Persistence of Vision (POV)

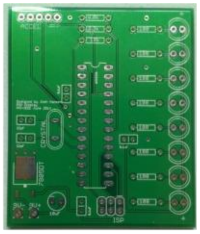
Display Build Instructions

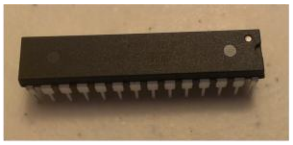
Josh Hansen

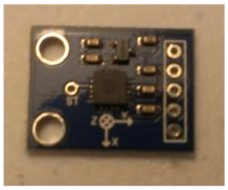
September 2014

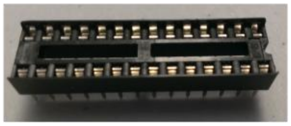
Step 1 - Component Check:

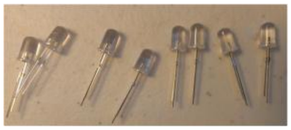
Ensure that you have all the required components:

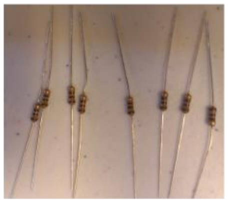

x1 PCB

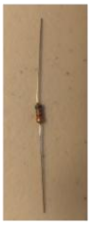

x1 ATMEGA48

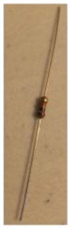

x1 Accelerometer

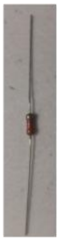

x1 28-pin socket

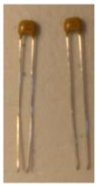

x8 5mm LEDs

