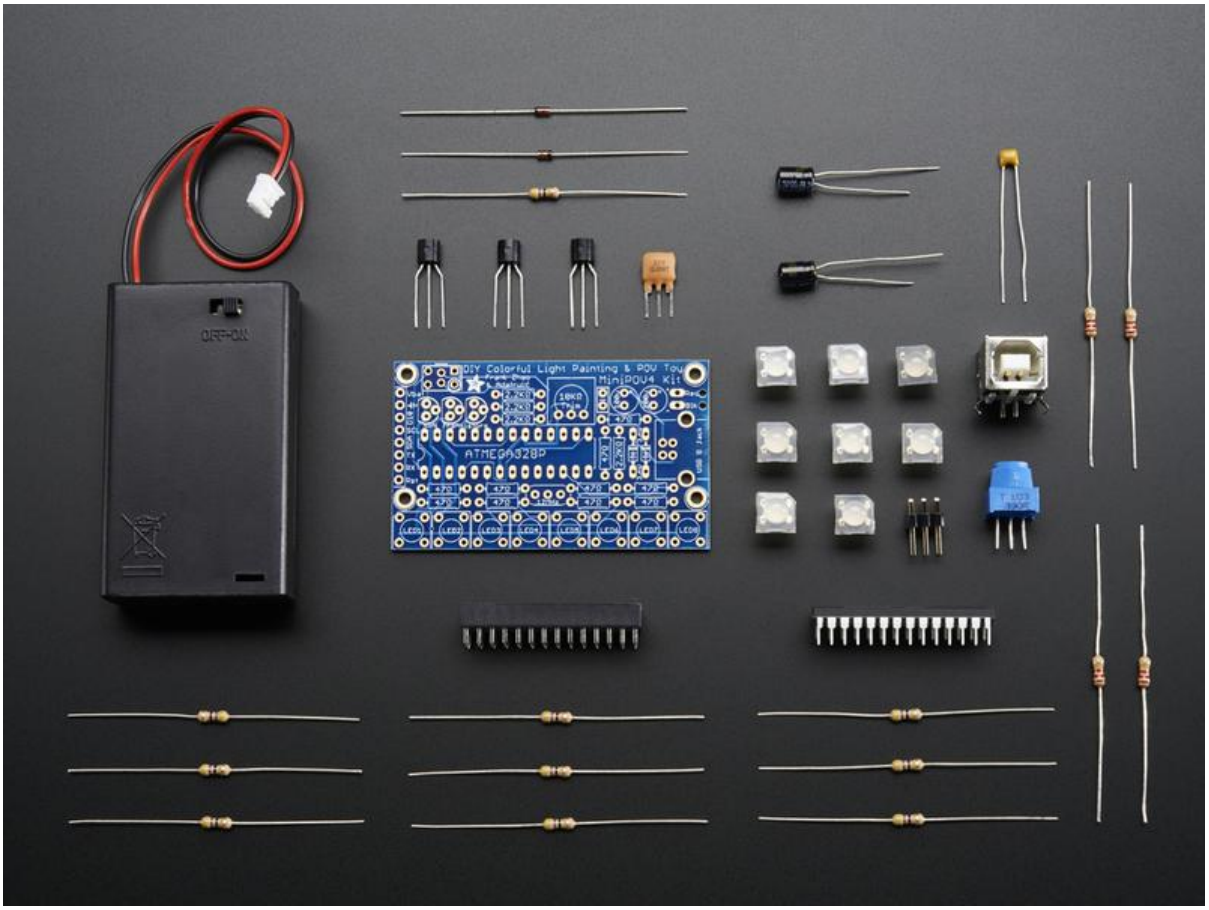
Adafruit MiniPOV3 Kit

Created by lady ada



https://learn.adafruit.com/minipov3

Last updated on 2022-12-01 01:57:30 PM EST

Table of Contents

Overview

Serial POV!

This project is the third revision of the MiniPOV. This version is nearly identical to the last version, MiniPOV2 but uses the serial port (possibly with a USB/Serial converter) instead of a parallel port, for programming. Because the programmer is built into the kit, one does not need a special "microcontroller programmer". This version can be used with PCs (Linux/Unix or Windows) and Macs (running MacOS X and with a USB/serial converter).

I intend this project to be an ideal starting place for anyone who would like to:

1. learn how to solder
2. learn how to assemble simple kits
3. learn how to program microcontrollers
4. make blinky stuff

Kits are available which provide all the components necessary to build & get going.

F.A.Q.

What is included in the kit?

The kit comes with 8 red LEDs, a preprogrammed microcontroller, a serial port connector and a AA battery case.

If you buy the kit from the Adafruit shop (), the preprogrammed code just blinks the

LEDs in order. Other stores often have a preprogrammed image already in it, for example the Make store has "MakeZine!" as seen on the front page.

Is a sensor included?

No sensor is included. There is a spot for a sensor on the PCB but that is more for people who are interested in modifying the kit. A sensor is not necessary for operation.

How does the kit know where your hand is when you wave it?

It doesn't! This kit is very simple, it just repeats whatever it's programmed to display over and over again. That means if you wave your hand back and forth, it will appear backwards half the time.

Can I attach this to my bike/fan/etc?

Sure, there are 4 holes for such a purpose.

It only works for a second...

For some reason, a batch of microcontroller chips (datecode 0627 ?)have been acting odd and shut themselves off after a few seconds. The problem seems to be that the RESET pin is being wonky, to fix this take a resistor (anywhere between 2Kohm and 20Kohm) and solder it to the top two pins of the microcontroller so that the RESET pin (pin #1) is connected through the resistor to the POWER pin (pin #20).

If this doesnt work, refer to the next question in the FAQ!

It's not working! Help!

Don't panic! Use the forums () for tech support.

Make it!

Ready?

This is a very easy kit to make, just go through each of these steps to build the kit:

1. Tools and preparation ()
2. Check the parts list ()
3. Assemble the kit ()
4. Installing the software ()
5. Programming/Changing the image ()

Preparation

Prep

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), 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).

Click here to upgrade to a Genuine Hakko FX-888 adjustable temperature soldering iron. (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).

Click here to buy a spool of lead-free solder (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)

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

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

Flush Diagonal Cutters

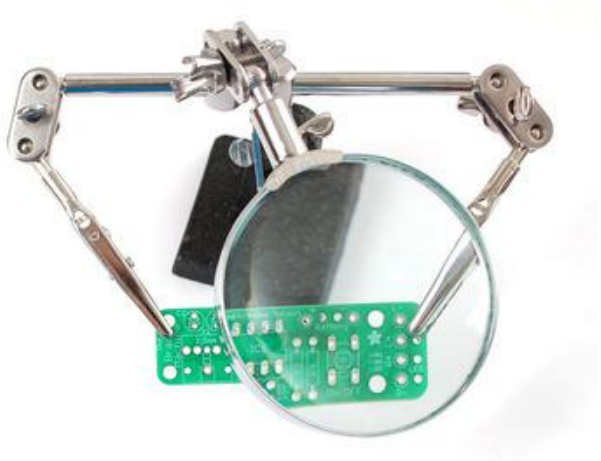
You will need flush diagonal cutters to trim the wires and leads off of components once you have soldered them in place.

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

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).



Helping Third Hand With Magnifier

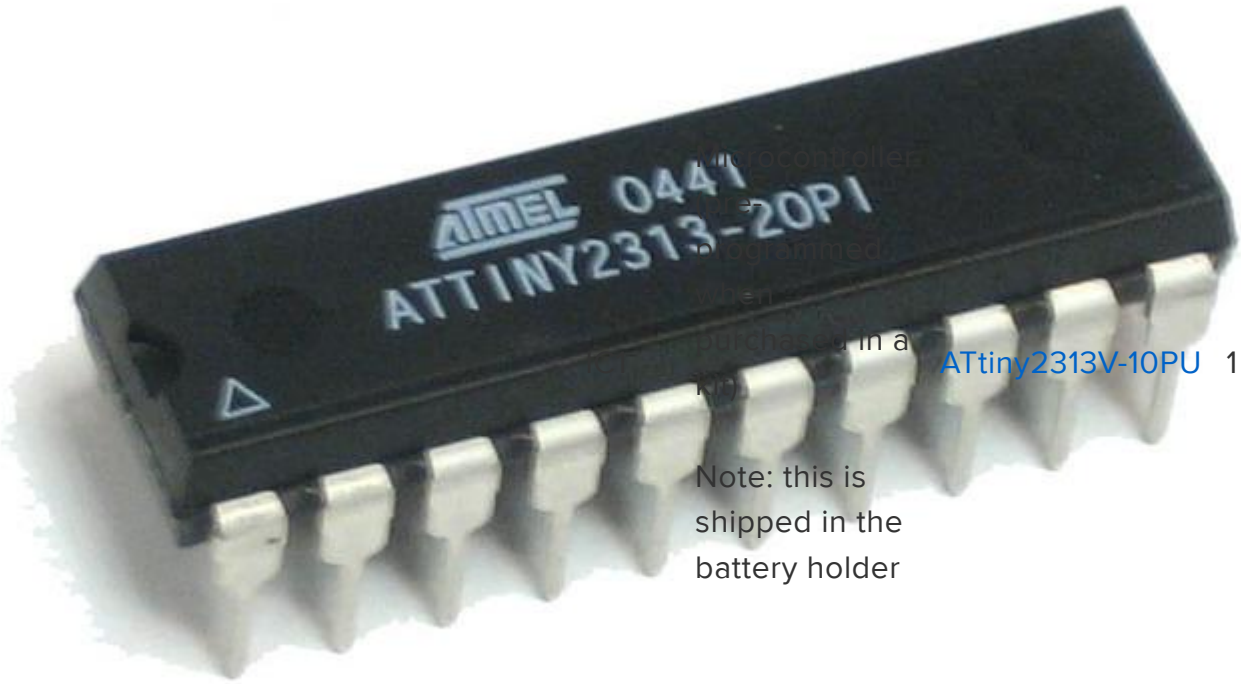
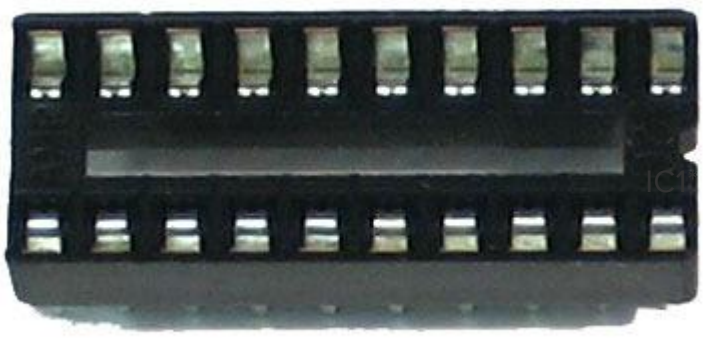
Not absolutely necessary but will make things go much much faster, and it will make soldering much easier.

