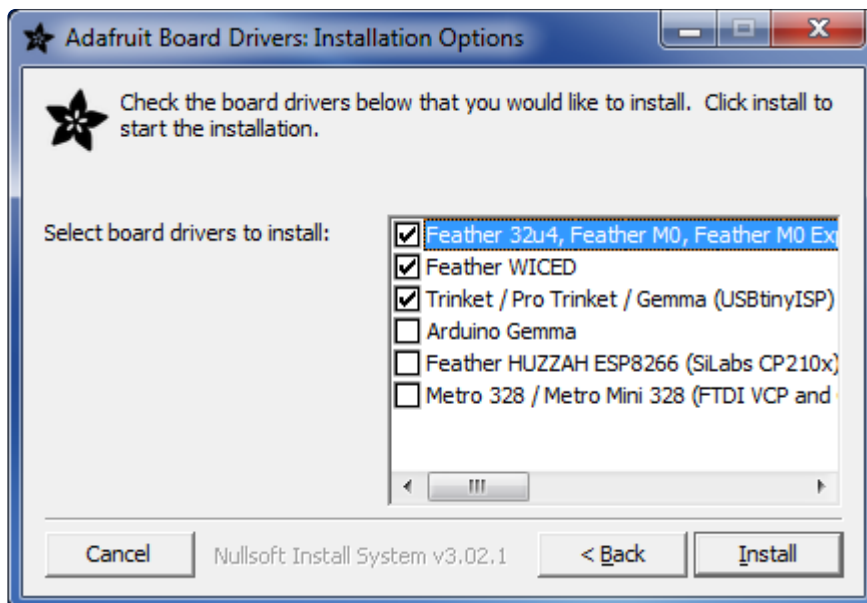
Download and run the installer



Run the installer! Since we bundle the SiLabs and FTDI drivers as well, you'll need to click through the license



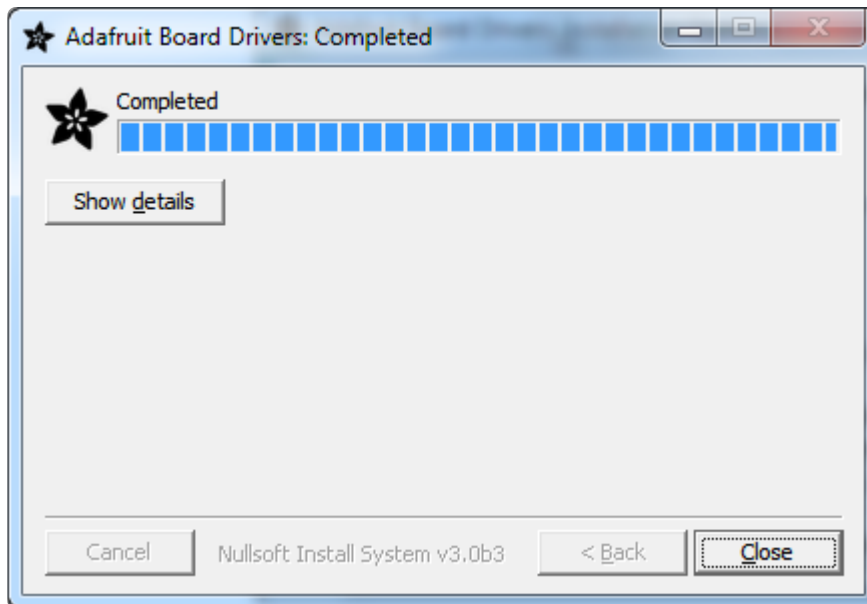
Select which drivers you want to install:



By default, we install the Feather 32u4 , Feather M0, Flora and Trinket / Pro Trinket / Gemma / USBtinyISP drivers.

You can also, optionally, install the Arduino Gemma (different than the Adafruit Gemma!), Huzzah and Metro drivers

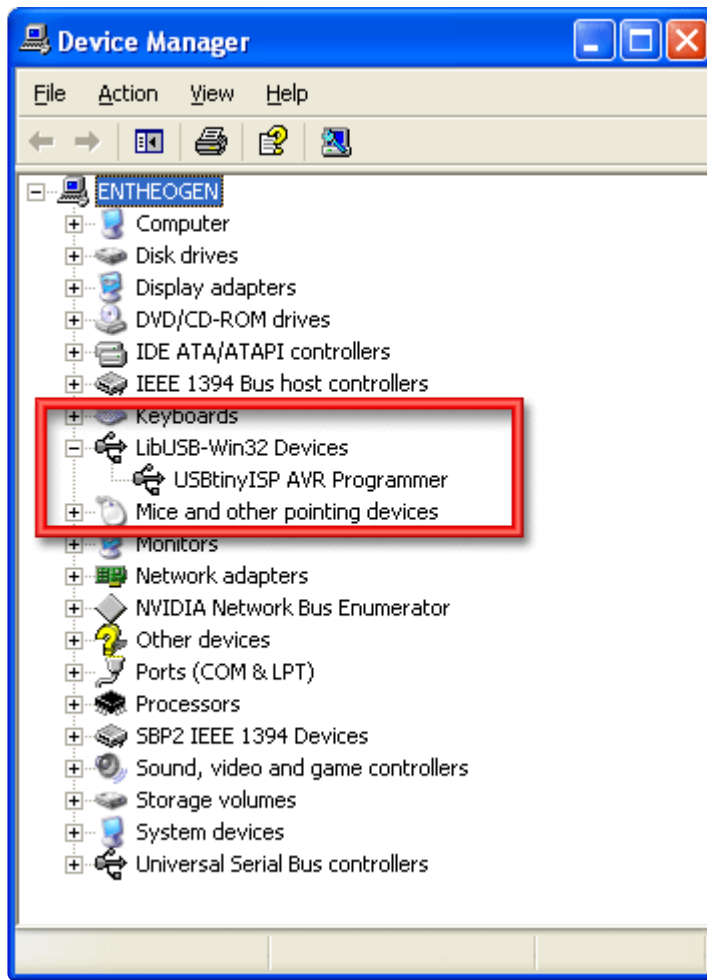
Click Install to do the installin'



Next, plug in the device into an open USB slot. You should get this popup:



Driver installed! Now go to your Device manager (Start Menu -> Settings -> Control Panel -> System -> Hardware) and look for the device:



Mac OS X & Linux

A driver is not required.

AVRDUDE

Using the programmer with AVRDUDE

AVRDUDE is a very popular command-line program for programming AVR chips. Avrdude version 5.5 and higher has built-in support for USBtinyISP! Look below for instructions for windows and mac on how to install the correct version of avrdude

To test that avrdude is working properly open a command line and run the command `avrdude -c usbtiny -p m8` while the device is plugged in (the green LED is lit).

```
C:\WINXP\system32\cmd.exe
C:\>avrdude -c usbtiny -p m8
avrdude: initialization failed, rc=-1
        Double check connections and try again, or use -F to override
        this check.

avrdude done. Thank you.

C:\>
```

You should get that response, which means that it communicated with the programmer but failed to find a connected chip.