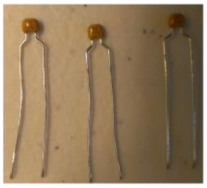

x8 100Ω resistors

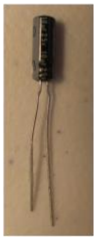

x1 6.8kΩ resistor

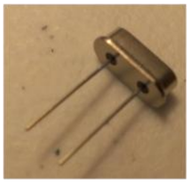

x1 10kΩ resistor

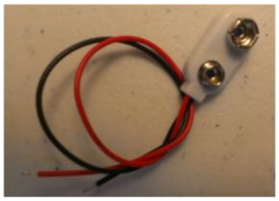

x1 3.3kΩ resistor

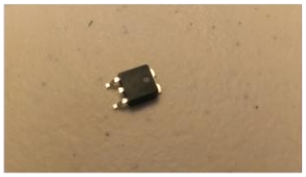

x2 22pF capacitors

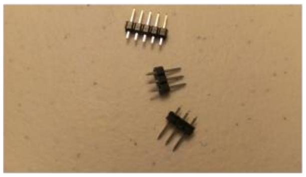

x3 0.1µF capacitors


x1 10µF capacitor


x1 16MHz crystal


x1 9V battery connector


x1 5V regulator

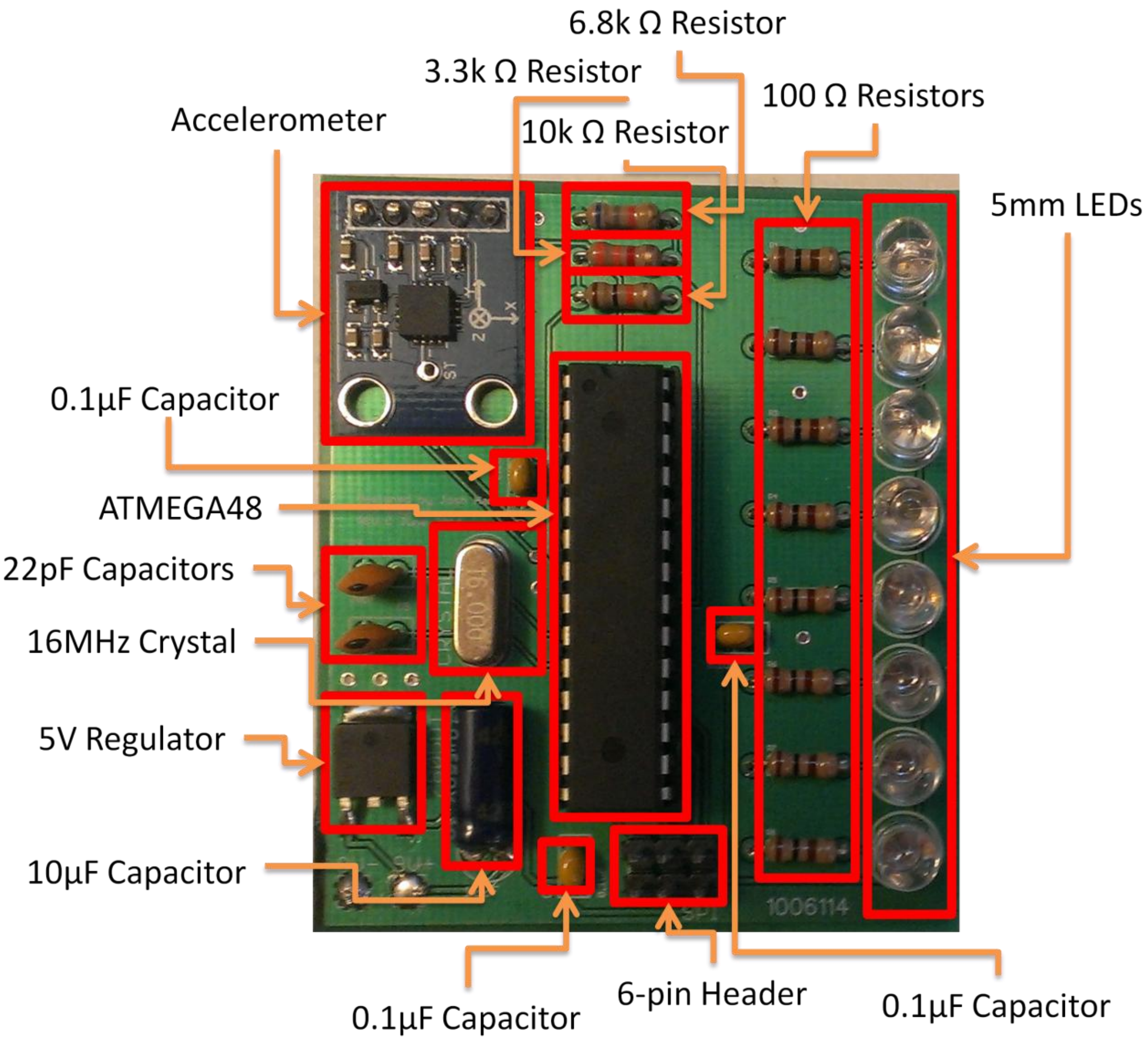

x3 Male headers

If for some reason you are missing any components, let an IEEE officer know. Note that a 9V battery is also required but not supplied in the kit.

Step 2 - Soldering:

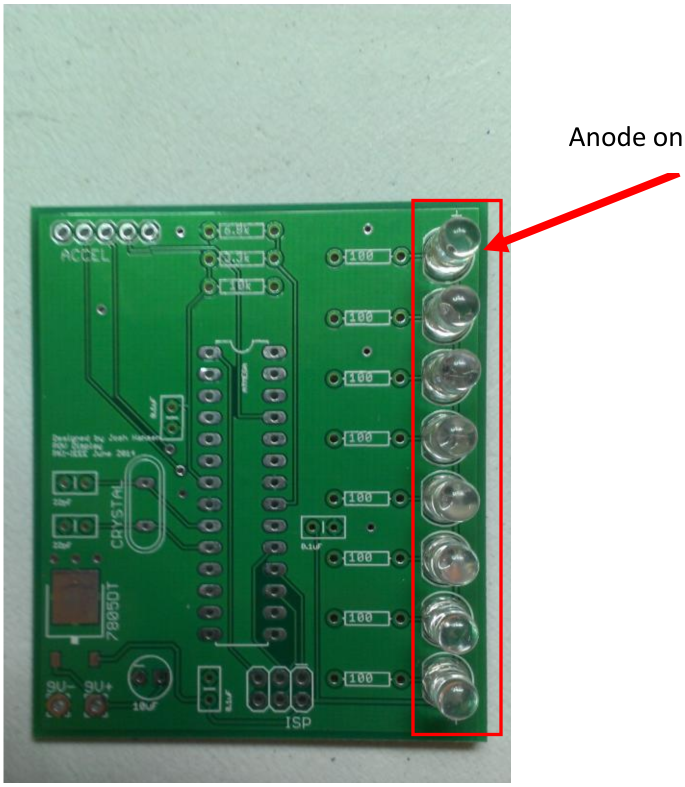
The components in this kit do not need to be soldered in any particular order and so the order the parts are soldered in this guide is arbitrary. However, the 5V regulator and the accelerometer are slightly more difficult to solder and therefore, if you do not have much soldering experience, it may be better to leave them until last.

If you have experience soldering and would like to just skip the step by step soldering instructions, use the following image as a guide:



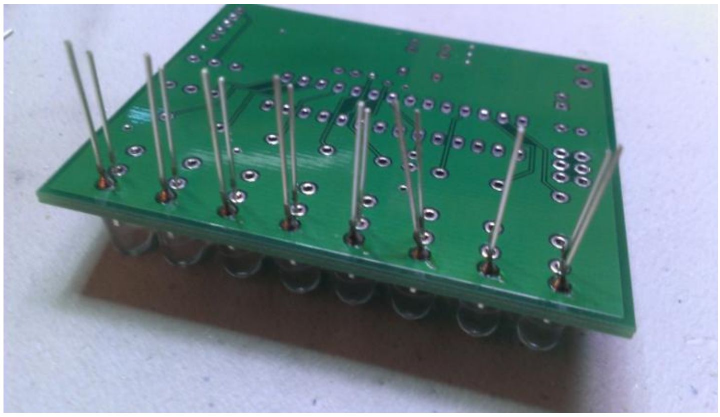
Soldering Step A - LEDs

Insert the eight LEDs into their respective holes on the right side of the PCB. Note that the anode (positive) lead of the LED must be inserted into the right-most hole. The anode of a LED is the longer of the two leads.



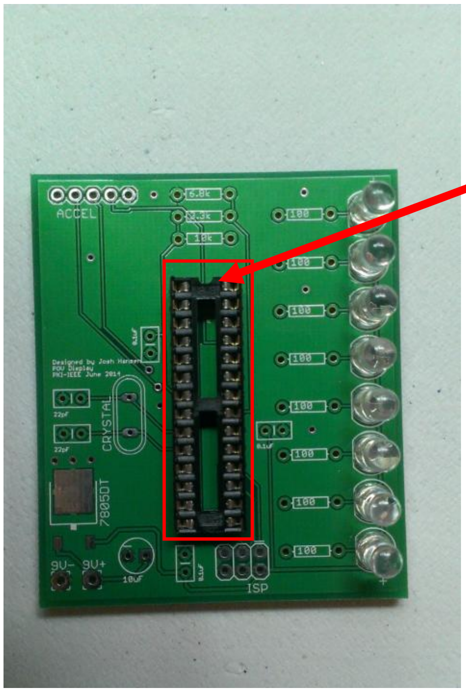
Anode on this side

To ensure that the LEDs are soldered flush with the PCB, solder only one lead of each LED. Then, while reheating the solder, push the LED flush with the PCB with your other hand. Once you're satisfied with the way the LEDs are mounted, solder the remaining lead of the LEDs.