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

Parts List

Parts List

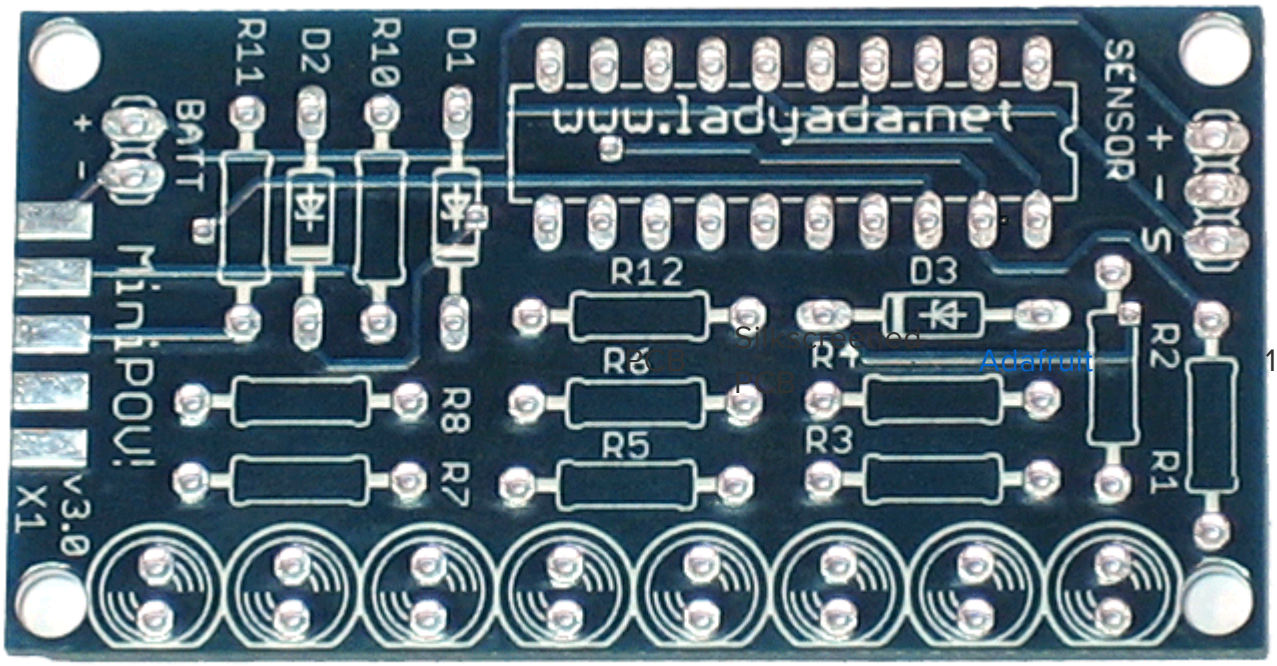
Check to make sure your kit comes with the following parts. Sometimes we make mistakes so double check everything and email support@adafruit.com if you need replacements!

Picture	Name	Description	More information	Q
	IC1	Microcontroller programmed with Purchased in a kit Note: this is shipped in the battery holder	ATtiny2313V-10PU	1
	IC1	20 Pin Socket Note: this is shipped in the battery holder	Generic	1
	R10 R11 R12	1/4W 5% 4.7K resistor (yellow purple red)	Generic	3

	R1-R9	1/4W 5% 100 ohm (brown black brown)	Generic	8
	D2 D3	5.6V Zener diode	1N5232B	3

U1		MPD SBH-321AS	1

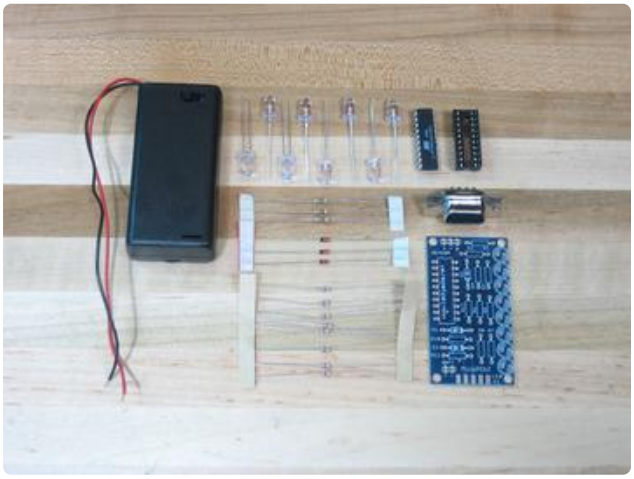
	D1-8	Red LED	5mm red diffused	8
	X1	DB-9 female connector w/ solder cup	Norcomp 171-009-203L001	1



Solder it!

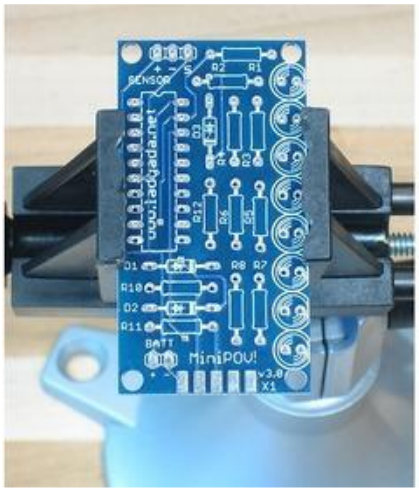
Solder up the kit

The first step is to solder the kit together. If you've never soldered before, check the tutorials on the preparation page ().



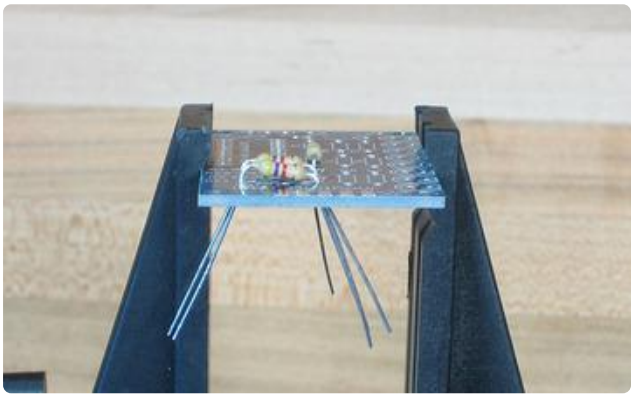
Check the kit to verify you have all the parts necessary.

Get your space set up with a good light, a vice or "third-hand" tool, diagonal cutters, and a soldering iron/solder.

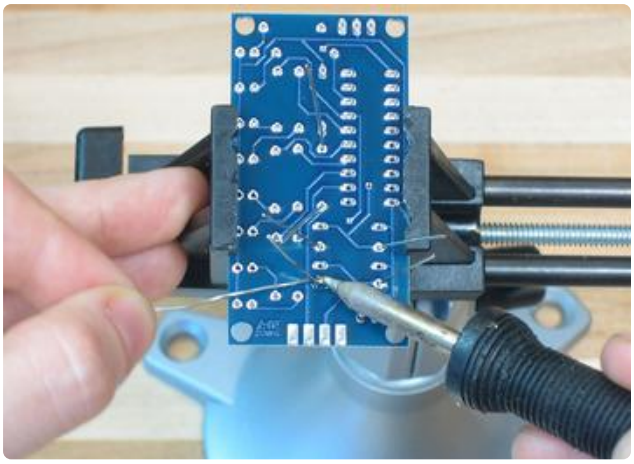


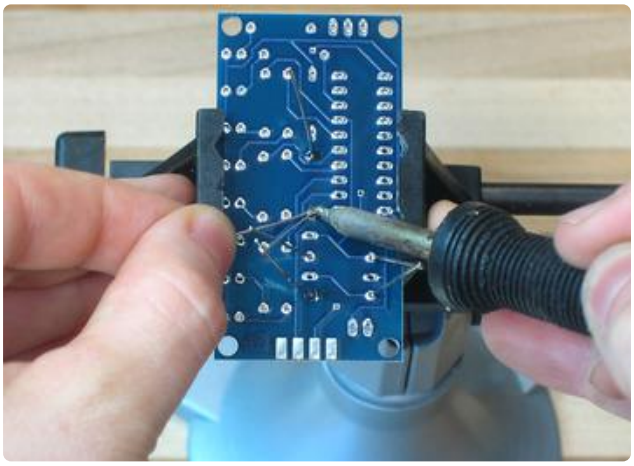
Put the circuit board in the vice, ready for soldering!

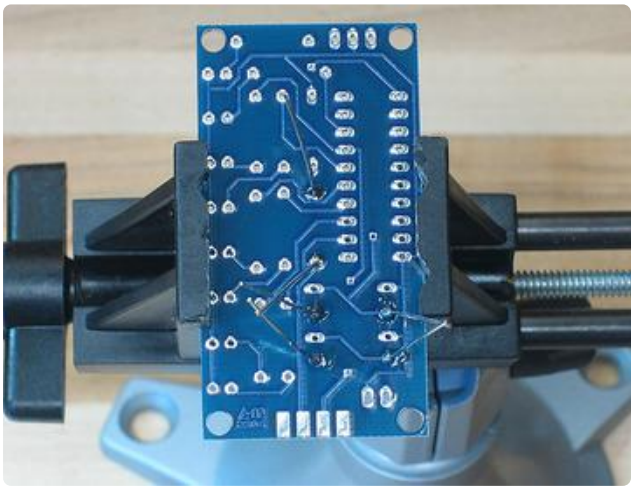
Place the 3 4.7K resistors as shown. Resistors are not 'directional' so don't worry which way they go in: it doesn't matter.



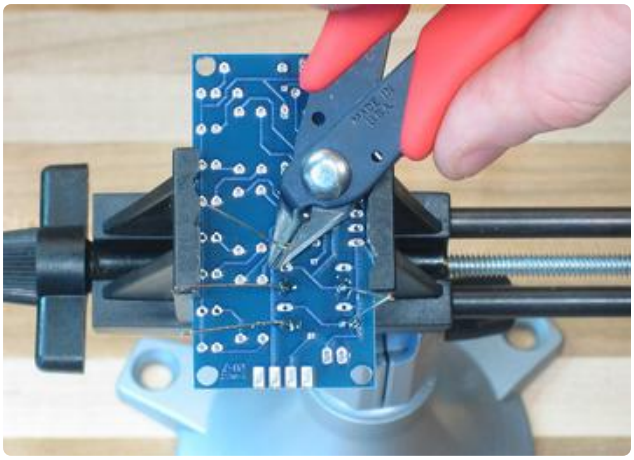
Bend the leads out so that when you turn the board over the resistors dont fall out.

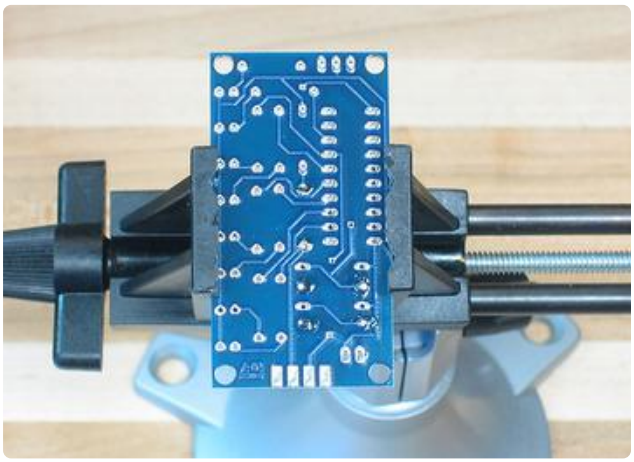




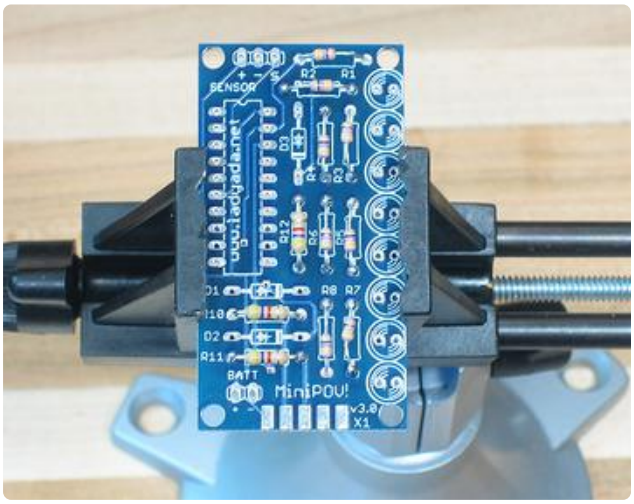


Make sure your solder iron is hot, hold it with your dominant hand. Use the other hand for guiding the solder in. Now steady your hands so that you can touch the very hot tip of the iron to the lead (resistor wire) and pad (circuit board hole) at the same time. Heat the two for 2 counts then dip the solder in, you should get a nice shiny blob as shown.