If the programmer is not found, you will get this response:

```
C:\WINXP\system32\cmd.exe
C:\>avrdude -c usbtiny -p m8
Could not find USB device 0x1781/0xc9f
avrdude done. Thank you.

C:\>_
```

You can try unplugging and replugging it (a reset may help) or check if the driver is installed, etc.

If you connect the programmer to a target (say an [attiny2313 target board \(\)](#)) and run a `avrdude -c usbtiny -p t2313` you should get the following which indicates it communicated properly with the chip.

```
C:\WINXP\system32\cmd.exe
C:\Documents and Settings>avrdude -c usbtiny -p t2313
avrdude: AVR device initialized and ready to accept instructions
Reading ! ##### ! 100% 0.02s
avrdude: Device signature = 0x1e910a
avrdude: safenode: Fuses OK
avrdude done. Thank you.

C:\Documents and Settings>_
```

Using it is simple, just indicate `usbtiny` as the programmer type. The port option is ignored as it always uses USB.

You can use the -B option to specify the ISP speed. By default the value is 10 which means 100KHz clock, this is good for target clock speeds > 500KHz. If you want the high speed clockrate (400KHz) for target frequencies > 4MHz you can use "-B 1" to speed up programming. To calculate the SPI frequency from the delay value, use this formula:

SPI clock frequency in KHz = $1000/(1.5+B)$ where B is the delay value.

In general, the clock frequency should be at least 4 times larger than the target clock frequency. Try "-B 32" if you're having clocking issues, that should handle even 128khz clocks.

Can't get it working? Dont worry, help is available in [the forums](#) ()!

For Windows

[For a tutorial on how to install WinAVR, check out this page which has step by step instructions.](#) () Make sure you get the Dec 20. 2007 release or newer. That one has avrdude 5.5 with usbtiny support!

Don't forget to install the driver too (and check the driver page for more info).

For Mac OS X

[For a tutorial on how to set up your Mac for AVR programming and development, check out this page which has step by step instructions.](#) ()

If you install the AvrMacPack, it comes with Avrdude 5.5 and usbtiny is supported out of the box!

Otherwise, if you installed OSX-AVR, finish with these steps.

Grab the avrdude zip file from the [download page](#) ().

Replace avrdude and avrdude.conf wherever you have them installed, probably /usr/local/bin and /usr/local/etc but not necessarily depending on your development system!

To find where avrdude was installed type which avrdude into a Terminal window and the directory will pop up. To move the files, type `mv ~/usbtiny-avrdude/avrdude` assuming the new avrdude binary is in a folder called usbtiny-avrdude in your home directory. To find out where the conf file is type `avrdude -v` into a Terminal window. Do the same thing for `avrdude.conf`.

(Make backups of the old version of course.)

Close the terminal and open a new one. If you type in `avrdude -c usbtiny -p t2313` (without the usbtinyisp plugged in) it should say `Could not find USB device 0x1781/0xc9f` If not, check to make sure properly replaced avrdude and avrdude.conf.

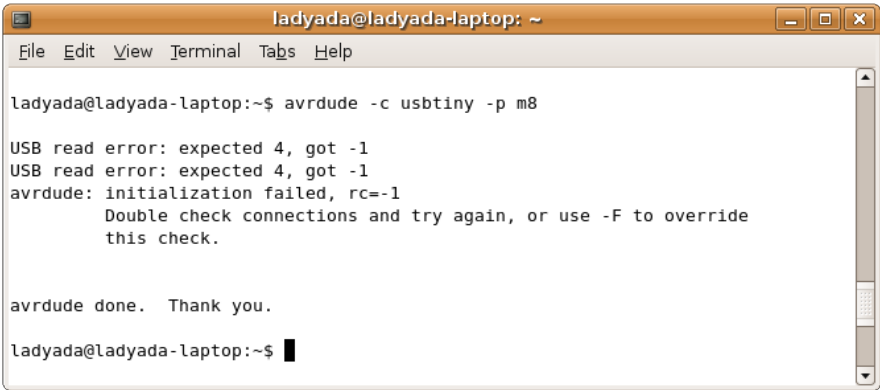
Now read about [how to use avrdude here!](#) ()

If that still doesn't work, you should compile it from the [source code](#) () which is guaranteed to work.

For Linux

[For a tutorial on how to set up your Linux machine for AVR programming and development, check out this page which has step by step instructions.](#) () Make sure you get avrdude 5.5 version or later!

If you get this response, it's a permissions problem with USB. You can run as root or...

A terminal window titled 'ladyada@ladyada-laptop: ~' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the command 'ladyada@ladyada-laptop:~\$ avrdude -c usbtiny -p m8' and the following output: 'USB read error: expected 4, got -1', 'USB read error: expected 4, got -1', 'avrdude: initialization failed, rc=-1', and 'Double check connections and try again, or use -F to override this check.' followed by 'avrdude done. Thank you.' and the prompt 'ladyada@ladyada-laptop:~\$' with a cursor.

```
ladyada@ladyada-laptop:~$ avrdude -c usbtiny -p m8

USB read error: expected 4, got -1
USB read error: expected 4, got -1
avrdude: initialization failed, rc=-1
      Double check connections and try again, or use -F to override
      this check.

avrdude done. Thank you.

ladyada@ladyada-laptop:~$ █
```

Thanks to a friendly user, there is a quick fix so you don't have to run it as root:

A udev rule, placed in a new rule file (named whatever you'd like) in `/etc/udev/rules.d/` (or wherever your distro will expect it) will set the permissions for the USBtinyISP correctly.


```
SUBSYSTEM=="usb", SYSFS{idVendor}=="1781", SYSFS{idProduct}=="0c9f",  
GROUP="users", MODE="0666"
```

or maybe

```
SUBSYSTEM=="usb", SYSFS{idVendor}=="1781", SYSFS{idProduct}=="0c9f",  
GROUP="adm", MODE="0666"
```

depending on your distro

(one line!) should do the trick. Sane does something very similar to allow regular users to access a scanner.

Another user suggests:

The udev examples given don't work on some systems as the SYSFS parameter is deprecated. The following rule works on recent Ubuntu systems and should probably work on other newer Linux systems:

```
SUBSYSTEM=="usb", ATTR{product}=="USBtiny", ATTR{idProduct}=="0c9f",  
ATTRS{idVendor}=="1781", MODE="0660", GROUP="dialout"
```

AVRStudio

The latest AVRStudio supports AVRDUDE so you don't need a bridge. This page up for reference

AVRISP/STK500v2 compatibility bridge

There's already lots of good software for programming and debugging AVR's, such as [AVRStudio \(\)](#) (the official development software from Atmel). However, AVRStudio only really supports [STK500 \(\)](#) and [AVRISP \(\)](#) programmers (the official programmers).