Soldering Step B - DIP Socket

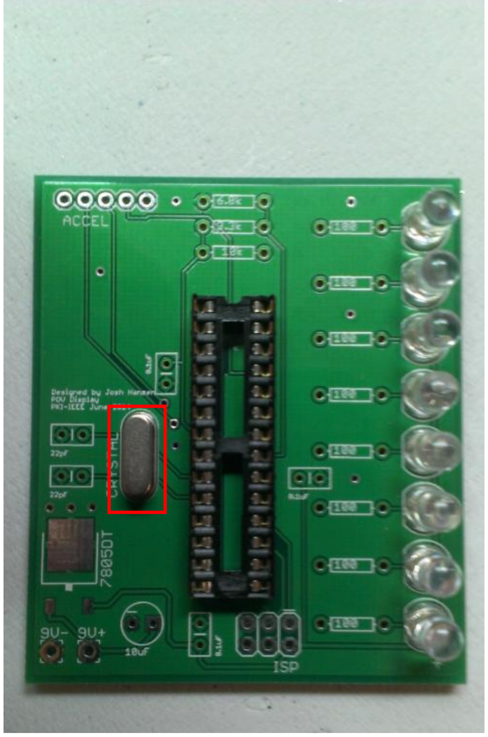
Insert the socket into the PCB with the notch at the top as show in the picture. Solder only one pin of the socket and, using the same method as the LEDs, push the socket flush with the PCB as you heat the solder. Once you are satisfied with the way the socket is mounted, solder the remaining 27 pins.



Notch at top

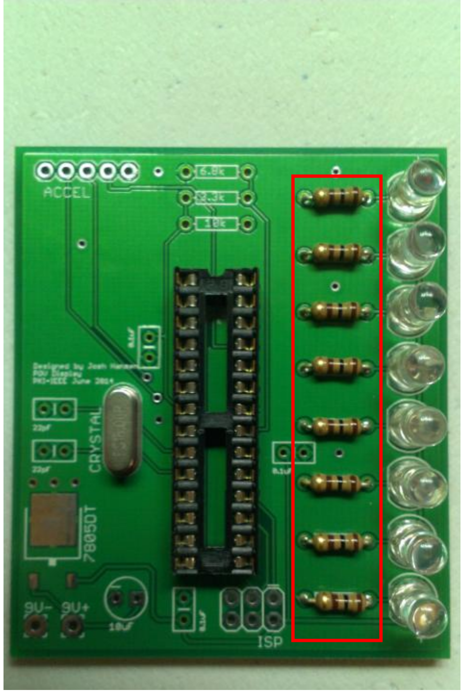
Soldering Step C - Crystal

Insert the crystal into the PCB as shown in the image and solder. Orientation does not matter as the crystal is not polarized.



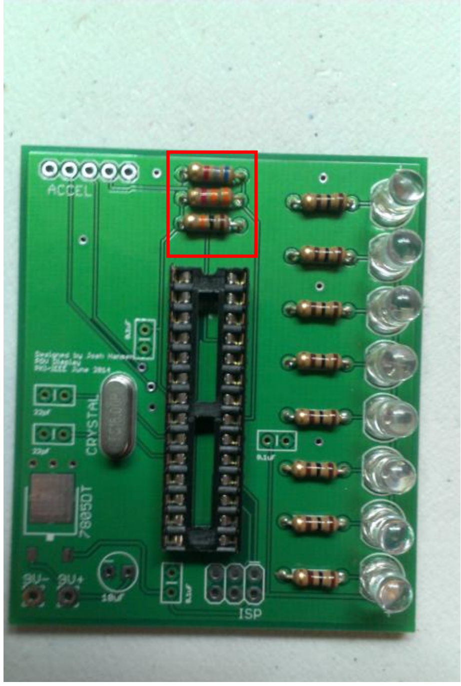
Soldering Step D - 100Ω Resistors

Insert and solder the eight 100Ω resistors into their respective hole on the left of the LEDs. Orientation does not matter.