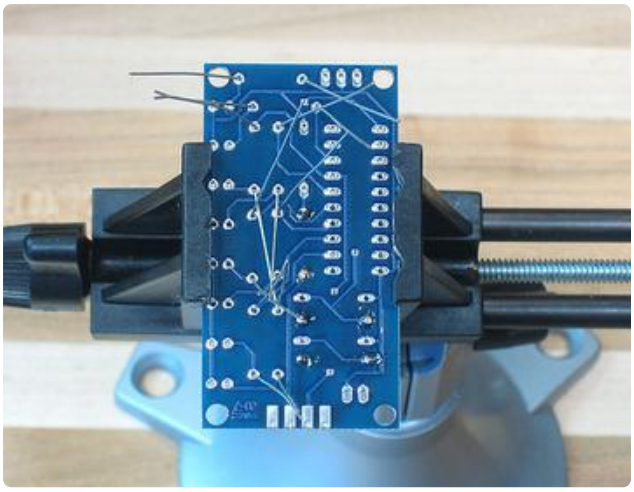


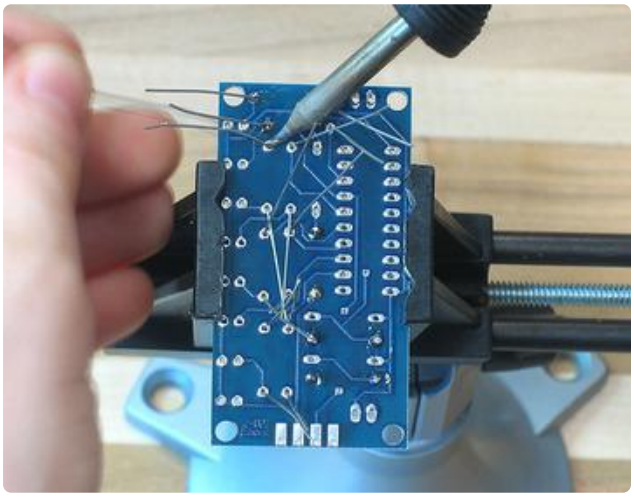


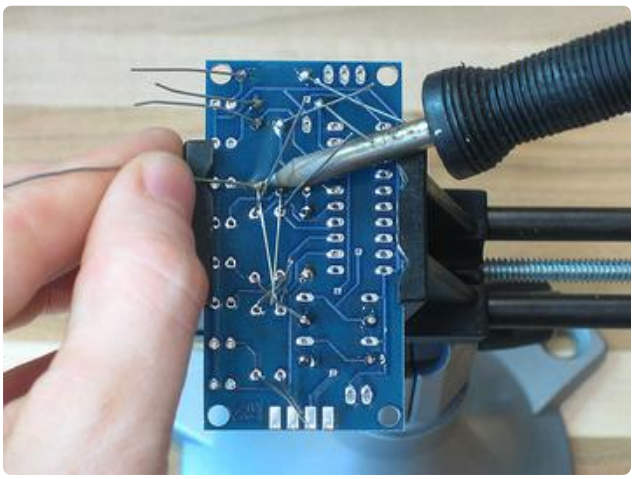
Cut the leads off so that only the blob remains. Be careful, the wires can fly out at you!



Place the 8 47 ohm resistors, just like the last time.

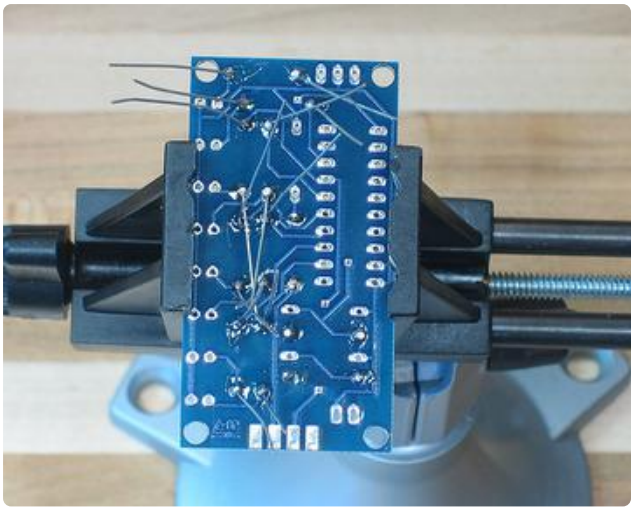


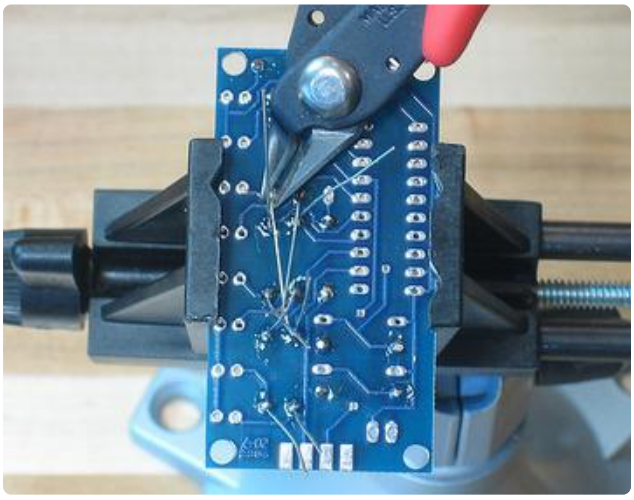


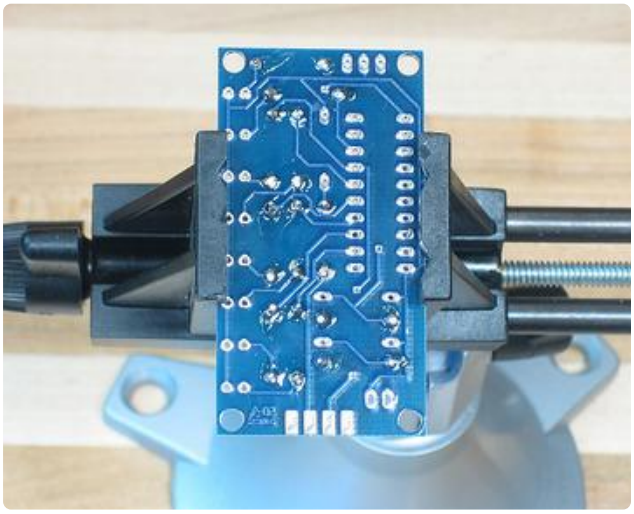


Turn the board over.

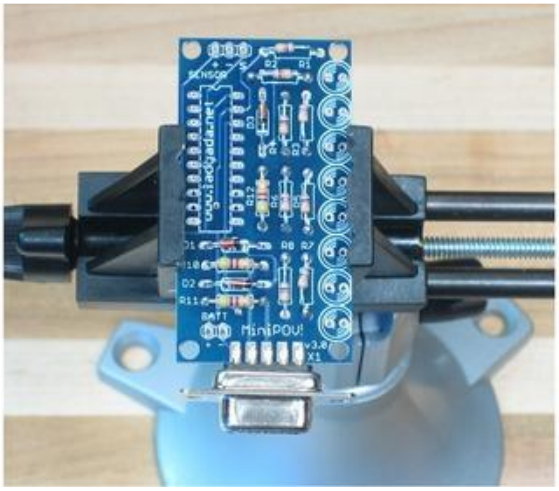
Solder the leads, clipping as you go if it's too clumsy to solder around the wires.





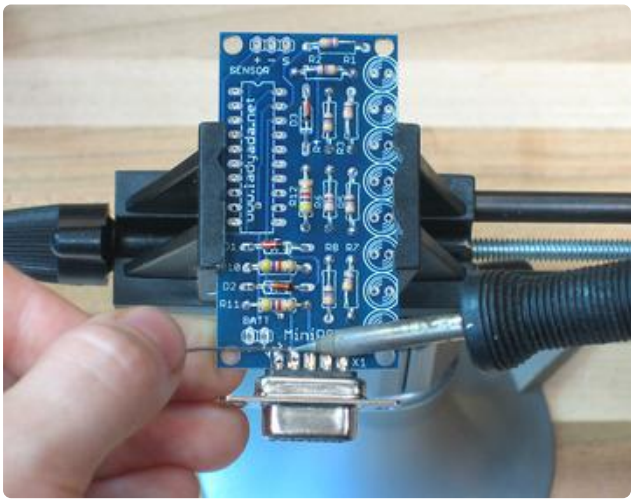


Clip the leads.

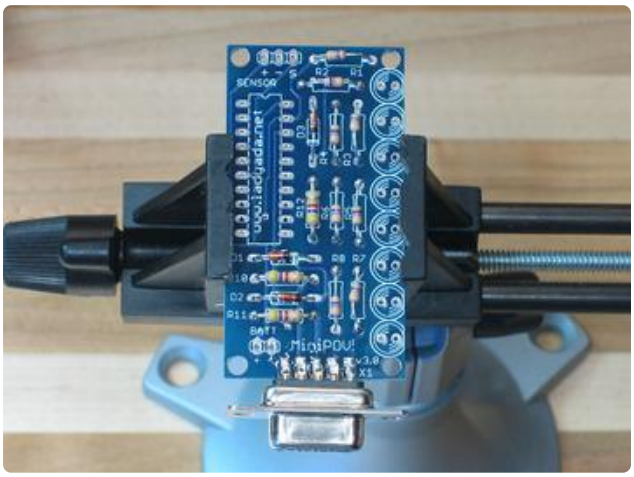


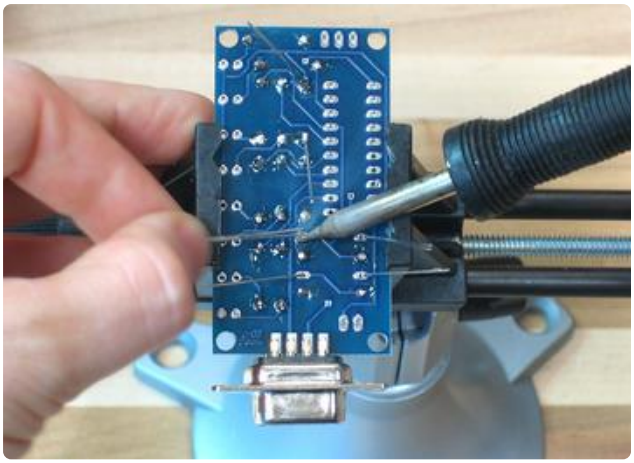
Place the three zener diodes and the serial port connector. The diodes are directional so don't put them in backwards. Note that there is a black stripe on the red glass, this stripe matches the white stripe on the silkscreen picture of the diode.

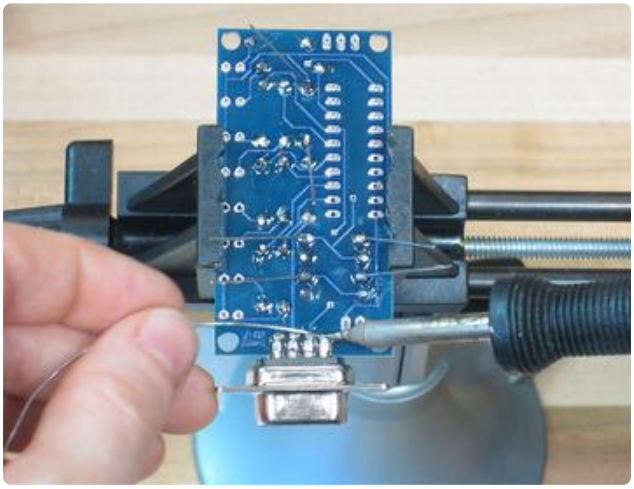
The serial port connector goes on only one way but it will be pretty obvious (because the two sides are different). The connector slides onto the end of the board and sandwiches it.