Since there are many times when you may want to use AVRStudio, I've written some software glue that will let you use your USBtinyISP in STK500/AVRISP compatibility mode.

Notes

Nearly all of AVRISP (STK500v2) functionality is emulated at this point.

- Oscillator calibration is not implemented at all, due to hardware constraints.

- AT89 programming is not tested and probably doesn't work.
- Word-mode flash programming isn't tested (although if someone can tell me of a chip that does this I'll test it).
- ISP clock speed setting is emulated, although the speeds are not 'true.' that is, a 400KHz ISP clock just means that at most, it will be clocked at 400KHz. In reality it's often slower due to the programmer coping with USB stuff.

For general development, I strongly suggest using [avrdude \(\)](#) as this is a bit slower and is probably flakier.

Please note, this software is in alpha. Your reports, comments and suggestions are appreciated!

(post it in the [forum \(\)](#))

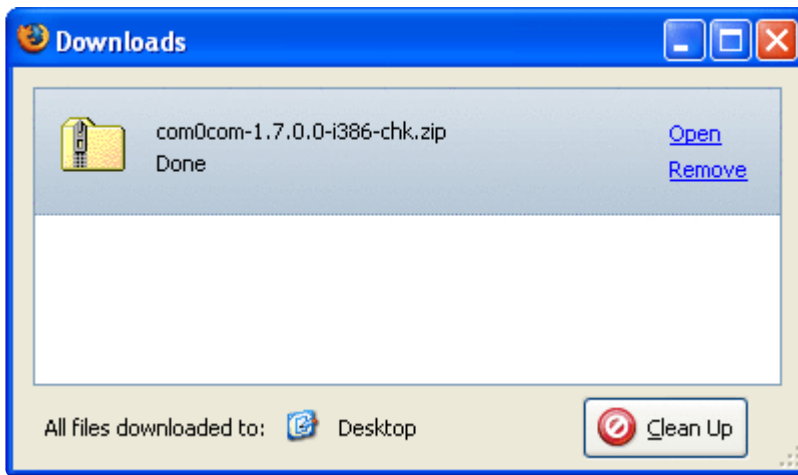
If you send a bug report please let me know which chip you are using and what specifically is failing.

Step 1. COM bridge

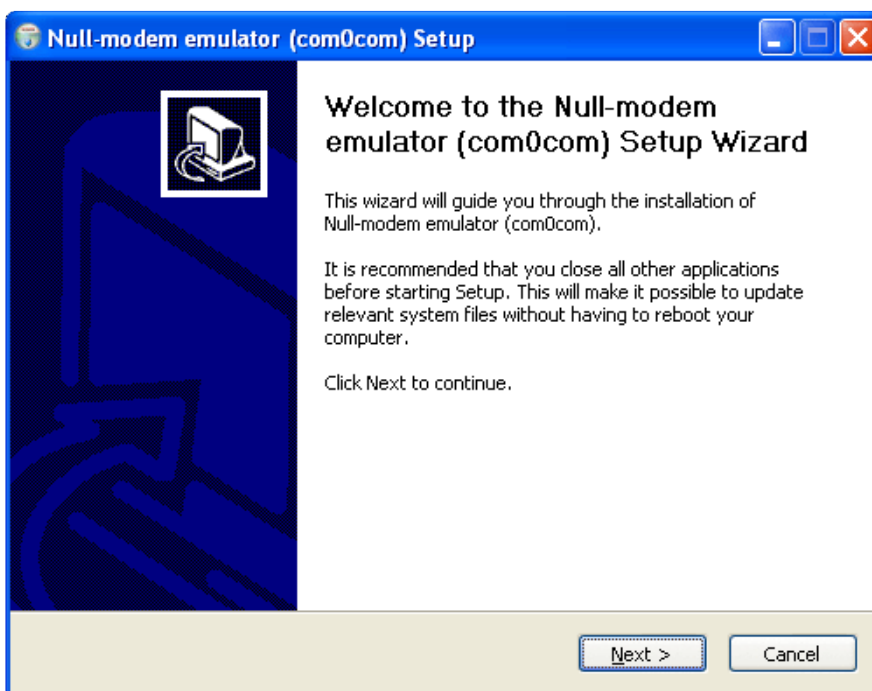
The AVRISP, which we will be emulating, usually connects to a PC through the serial (COM) port. The trick here is to install a COM bridge, a piece of software that makes two virtual COM ports that are hooked up to each other so when you write to one it appears on the other. The compatibility software sits on one COM port, pretending to be a AVRISP while the AVRStudio software talks on the other, thinking that it's connected to a genuine programmer!

You'll only have to install this software once.

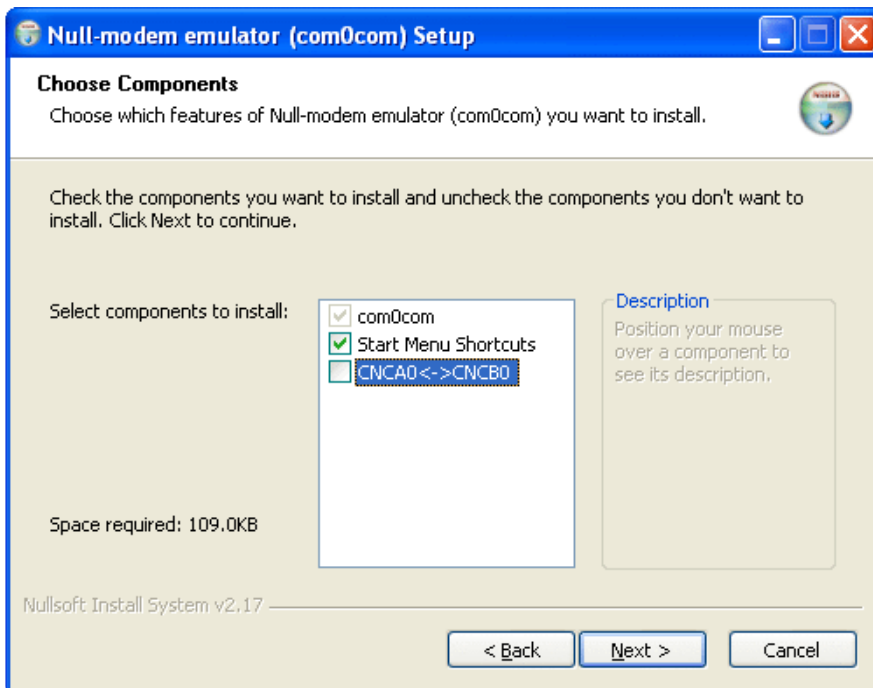
[Download \(\)](#) the free [com0com \(\)](#) bridge [from sourceforge \(\)](#). Make sure to grab the binary, not source code, version.