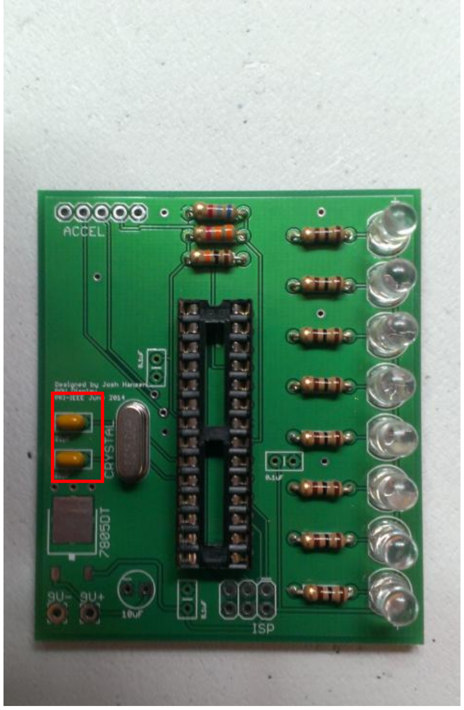
Soldering Step E - Remaining Resistors

Insert and solder the three remaining resistors into their holes above the DIP socket. Refer to the silkscreen or the image at the beginning of this guide for reference.



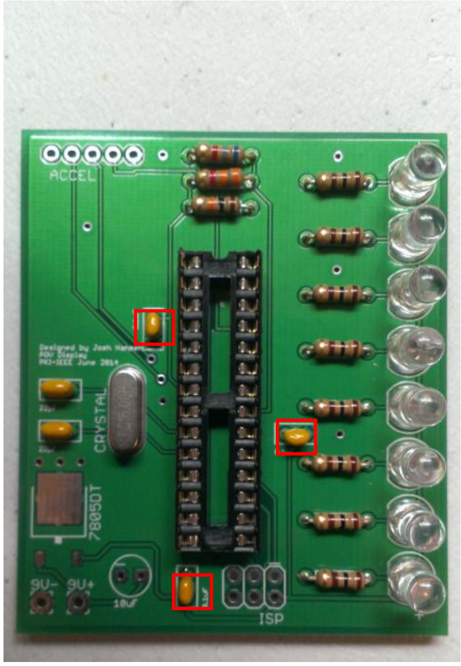
Soldering Step F - 22pF Capacitors

Insert and solder the 22pF capacitors into their holes to the left of the crystal. These capacitors are not polarized and so orientation does not matter.

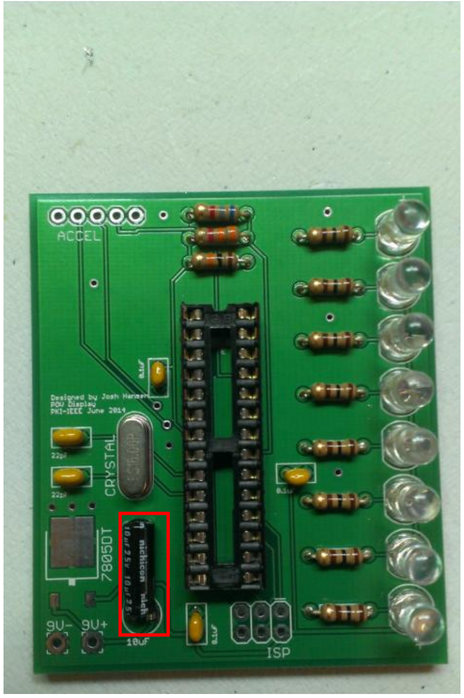


Soldering Step G - 0.1µF Capacitors

Insert and solder the three 0.1µF capacitors into their holes. Refer to the silkscreen or the image at the beginning of this guide for reference. These capacitors are also not polarized. If you want the capacitors to sit flush with the PCB, straighten out the leads before inserting them into the holes.