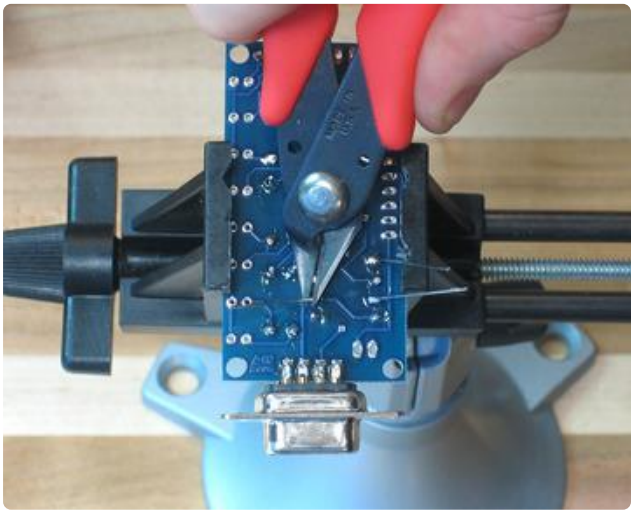


Start by soldering on the serial port connector. Make sure that you're actually soldering the pins to the circuit board (solder underneath) rather than just filling the pins with solder.

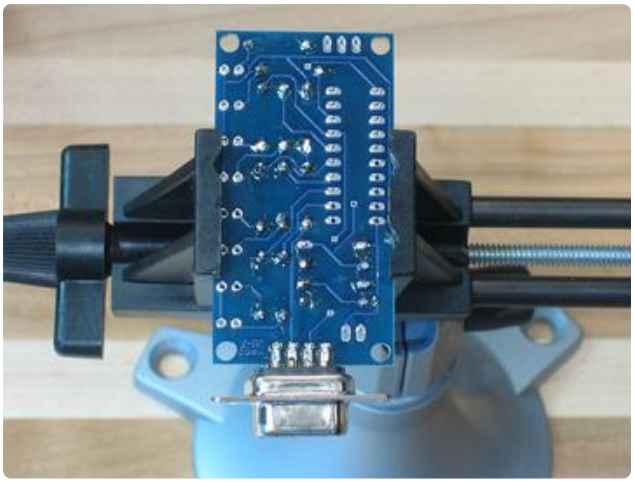


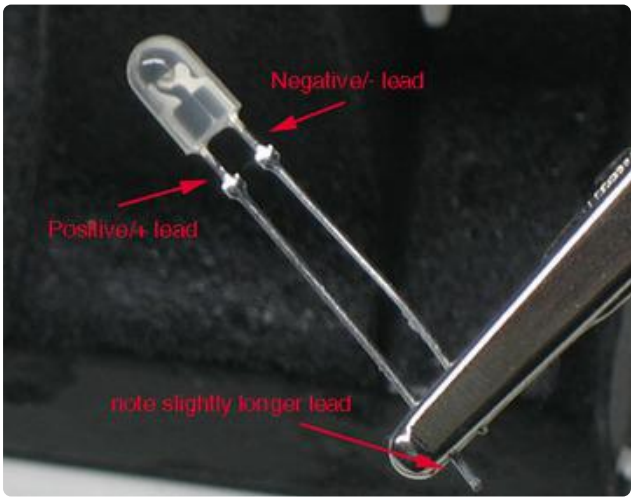


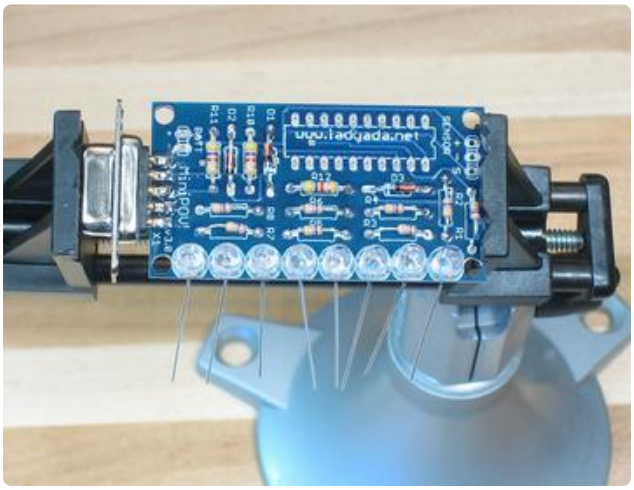




On the other side, solder and clip the diodes and solder the other 4 pins of the serial port connector.







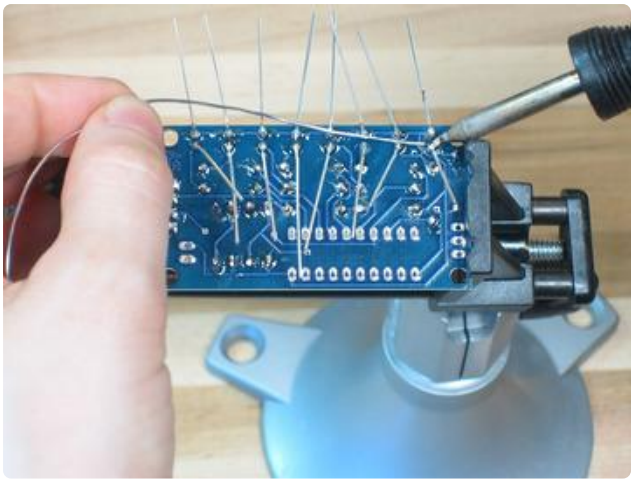
For the next step, you will place the 8 LEDs. LEDs have a 'direction' so if they're backwards they don't work. There are three ways to tell the direction.

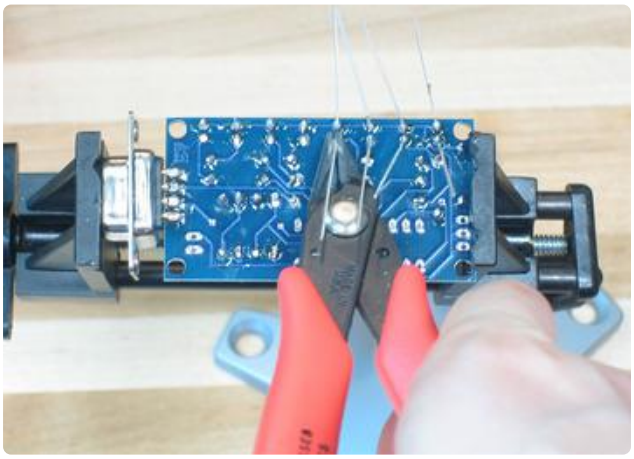
One is that the postive lead is longer than the negative one.

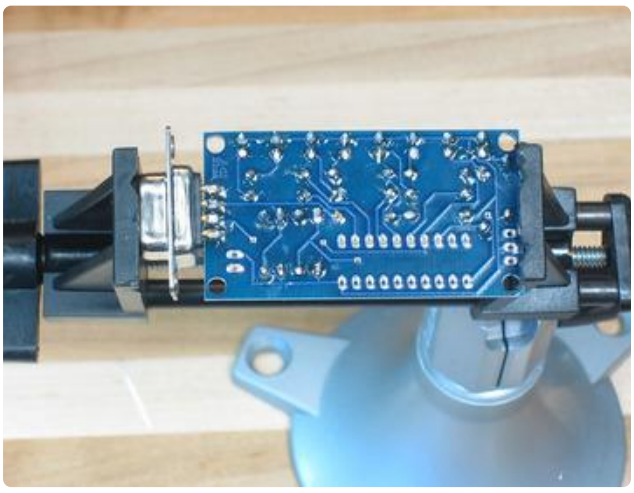
Second, the negative side inside the plastic is larger and has a 'cup'.

Third, the negative side has a flattened section (feel it with your fingers).

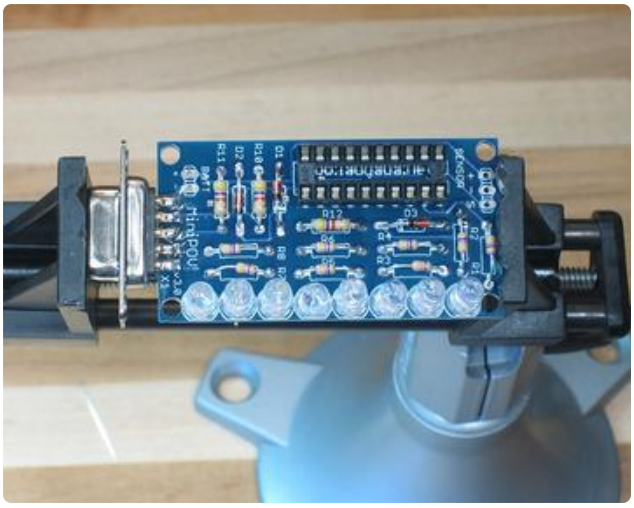
Place the 8 LEDs so that the negative side is nearest to the edge.

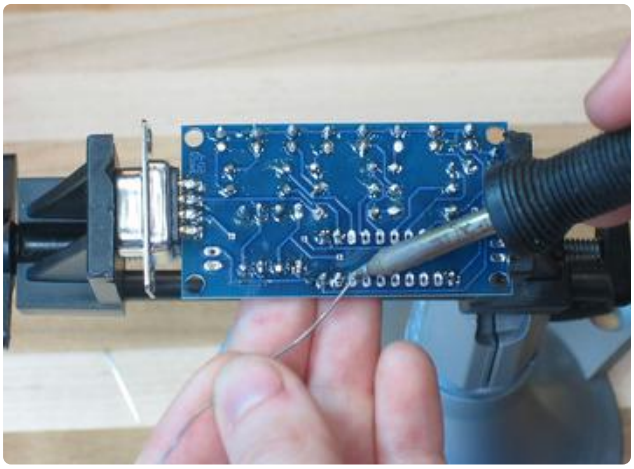






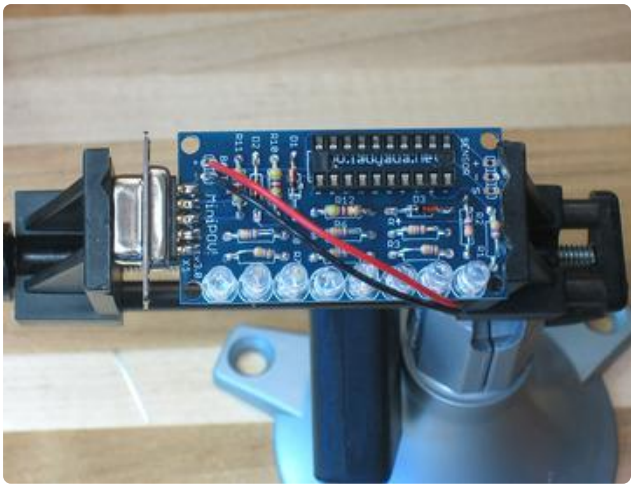
Turn the board over and solder in the LEDs, then clip the leads.

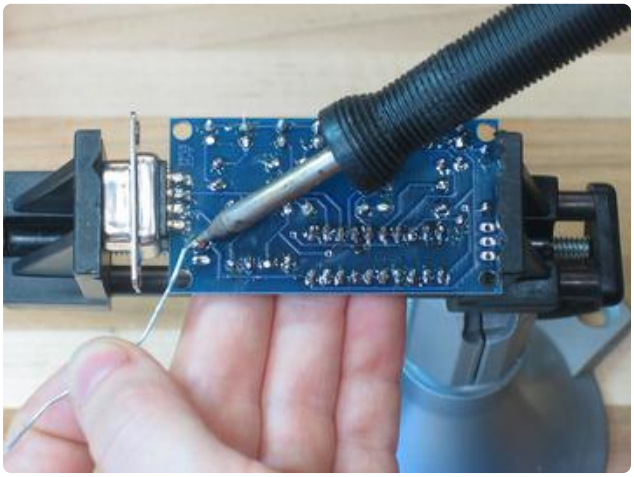




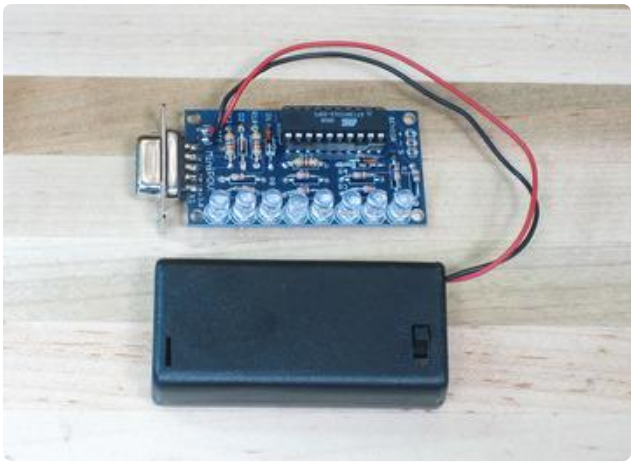
Next, place the microcontroller socket. Note that there is a little notch at the top, this tells you which way to put in the microcontroller. There is also a notch in the silkscreen image so match the two sides up, it will make it less likely for you to put the microcontroller in backwards (which could damage it).