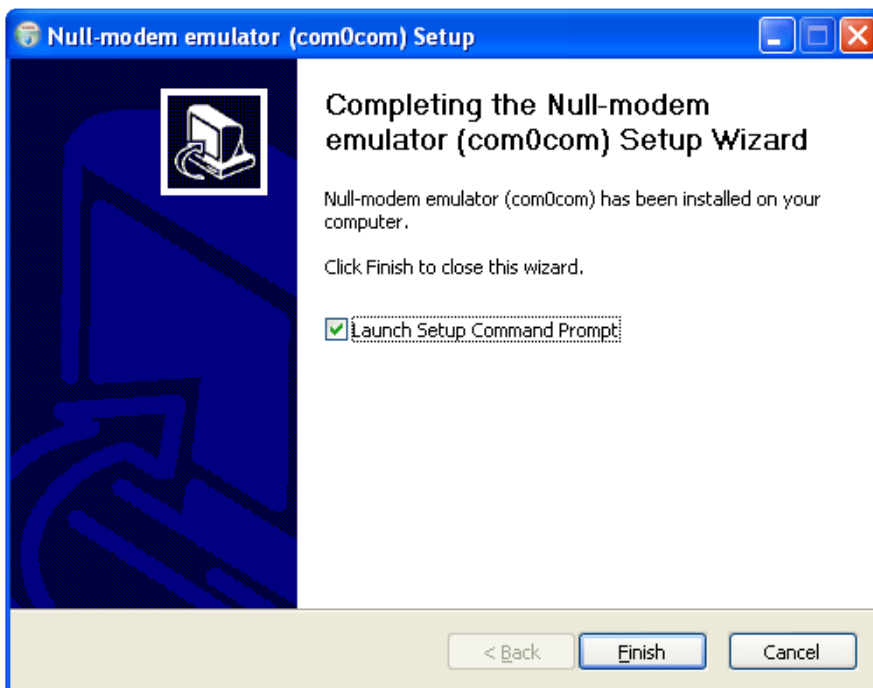
Save, extract and run Setup



Click Next



Important! Unclick the CNCA0<->CNCB0 checkbox!

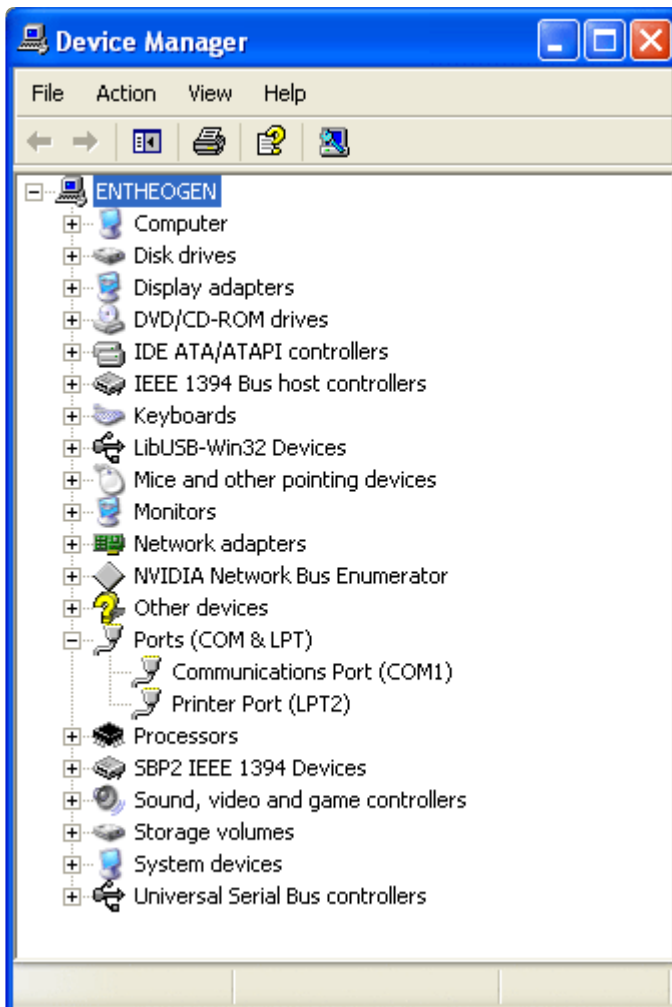


Launch the setup command prompt

```
Setup for com0com
Enter 'help' to get info about usage of Setup for com0com.
command> help
Setup for com0com
Usage:
  loptions l <command>
Options:
  --output <file>      - file for output, default is console
Commands:
  install <n> <prmsA> <prmsB> - install a pair of linked ports with
  or                          identifiers CNCnA<n> and CNCnB<n>
  install <prmsA> <prmsB>    (by default <n> is the first not used number),
                             set their parameters to <prmsA> and <prmsB>
  remove <n>                - remove a pair of linked ports with
                             identifiers CNCnA<n> and CNCnB<n>
  change <portid> <prms>    - set parameters <prms> for port with
                             identifier <portid>
  list                       - for each port show its identifier and
                             parameters
  preinstall                - preinstall driver
  update                    - update driver
  uninstall                  - uninstall all ports and the driver
```

You can type in help for a list of commands.

We want to install two ports, first check the device manager (Start->Settings->Control Panel->System->Hardware)