Soldering Step H - 10µF Capacitor

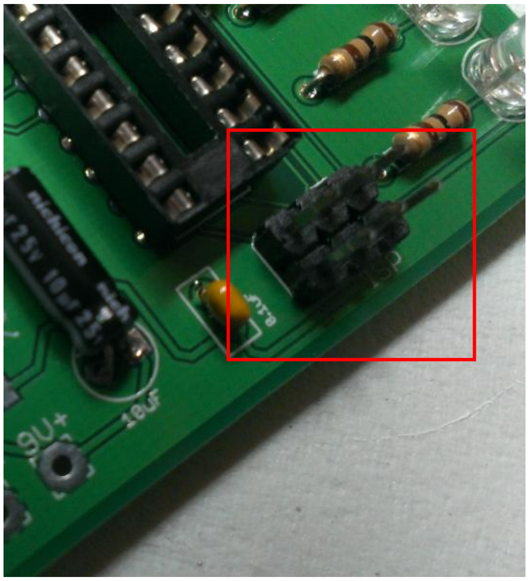


The 10µF capacitor is polarized and so the ORIENTATION DOES MATTER.

Insert the longer of the two capacitor leads into the right hole. The negative hole for the capacitor is marked on the silkscreen. Before soldering the capacitor, bend it forward and down so it lays flat against the PCB as shown in the picture.

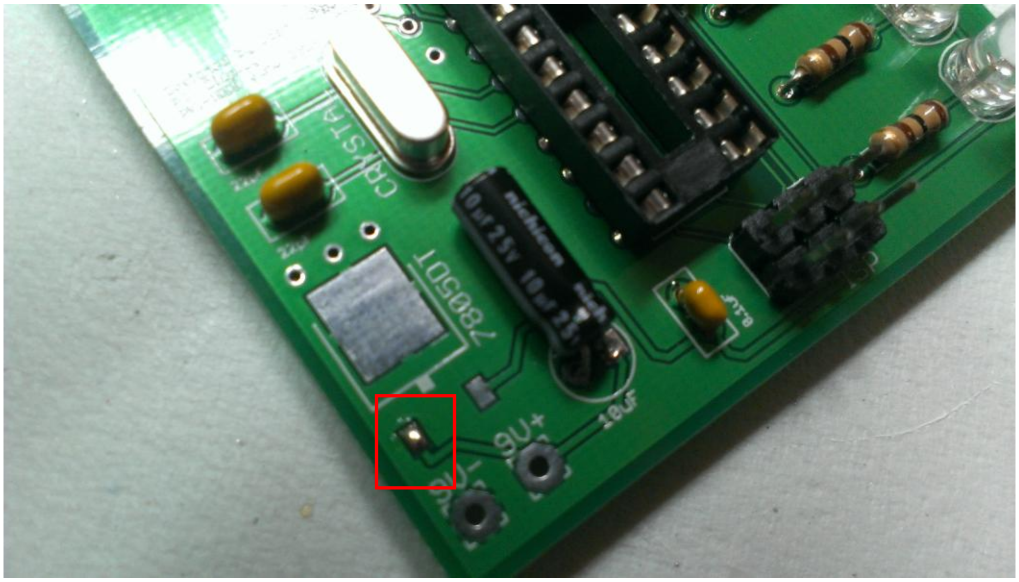
Soldering Step I - 6 Pin Programming Header

Insert the two 3 pin headers into their holes below the socket. The headers will only align correctly one way so you may need to reinsert one of them. Solder one pin on each row and then make sure all the pins are aligned before soldering the remaining pins.