When you turn the board over, hold the socket with a finger because it will fall out easily.





Solder in the two battery holder wires, red is + and black is -



Place the microcontroller in as shown and put in 2 AA batteries, be sure to turn it on with the switch!

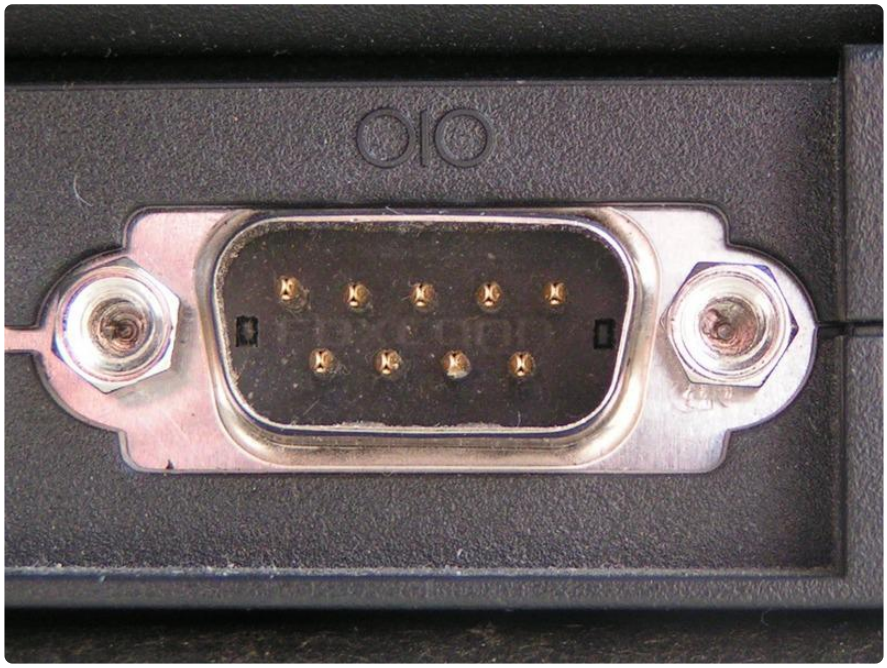
By default, the LEDs light up in sequence. If you bought a special version of the kit, it may have a preprogrammed image.

Software

Overview

Now that it's assembled, you can reprogram the chip with persistance of vision code, to display custom messages & images!

To program the chip you will need a computer with a serial port (basically, any PC) or a USB/Serial converter for computers without serial ports.



This is what a serial port looks like

I have tried KeySpan brand converters and the GWC AP1100 (PL-2303 chipset, $17 from NewEgg () or Jameco ()) with suceess. If you're using that or some other PL-2303 chipset ones on MacOS X then be sure to install this version () instead of the one that comes on the driver CD. There's also been reports of success with the FTDI chipset, this one is $11 from tigerdirect (). Note that not all converters work (especially the real cheap ones) so if you own one already, try the one you have (and let me know!) but there's virtually no way for me to 'make it work with X brand that I have'.

Note: if you are trying this out with a new microcontroller (i.e. not from a kit) you'll need to burn the fuses first, otherwise it won't work right and might not be programmable. Run make burn-fuse before you type in make program-minipov, while the minipov is powered and connected to the serial port. You might have to do it a couple of times if it's failing.

Note 2: When plugged into the serial port, the LEDs may light up oddly or dimly, this is normal.Still, the minipov3 must be powered on to be programmed.

Right now there's thorough instructions for:

- Windows ()
- Linux & other unix machines ()
- Mac OS X ()

Windows

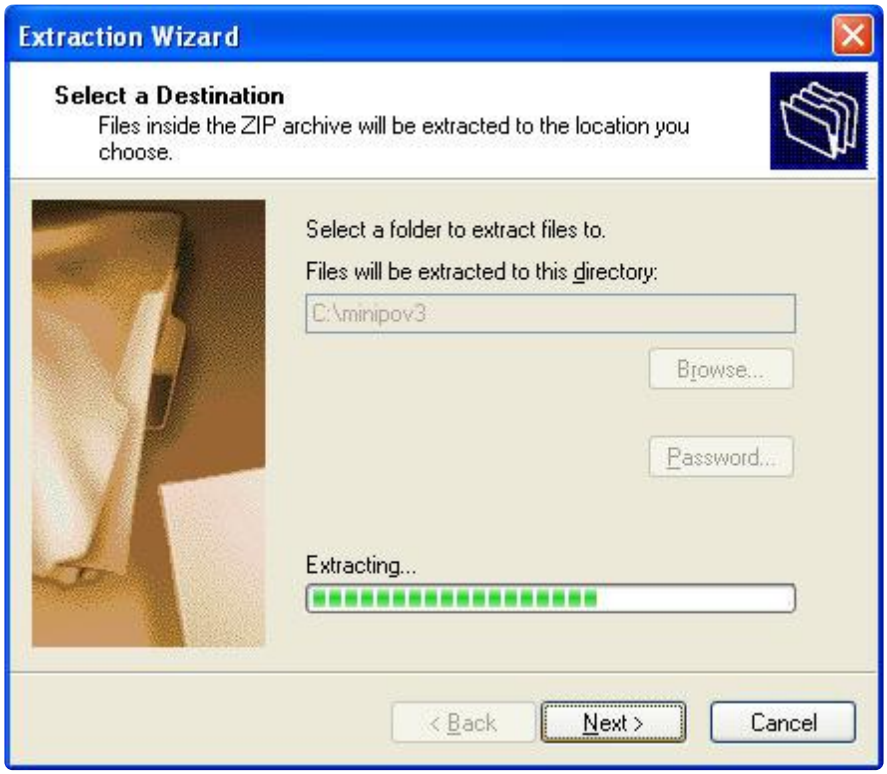
First, you'll need to setup WinAVR. Step by step instructions are here (). Once you're done with that, come back here and do the remaining steps.

If you're using a USB to Serial converter cable, you'll need to make some small modifications to the WinAVR package to support the MiniPOV3:

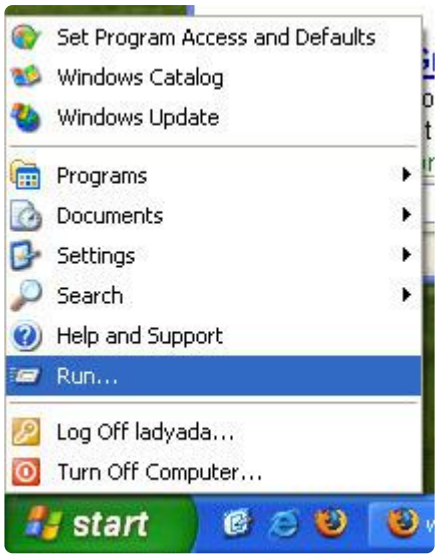
1. Download the zip file of the modified AVRDUDE software ()
2. Uncompress it onto the desktop
3. Open up the folder C:\WinAVR\bin (or wherever WinAVR was installed, perhaps C:\WinAVR-2008-01-06\bin)
4. Rename the two files avrdude.conf and avrdude.exe in C:\WinAVR\bin to avrdude-backup.conf and avrdude-backup.exe
5. Copy the two files avrdude.conf and avrdude.exe from the uncompressed folder in step 2 into C:\WinAVR\bin

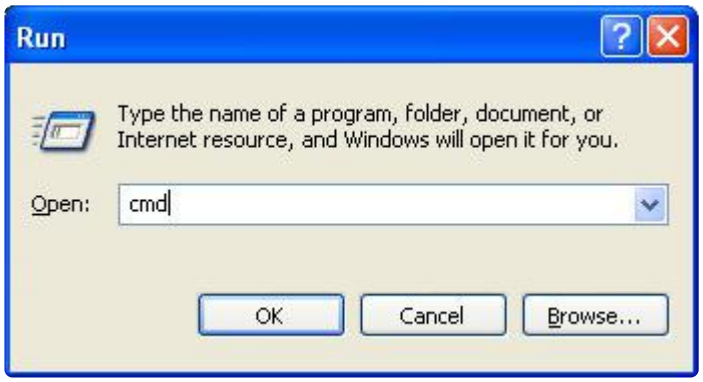
Now download the zip of example source code () for the Minipov3

Uncompress it somewhere convenient, the C:\ drive is a good place and is short enough to type in the command line.



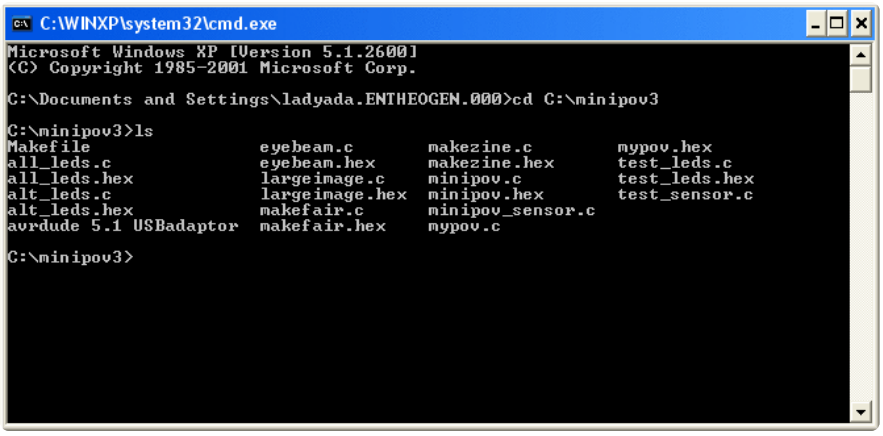
Open a command window





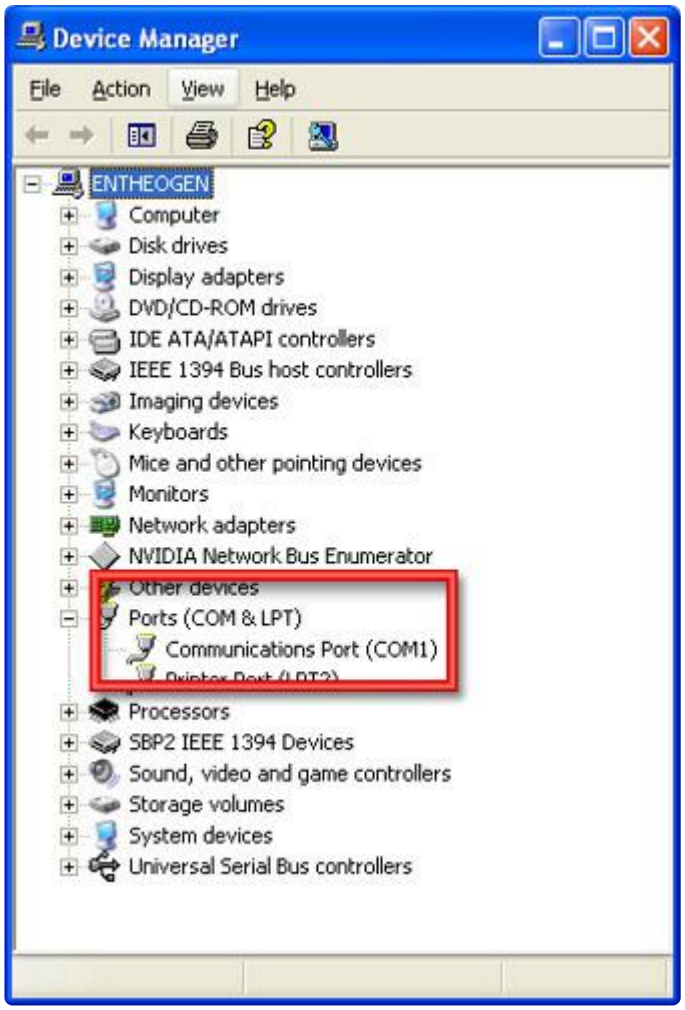
And cd (change directory) to the directory where you expanded the source code files (here, I expanded it into C:\ so the files are in C:\minipov3\ note the quotes allow me to specify a name with a space in it, otherwise it will get confused)