Under ports you'll see a list of COM ports. Both of the virtual ports we're making have to be between COM1 and COM9. Looks like only COM1 is being used.

```
Setup for com0com
PortName=<name>          - set port name to <name>
                          <port identifier by default>
EmuBR=<yes|no>           - enable/disable baud rate emulation
                          <disabled by default>
EmuOverrun=<yes|no>      - enable/disable buffer overrun
                          <disabled by default>

Special values:
-                          - use driver's default value
*                          - use current setting

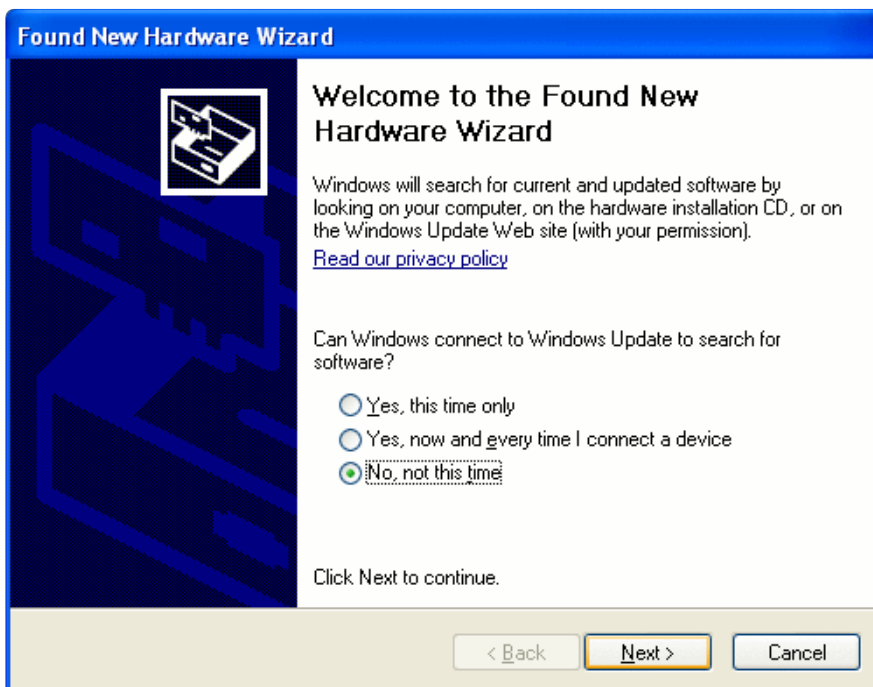
Examples:
install - -
install 5 * *
remove 0
install PortName=COM2 PortName=COM4
install PortName=COM5, EmuBR=yes, EmuOverrun=yes -
change CNCA0 EmuBR=yes, EmuOverrun=yes
list
uninstall

command> install PortName=COM2 PortName=COM6
          CNCA0 PortName=COM2, EmuBR=-, EmuOverrun=-
          CNCB0 PortName=COM6, EmuBR=-, EmuOverrun=-
command>
```

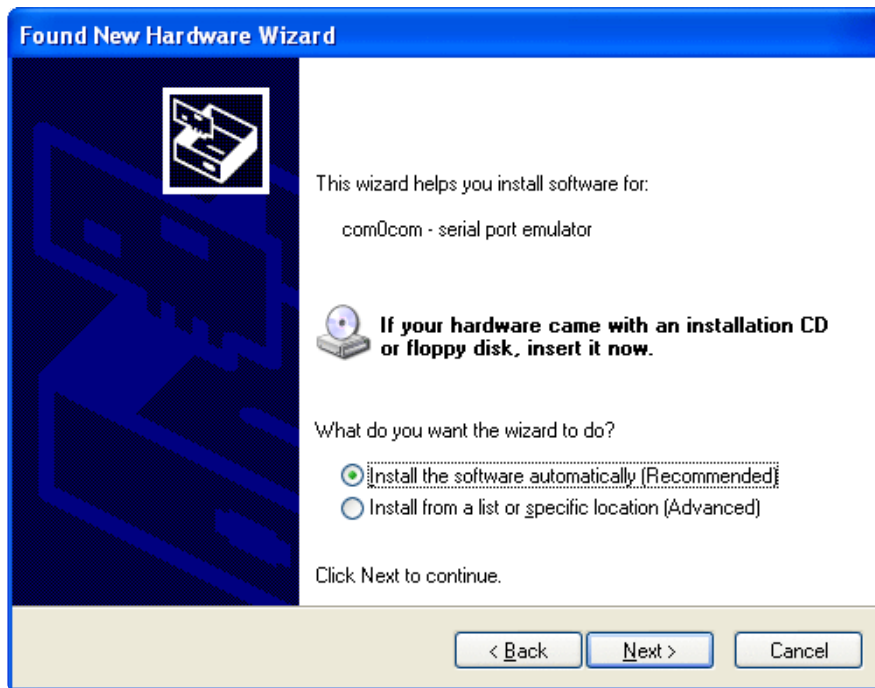
Type in `install PortName=COM2 PortName=COM6`, changing the COM2 to COM3 or COM4 if any of those are already being used. The second port should be anything between COM5 and COM10. Keep track of these ports because you'll need to refer to them later.



Once you hit return, windows will popup a "found new hardware" wizard as it thinks there are two more COM ports installed.

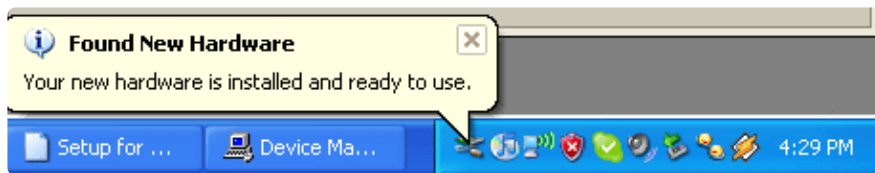


Select "No not this time", click Next

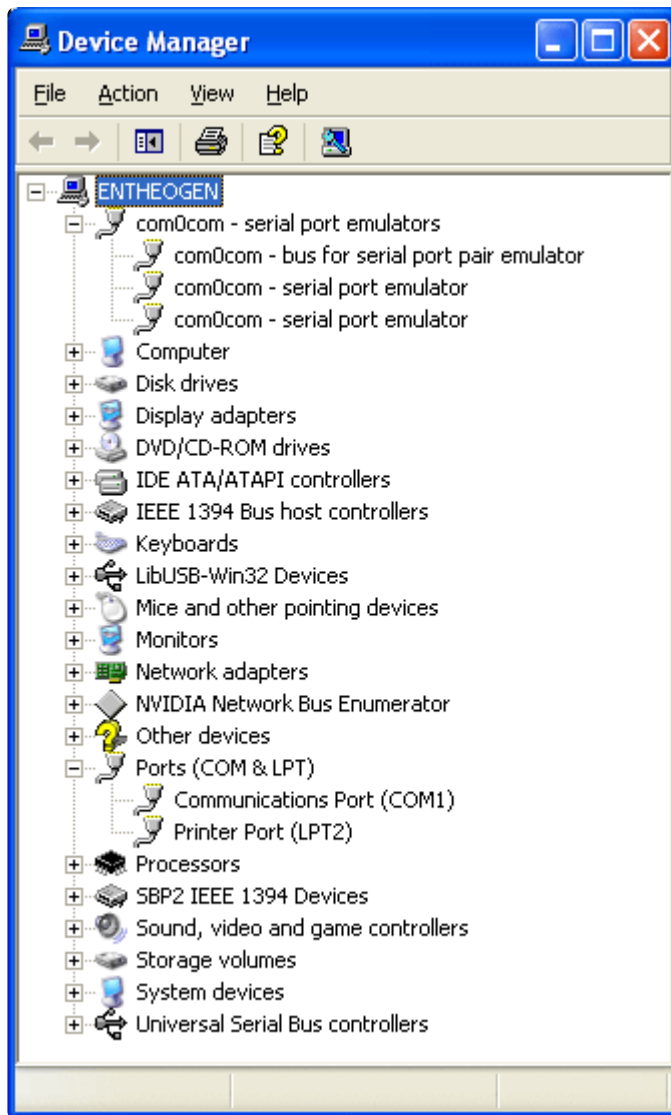


Select "Install automatically" and hit next.