Run dir or ls and verify that there is a file called minipov.hex

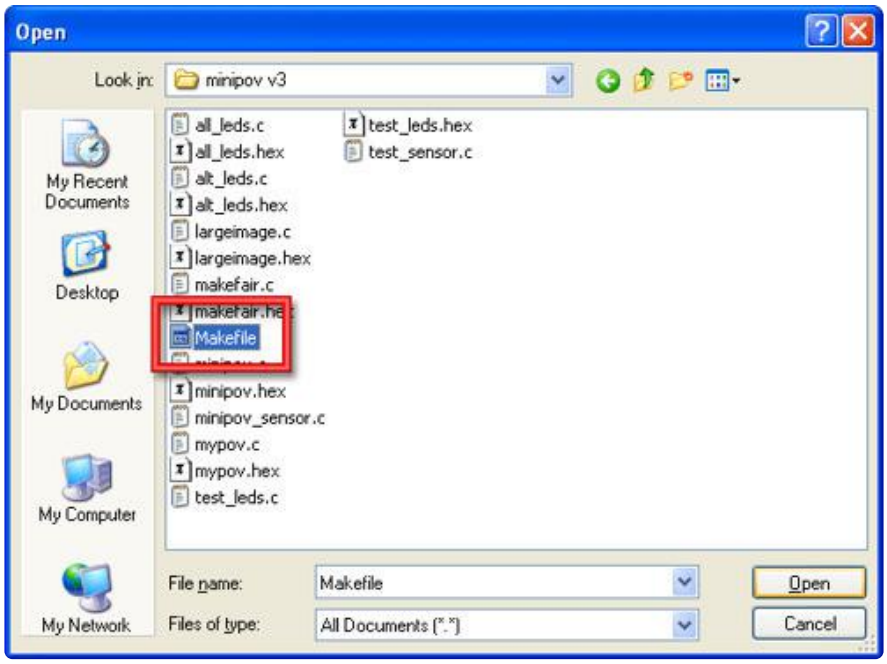


Plug in the MiniPOV into the serial port, and turn on the battery pack (it must be powered to be programmed even if it's looking like it 'turns off' when attached to the serial port).

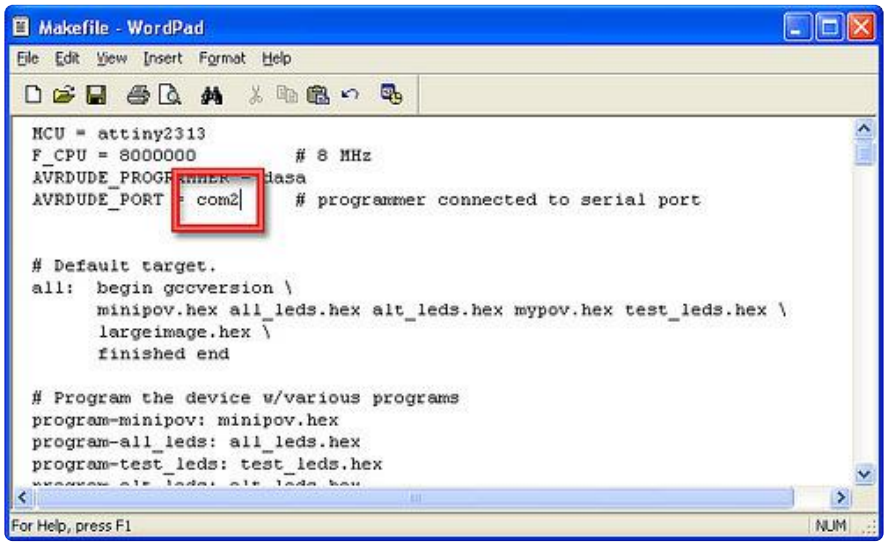
Now it's time to figure out what COM port you are using. By default almost all Windows computers have only COM1 but if you are using a USB adaptor or have a different configuration, you will have a different COM port. Open up the Device Manager (under the System control panel) and look under Ports.



If you are not using COM1, you will have to edit the file for the microcontroller programmer to tell it where to look for the Minipov3. Open the file named Makefile file in the C:/minipov directory with a program like Wordpad (included with windows) be sure to select "All Documents" type in the "file type" dropdown menu as Makefile doesn't end in .txt or .doc.

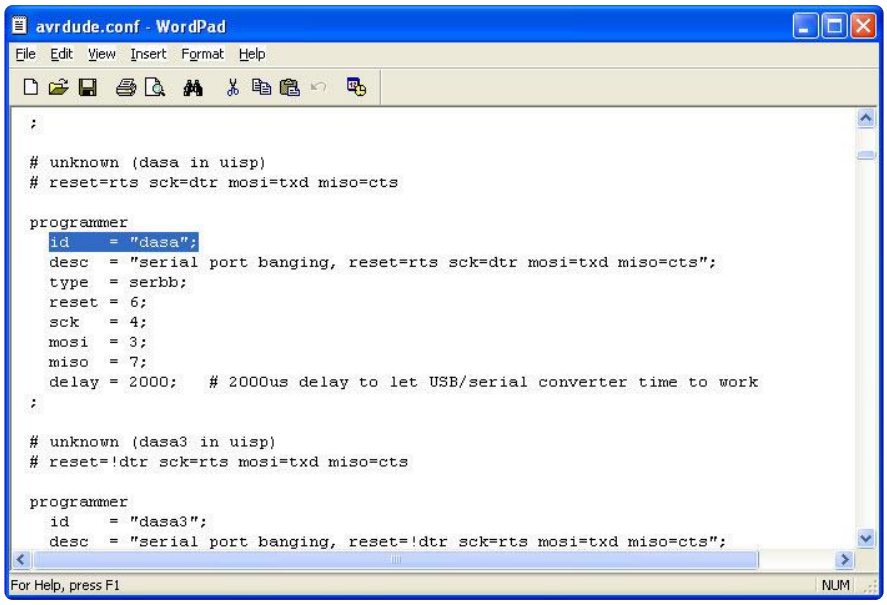


Change the AVRDUDE_PORT to COM2 or COM3 or whatever you're using.

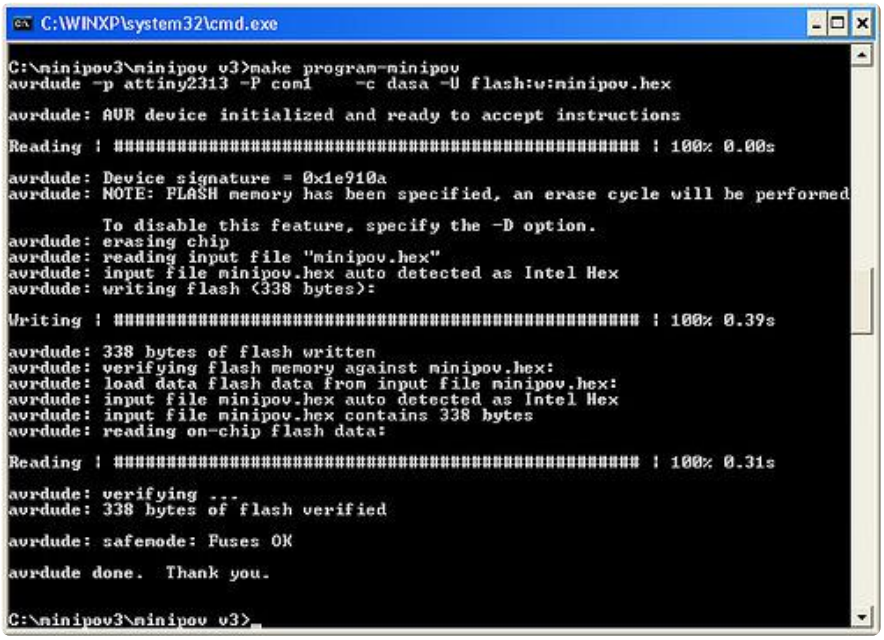


Save the file as a plain text file.

Next, if you're using a USB to serial adapter cable, open up the file C:\WinAVR\bin\avrdude.conf in Wordpad and search for "dasa" so that it will take you to the part shown below. Make sure you see a line with "delay = 2000" in it as shown below. That means that we are telling the programmer to go slow (wait 2 milliseconds between commands) because otherwise it gets confused. If you're not using a usb to serial converter, this line isn't necessary, but it doesnt hurt either.



Save the file as a plain text file. Now go back to your command window you had opened before. Run make program-minipov, this will start the programming procedure.



If you're using a USB/serial converter, it'll take a long time (up to 3 minutes!) to program the MiniPOV. You can adjust the delay to try and shorten the time. If you're getting errors try increasing the time in the avrdude.conf file. 2000 to 3000 is a good starting point, make it larger (5000 or more) to increase the delay.

If you get a bad response, such as

avrdude: initialization failed, rc=-1
Double check connections and try again

It means you probably have something connected up wrong. Check your soldering, are there any bridges or unsoldered parts? Are the diodes in correctly? Is the chip seated well? Is it turned on? Try connecting directly to the computer (not using a serial extension cable) or try a different USB to serial adaptor. Don't use -F to override the initialization check even though avrdude suggests it! It will not make things work, it will only make debugging more confusing!

Once you've gotten the programming procedure running, you can now create your own new and exciting messages ()!

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

MacOS X

First, you'll need to setup avr-gcc and related tools. Step by step instructions are here (). Once you're done with that, come back here and do the remaining steps.

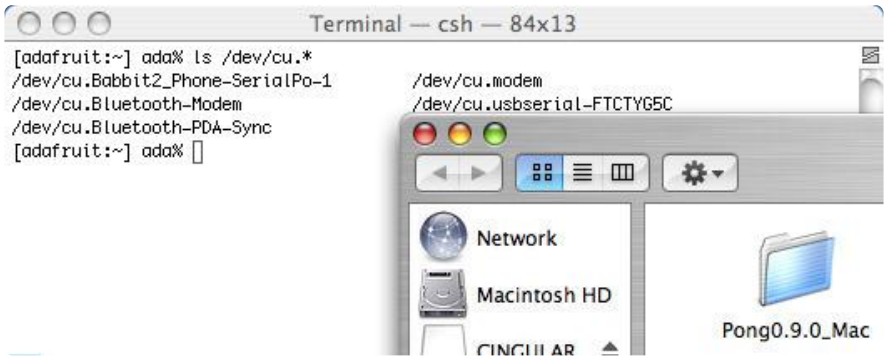
All the software is installed, now you just have to get the minipov3 firware!

Step 8. Setup MiniPOV firmware

Download the minipov3 firmware (), uncompress it and put the minipov3 folder into your Home directory.

Plug in your USB/Serial converter, making sure the driver is installed properly. If you're using a PL2303 chipset type adaptor, use this driver () which works much better than the ones often distributed with these adaptors.

Next you have to figure out what the name of the USB adaptor device is. In a Terminal window, type: ls /dev/cu.*
and look at the output. It should be something like /dev/cu.usbserial in this case its /dev/cu.usbserial-FTCTYG5C whatever that means.