You may have to do this twice, once for each COM port.



Next, check the device manager again, your new ports should have shown up!



Step 2. Install AVRStudio

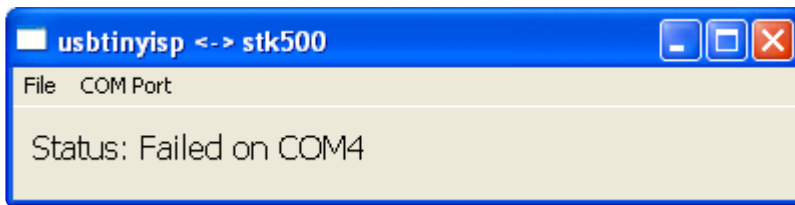
Or whatever STK500 software you'll looking to use.

Step 3. Download USBtiny500

[Grab this from the download page \(\)](#). It's the software we run to provide the bridging.

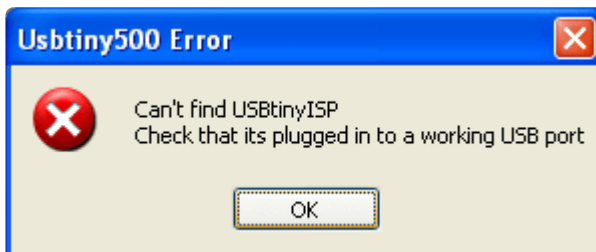
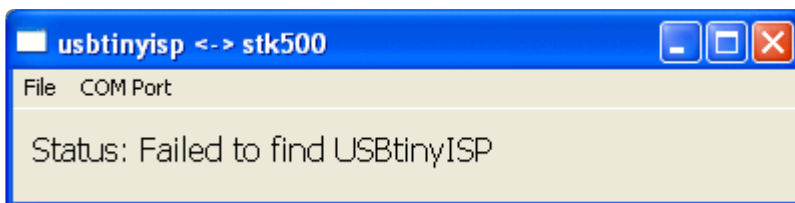
Install the software and run USBtiny500.

Select one of the COM ports from the pair you made using com0com. If you select one that is not available you'll get the following warnings:



The software remembers the COM port you've selected so you should only have to do this once.

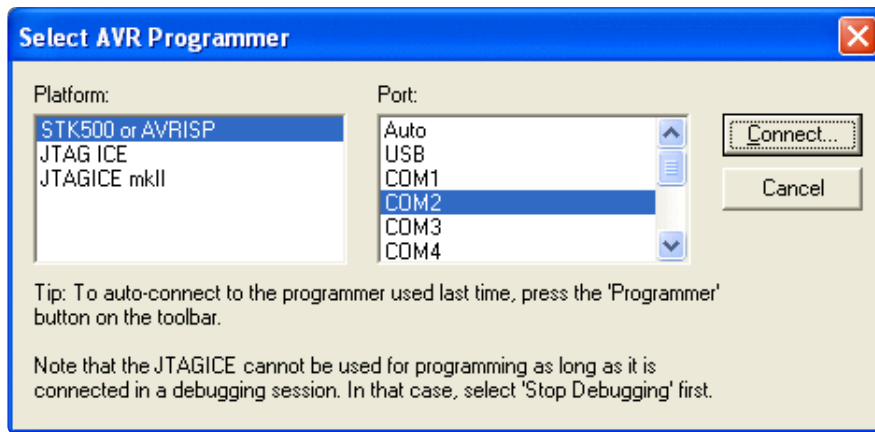
Next up the software looks for a USBtinyISP, if it fails to find one it will display:



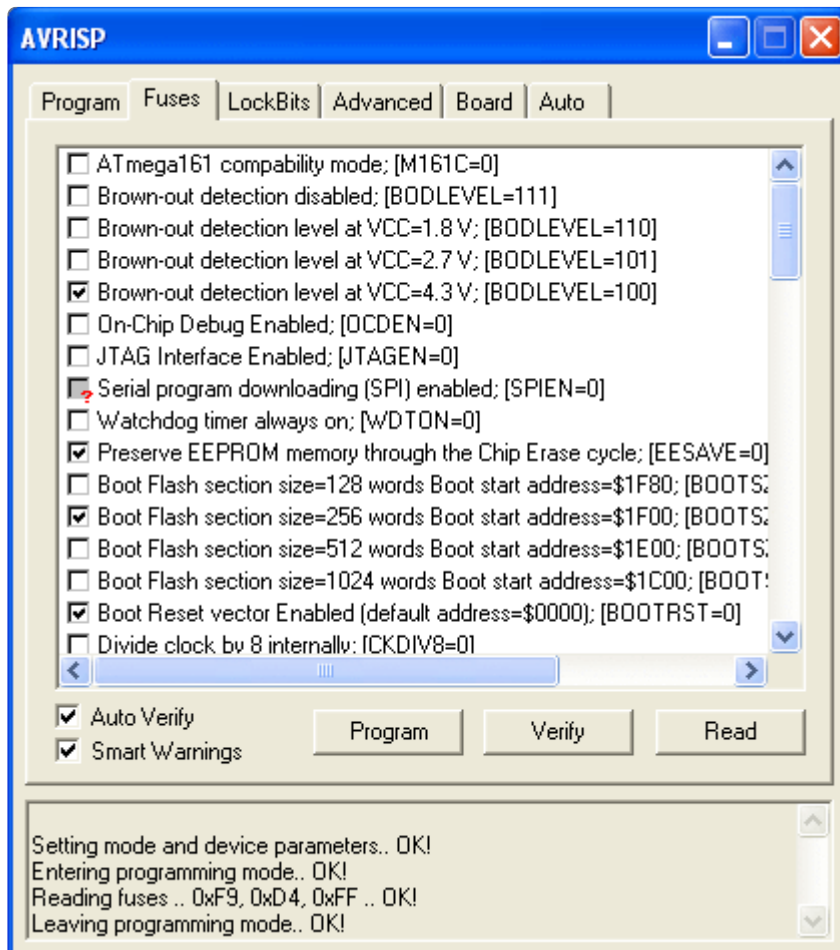
But if both the COM port and the programmer are found you'll get a ready message:



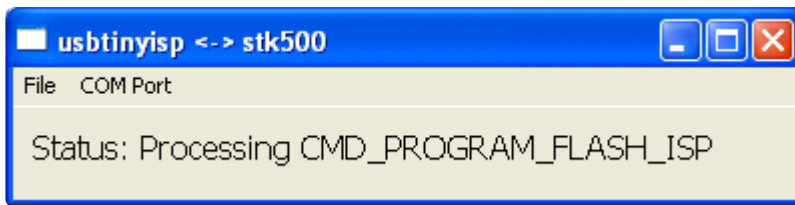
Next, start up AVRstudio, and open the programmer communication panel. Select AVRISP and choose the other COM port from the pair you made with com0com. AVR studio remembers this selection so you'll only have to do this once.



You should be able to communicate to your target, through the software. If you're having problems talking to the chip, check the power, whether a crystal or clock is necessary and that the ISP clock is not too fast! You can change the ISP clock in the Board tab of AVR studio:



The latest message passed through the bridge is displayed in the usbtiny console:



Download

Windows Drivers

For windows, we use a modified LibUSB driver. You can download it here:

- [Windows USBtinyISP signed driver \(\)](#) built with libusb v1.12. The windows binaries below are built for this driver. If you want to use the latest WinAVR use this

For historic reasons, we have a:

- [Windows USBtinyISP driver \(\)](#) built with libusb v1.10. Use this for older WinAVR's. Don't use this unless you have to interface with old versions of software that are bound to libusb v1.1

AVRDUDE

[AVRDUDE is the recommended software to use if you want to program/flash an AVR microcontroller. \(\)](#)

Hardware/Firmware Files for v2.0

The latest!

- [EagleCAD schematic and board files \(\)](#)
And [schematic image \(\)](#)

The firmware is based off of the [USBtiny code originally here \(\)](#).

- [Zip with all v2.0 firmware files \(\)](#) (relevant files including precompiled hex file are in the spi subfolder)

Use the precompiled .hex file and Makefiles as your compiler may not be able to squeeze the code down to fit in the chip. Modify the Makefile in spi as needed and, in the spi folder, type in "make fuse flash" Beyond that, you're on your own. [You must use avr-gcc v3.4.6 and avr-libc v1.4.4 as part of Winavr-20060421 \(\)](#) to compile the firmware. Please do not post to the forums asking for help on how to compile or burn the firmware.

Please note this firmware contains the Adafruit VID/PID, while the firmware is open source the VID/PID is not. To use in your own products, replace those values in usbtiny.h You can purchase a USB VID for yourself at <http://www.usb.org/developers/vendor/>

Hardware/Firmware Files for v1.0

The board design is not single sided, but it's close: you will need to solder in the 5 wires that would go on the top. I have successfully toner-transfer etched this design.

- [EagleCAD schematic and board files \(\)](#)
- [Schematic in PNG format \(\)](#)

This firmware is based off of the [USBtiny code originally here \(\)](#).

- [Zip with all v1.0 firmware files \(\)](#) (basically modified USBtiny-spi) [You must use avr-gcc v3.4.6 and avr-libc v1.4.4 as part of Winavr-20060421 \(\)](#) to compile the firmware. Please do not post to the forums asking for help on how to compile or burn the firmware.

[v1.04 is a possible hotfix for some flakiness \(\)](#), you can try either one.

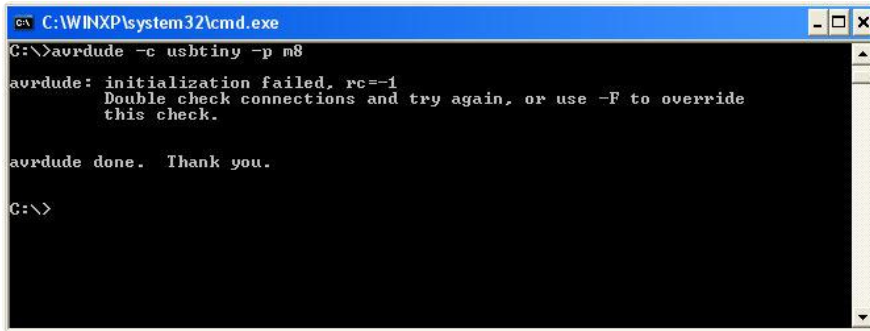
USBtiny500 compatibility bridge for AVR Studio

[Download the latest installer \(\)](#)
[And source code \(\)](#)

HELP!!!

Frequently Asked Questions

I'm running avrdude and I get "Initialization failed, rc=1"

A screenshot of a Windows command prompt window titled "C:\WINXP\system32\cmd.exe". The window shows the following text:

```
C:\>avrdude -c usbtiny -p m8
avrdude: initialization failed, rc=-1
        Double check connections and try again, or use -F to override
        this check.

avrdude done. Thank you.

C:\>
```

This response from avrdude means that it can talk to the programmer but the programmer can't talk to the chip.

Check:

- Are your 10 and 6 pin cables correct? compare with the pictures in the manual.
- Are you either providing power to the chip (have the jumper in place) or are providing power to the programmer through the VCC header pin? If the jumper is not in place, the buffer chip (74ahc125) will require at least 2.5V from the target.
- If you programmed your chip to have a very slow clockspeed [use the -B flag, as shown here \(\)](#) to slow down the chip. "-B 32" should do the job most of the time, but you can go as slowly as "-B 250"
- Is the chip powered? AVCC, VCC, and all GND pins must be connected.
- Does it have a clock or crystal (if necessary?)
- Is anything keeping the MISO/MOSI/SCK/Reset pins from switching? (ie are the loaded down)
- Does the target chip need a crystal? is the crystal oscillating?
- Are you sure its wired up correctly? Use an oscilloscope to watch the reset line on the chip, it should flicker up and down. Watch the SCK pin and make sure you see a 8-pulse clock train. Check that you didnt get MISO/MOSI swapped. etc.
- Do you need to 'jumper' the output 1.5K resistors? If you are not using the USBtinyISP for SpokePOV communication, this is recommended, especially with target chips that have something connected to the MISO/MOSI/SCK lines. Check the end of the [soldering instructions \(\)](#) for how to do this

Most of the time, your programmer is working fine, check your cables and pins and that the target device is properly wired up.

We have never encountered a USBtinyISP that got to this point where it talks to the computer but not a chip and it is the kit that is defective. It is ALWAYS a bad chip or a wiring, clocking, bit-speed, powering or output-resistor problem.

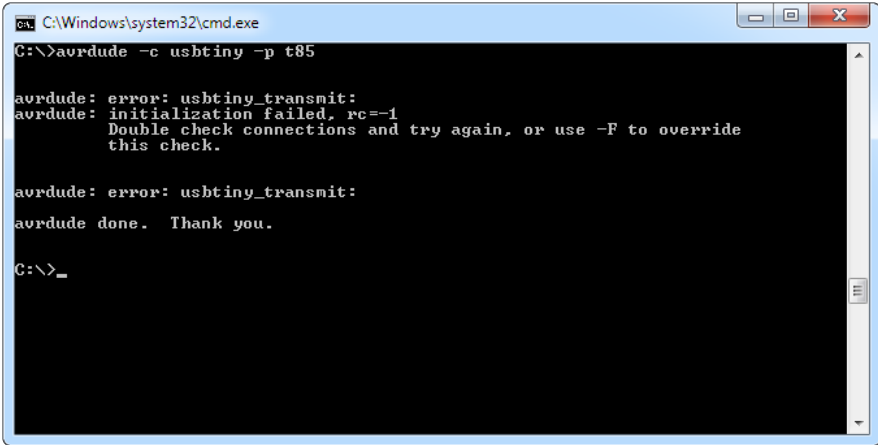
It doesn't work with a USB 3 port

We've noticed that USB 3 ports are sometimes a little more sensitive than USB 2. [In this case you can try changing the cable \(see this article\) \(\)](#) or placing an every-day USB hub between the USBtinyISP and your computer.