If you're using an FTDI-chip based adapter, it will show up as /dev/cu.usbserial-FTCxxxx. If you're using a PL2303 chipset adapter, it'll show up as /dev/cu.PL2303-1B1

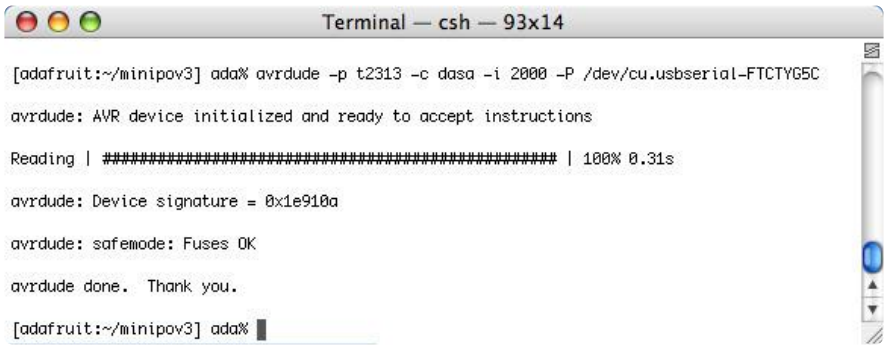
Step 8b. Download patched avrdude if necessary

For some reason, FTDI-chip based adapters need some special help programming these chips. You'll need to use a modified verson of avrdude so download the package from here (). Replace /usr/local/AVRMacPack/bin/avrdude and /usr/local/AVRMacPack/bin/avrdude.conf with the patched versions you just downloaded.

Step 9. Program some chips!

Power up your working MiniPOV and plug it into the adapter. Now in a Terminal window, type in avrdude -p t2313 -c dasa -P /dev/cu.usbserial-FTCYG5C

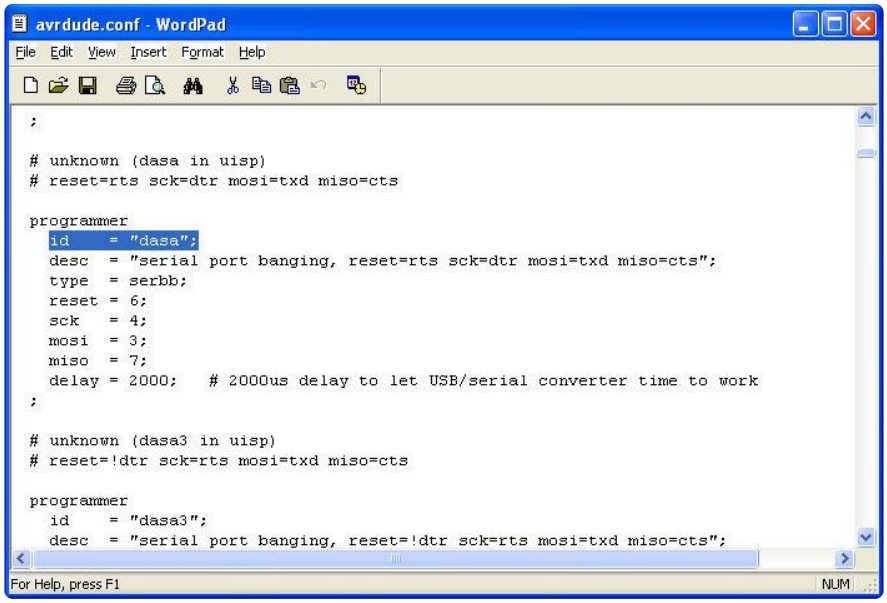
The -p t2313 indicates that this is a attiny2313 type chip
the -c dasa indicates its a serial port programmer
the -P /dev/cu.usbserial-FTCYG5C indicates where to find the USB-serial converter



You should get a similar output. If the device signature is wrong or if the chip didnt respond, or if you get a bad response, such as:

avrdude: initialization failed, rc=-1
Double check connections and try again

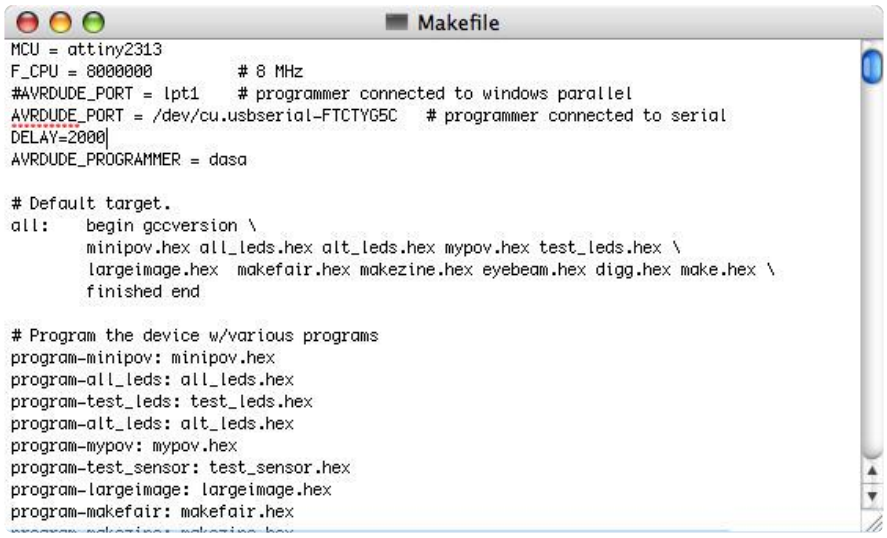
It could be that the delay is not long enough. Open up the file /usr/local/AVRMacPack/bin/avrdude.conf using TextEdit and and search for "dasa" so that it will take you to the part shown below. Make sure you see a line with "delay = 2000" in it as shown below. That means that we are telling the programmer to go slow (wait 2 milliseconds between commands) because otherwise it gets confused.



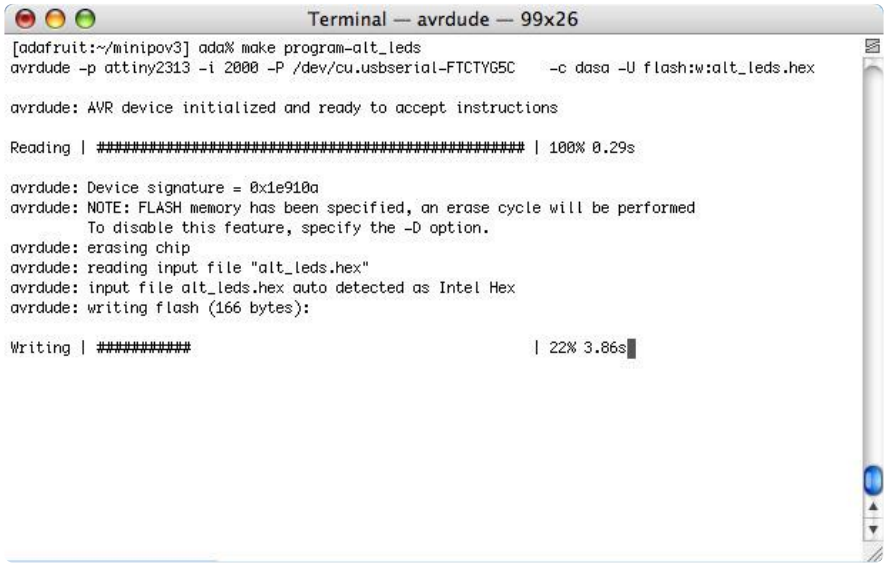
Try increasing the delay value to 3000 or 4000.

If it -still- doesn't work, it means you probably have something connected up wrong. Check your soldering, are there any bridges or unsoldered parts? Are the diodes in correctly? Is the chip seated well? Is it turned on? Try connecting directly to the computer (not using a serial extension cable) or try a different USB to serial adaptor. D on't use -F to override the initialization check even though avrdude suggests it! It will not make things work, it will only make debugging more confusing!

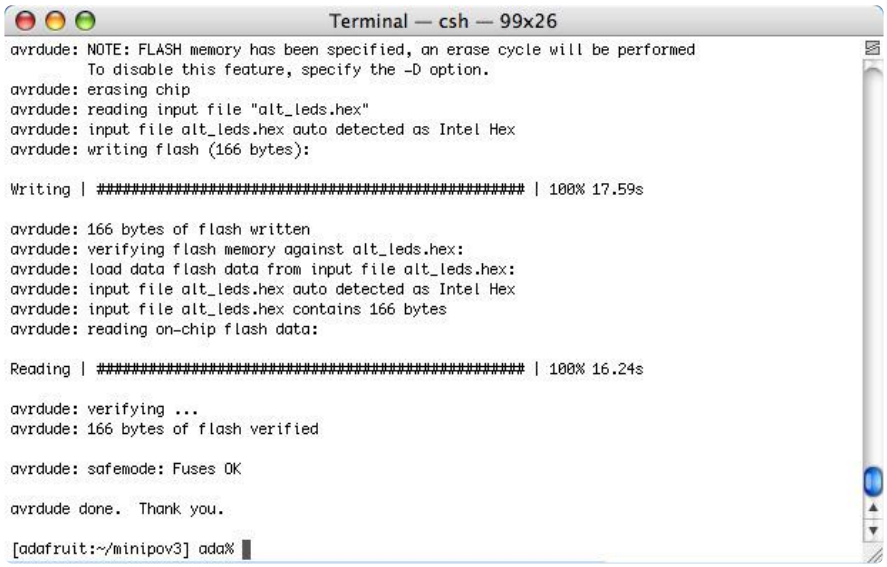
Next, open the Makefile in the minipov3 firmware directory with TextEdit and look at the top couple of lines:



The Makefile automates most of the typing, so things like the port, programmer type and chip are defined once. Change the AVRDUDE_PORT assignment from COM1 (a windows serial port) to the serial port you found earlier, like shown. Dont change anything else. Now save the file and go back to the Terminal.



Type in make program-alt_leds which will program the chip with the alt_leds.c program.



It should be successful. If it isn't, try increasing the delay until you get verified working results.

Once you've gotten the programming procedure running, you can now create your own new and exciting messages ()!

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

Linux (& other Unix-y machines)

Follow the instructions for getting avr tools set up from here. ()

1. Download the zip of example source code ()
2. Uncompress and cd into the source directory.
3. Run make and verify that avr-gcc was found and everything compiled all happy (there should be a bunch of .hex files now).
4. Plug in the MiniPOV into the serial port, and turn on the battery pack (it must be powered to be programmed even if its looking like its powered off of the serial port).
5. Edit the Makefile in the minipov directory to change the AVRDUDE_PORT to /dev/cuaa0, /dev/ttyS0, /dev/ttyUSB0, /dev/cu.KeySerial1 or /dev/cu.usbserial or whatever you're using. (Check your distribution docs for info on what the serial ports are called!)
6. Run make program-minipov, this will start the programming procedure.