I'm running avrdude and I get "error: usbtiny_transmit: initialization failed, rc=-1"

This response means that it found the USBtiny, but could not communicate with it. The likely reason is that you found a version of avrdude that was not from WinAVR. Differences in new versions of avrdude mean that it doesn't work with the libusb-win32 USBtinyISP driver.

For more information, see https://github.com/adafruit/Adafruit_Windows_Drivers/issues/22



```
C:\Windows\system32\cmd.exe
C:\>avrdude -c usbtiny -p t85

avrdude: error: usbtiny_transmit:
avrdude: initialization failed, rc=-1
        Double check connections and try again, or use -F to override
        this check.

avrdude: error: usbtiny_transmit:
avrdude done. Thank you.

C:\>_
```

It's not working!

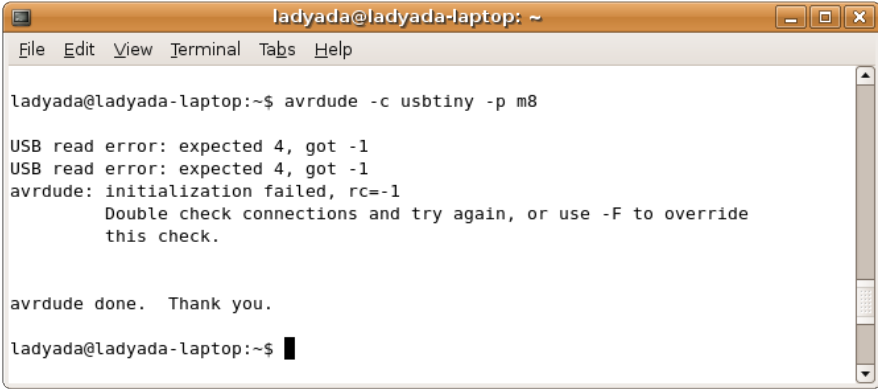
Check:

- Are all the parts in correctly?
- Are there any cold-solder joints or bridges?

- Are the cables attached properly?
- Are the cables in backwards? (The red wire should be at the #1 spot)
- Is the green light on, indicating USB enumeration was successful? (note with some Macs it appears the green light only turns on after Avrdude talks to the programmer, weird!)
- Is the driver installed (windows)?
- Is the device plugged into a target board? (I.e. is it connected to a chip)
- Is the target chip powered?
- Does the target chip need a crystal? is the crystal oscillating?
- Are you trying to provide 5V USB power to a device that is already powered?

before posting to the forums!

I'm running avrdude and I get "USB read error: expected 4, got -1" (or something similar)

A terminal window titled 'ladyada@ladyada-laptop: ~' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the command 'avrdude -c usbtiny -p m8' and the following output: 'USB read error: expected 4, got -1', 'USB read error: expected 4, got -1', 'avrdude: initialization failed, rc=-1', 'Double check connections and try again, or use -F to override this check.', and 'avrdude done. Thank you.' The prompt 'ladyada@ladyada-laptop:~\$' is visible at the end.

```
ladyada@ladyada-laptop:~$ avrdude -c usbtiny -p m8
USB read error: expected 4, got -1
USB read error: expected 4, got -1
avrdude: initialization failed, rc=-1
      Double check connections and try again, or use -F to override
      this check.

avrdude done. Thank you.
ladyada@ladyada-laptop:~$
```

There's a priv's problem, [check the usbtinyisp avrdude linux instructions for more info \(\)](#).

My 64 bit computer doesn't seem to work!

Some very old kits have this problem, thanks to intrepid assistants, a patch has been submitted to avrdude project, if you want to fix it yourself, simply replace the lines in `usbtiny.c` in avrdude that have

`sizeof(res)`

with

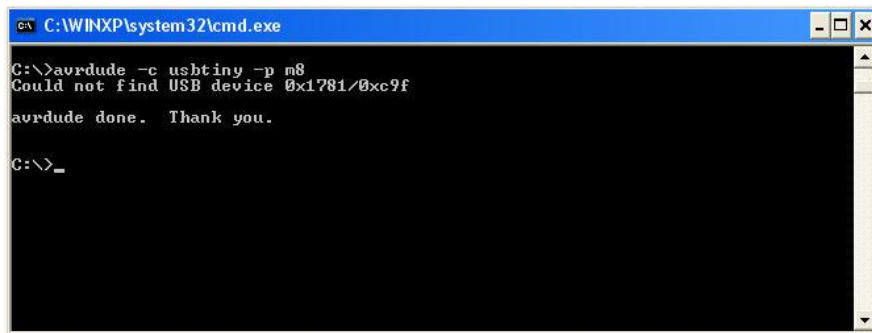
`4`

This was fixed in 2010 or so

I'm having trouble compiling/burning the chip for this project...

Use the precompiled .hex file and Makefiles as newer avr-gcc compilers may not be able to squeeze the code down to fit in the chip. Modify the Makefile in spi as needed and, in the spi folder, type in "make fuse flash" Beyond that, you're on your own!

I'm running avrdude and I get "Could not find USB device 0x1781/0xc9f"



```
C:\WINXP\system32\cmd.exe
C:\>avrdude -c usbtiny -p m8
Could not find USB device 0x1781/0xc9f
avrdude done. Thank you.
C:\>_
```

This response means it could not find the programmer. There are many possible reasons:

1. If you are using the latest WinAVR (which was built with an old version of the libusb driver library) you must use the v1.10 driver. Uninstall the driver you have and make sure you get the right one installed.
2. Try unplugging and re-plugging it in
3. Make sure the driver (windows only) is installed.
4. Make sure the green LED is lit (which means its powered and has performed USB enumeration).
5. Make sure you have libusb installed and that it is the correct version. (linux/mac)
6. Make sure you have replaced the old version of libusb0.dll with the one from the download page.
7. Make sure you are not providing power to the target through USB when it already has power. Check the jumper in the end of the device and remove it, if so.
8. If you programmed the usbtinyisp chip, make sure the fuses are set properly.

I'm having trouble building this project from scratch...

Beyond the advice, here if you can't get this project working on your own I suggest buying a kit. It's not a simple project to debug!