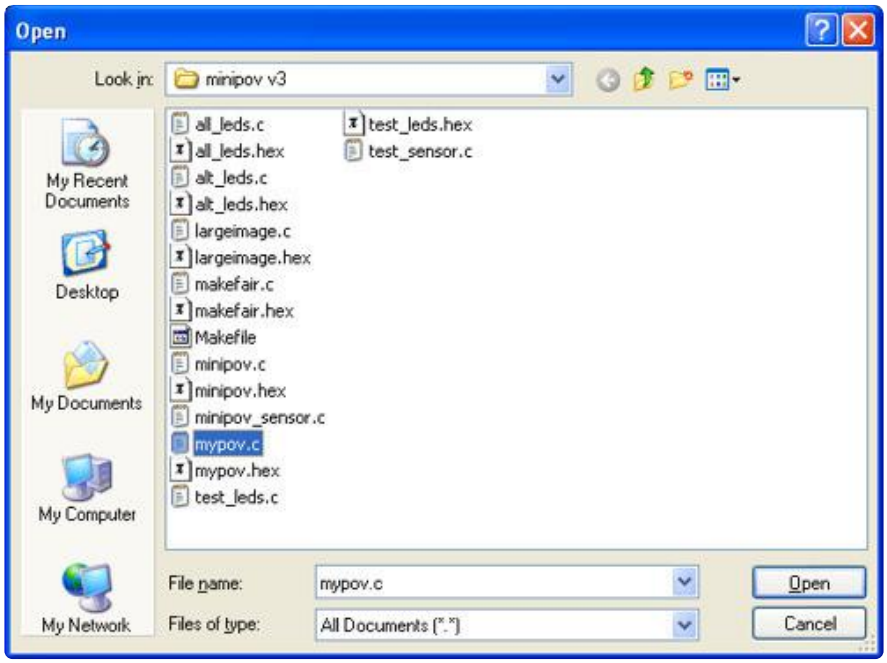
(You should cheat and look at the screenshots below for Mac OS X as they are nearly identical to Linux/Unix)

Customize

Programming/Changing the Image

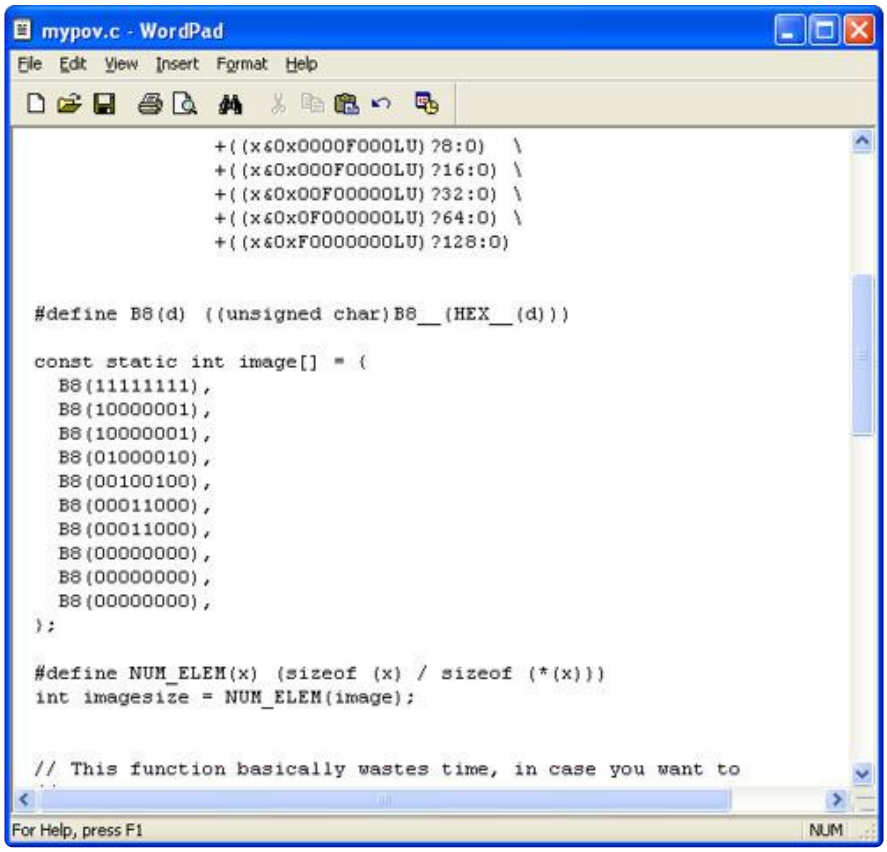
Innovate

Now it's time to put a custom image into the POV toy. Using your favorite text editor, for example, Wordpad, open up the file mypov.c in the minipov3 firmware directory you extracted earlier.



Near the top of the file is a table called image[]. Each entry in the table is an 8 bit number (which is displayed in binary: B8(00000000) through B8(11111111). The 'B8()' just indicates to the compiler it's in binary not in decimal). When the microcontroller starts up, it basically goes through the table and uses the value to determine whether each LED is on or off. For example, if the value is 01010101, then all the odd LEDs are on. If the value is 00000011, then the bottom2 LEDs are on.

After drawing the desired design on graph paper, go through each line and edit imag e[], adding or removing lines to make the table the right length. The table can be more than 500 lines long. (Hint: the easiest way I've found to do this is to copy enough 'B8(00000000),' lines to fit the size of your design, then enter in the 1's in insert/overwrite mode, which almost all text editors have). If this seems a little confusing, try opening up minipov.c to see how the minipov message is made.



Compile mypov.c by opening a command window as you did before, cd'ing to the minipov3 directory and typing in del mypov.hex (or rm mypov.hex) and then make mypov.hex. This should create a new mypov.hex which you can upload to the MiniPOV by typing make program-mypov just like the way you programmed in the minipov.hex code before.

Automated solutions!

Don't want to poke at the code with wordpad? Generate the table using Repulsor's handy website ()! Just copy it into your mypov.c table.

Or this one that will make largeimage.c with a custom text message (), just rename it to mypov.c when you save it to use the above instructions.

If you're running windows, Magician Soft has written their very own POV message generator ()! Awesome!

Design

Hardware Design

Microcontroller

A microcontroller is, essentially, a chip that has a program burned into its flash memory. When power is given to the device, it runs the program. A microcontroller can often perform many functions in parallel, such as reading sensors via an A/D, displaying on an LCD, processing data, interacting with other chips, etc. by using interrupts. They're a powerful electronics development tool because they are very inexpensive (on the order of a few bucks), have ROM, RAM, EEPROM, and a processor core already integrated, and can be programmed using popular languages such as C, BASIC and assembly. For this reason, I find them to be an excellent introduction to electronics for those who have some experience with computers and software.

The microcontroller used for the MiniPOV2 is an Atmel () 20-pin RISC device called the ATtiny2313. This chip was chosen because it is the cheapest one that has an internal oscillator (no external crystal/clock necessary) and enough I/O pins to give every LED a pin.

LEDs

LEDs are current driven. That is to say, one should design the circuit with a given amount of current going through every LED instead of deciding what the voltage across the LED is. Basically this means one needs to have a resistor in series with the LED, and the larger the resistor, the less current. Pretty much every LED function best with approximatly 5mA to 20mA of current running through them.

To calculate the resistor size, first one must determine what the (a) supply voltage is and (b) what the LED voltage drop is. For MiniPOV, (a) is 3V since each AA battery provides 1.5V. For (b), one looks up the 'forward voltage drop' in the datasheet. This value is dependant on what the LED is made out of, which also determines the color. Pretty much all red LEDs have a forward drop of 2V. Blue, white, UV and some green LEDs have a forward drop of 3.4V. Note that, in general, the supply voltage must be larger than the forward drop for the LED! Now subtract the two and divide by the current to get the resistor size in ohms: (3V - 2V / .02A = 50 ohms.) Since these resistors 'choke' off the amount of current going thru the LEDs, they are called 'choke resistors.'

There are a few easy to use LED calculators () online.

Batteries

The MiniPOV2 kit comes with a 2xAA battery case. The resistor values were chosen with the expectation that you'll use alkalines, but rechargables should be OK too. In fact, any high-capacity 2.5 to 5VDC supply can be used (but certainly not a 9V!). Of course, if the voltage goes up, the choke resistors for the LED have to be larger to keep the same brightness.

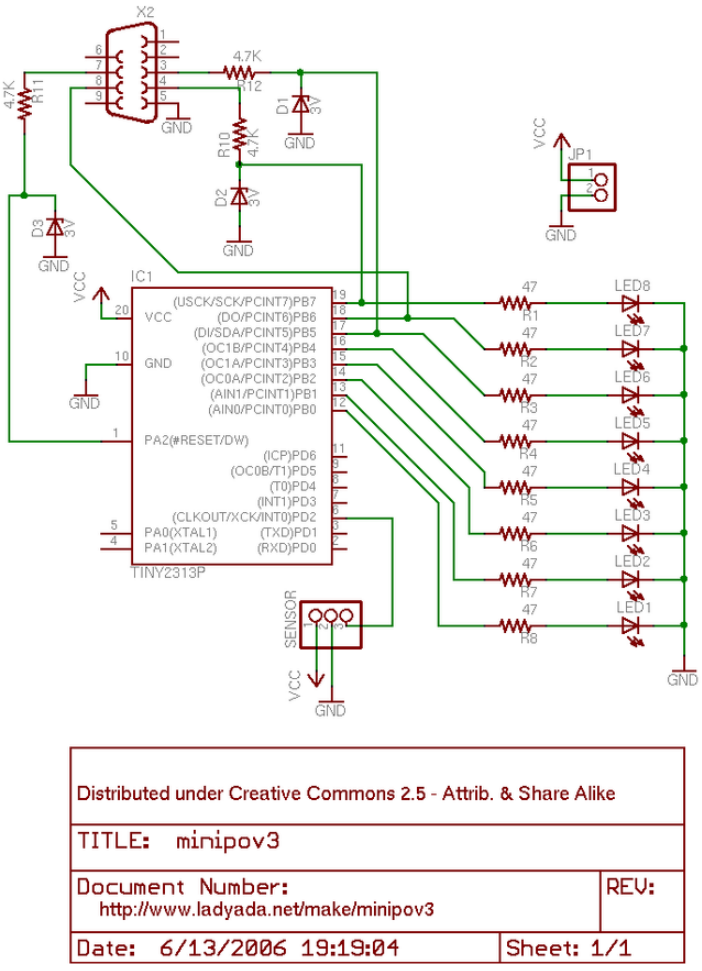
When programming, I suggest making sure you're using fresh batteries, preferrably alkalines. Sometimes the MiniPOV can act flakey if the batteries are low.

Serial Programmer

Previous versions of the MiniPOV used a parallel port plug to communicate with a PC for programming. The parallel port is well suited for this because it sends signals at 0-5V. The serial port, while more common, uses +-10V. This means you can't just wire it up directly to the microcontroller. Often times people use serial port converter chips such as the MAX232 or similar. Recently, people have figured out how to (ab)use the serial port into programming microcontrollers for cheap by "bitbanging" the modem control lines. To convert the voltages, 5.1V zener diodes are used to clamp the +-10V down to -0.5V to 5V. Since a lot of people have managed to use this method for a while, it seems like it's good enough for a kit. The particular wiring standard used is called "dasa" (direct serial) and can be used with uisp/avrdude/ponyprog.

The good news is that serial ports are more common than parallel ports and even when there is no serial port, you can always use cheap ($15) USB/serial adaptor cables. I've tested both Keyspan (TUSB chipset) and PL2303-chipset based adaptors. Note that if you're using the PL2302-chipset ones on a Mac you'll need to download/install an updated driver and not use the one that comes with the adaptor.

Schematic



Download

Files & Downloads

AVRDUDE programming software for USB-Serial adapters

If you have an onboard serial port (like a PC or some laptops) you can just use the default install of AVRDUDE. However, if you have a Mac or laptop or PC without a serial port and you want to use a USB/Serial adaptor then you should upgrade the default install with these files. Then you can specify a slow-down delay necessary to make it work with such adaptors.

- For windows, a zip file with a newer avrdude.exe and updated avrdude.conf ().
  Copy these into C:/WinAVR/bin (or wherever you installed WinAVR/AVRDUDE)
- For MacOS X/Linux/Unix (or windows if you want to install cygwin and compile it), a tgz file with updated source (). Just do a "./configure" then "make; make

install" as usual. (here is just the executable & updated avrdude.conf for MacOSX) ()

To download non-modified AVRDUDE programming software, please follow the instructions of the Software page ().

USB/Serial converter driver

If you're using the PL2303-chipset adaptor on a MacOS X machine, download this driver () and install it, NOT the one that comes with the device!

Firmware

Here is a zip file with all the firmware () (hex and source and Makefile)

Hardware

Here are the EagleCAD schematic and board () layout files!

Buy Kit

Buy Kit ()

Forums

Forums ()

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Adafruit:
20