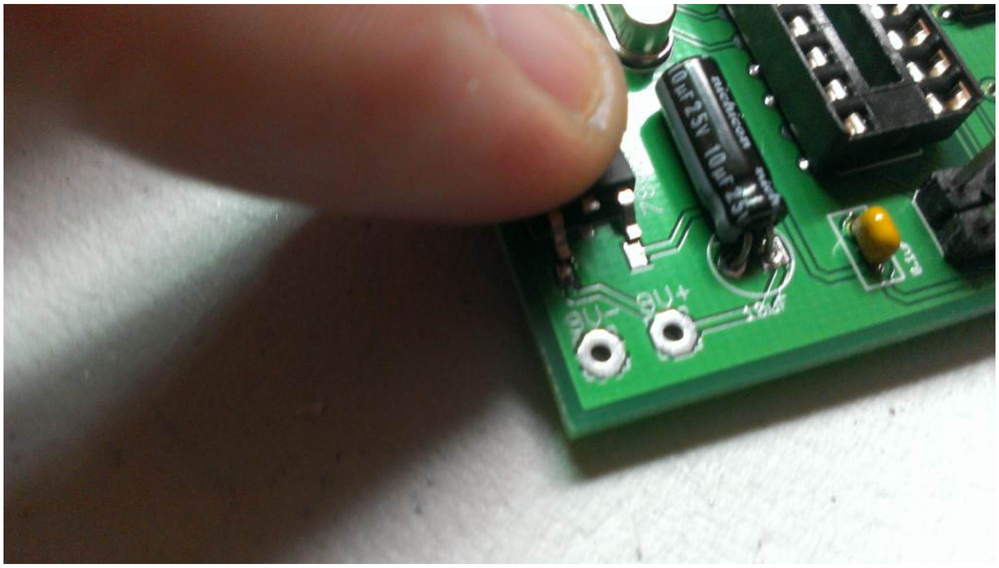


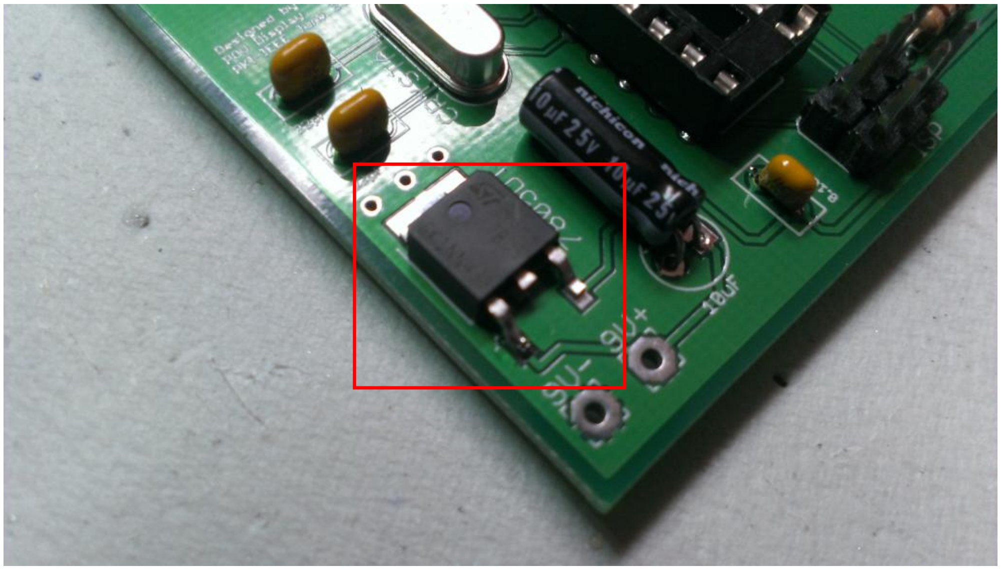
Soldering Step J - 5V Regulator

The 5V regulator is a surface mount part. Instead of having holes for leads, it has pads. Begin by first applying a small amount of solder to the lower left pad as shown in the picture.

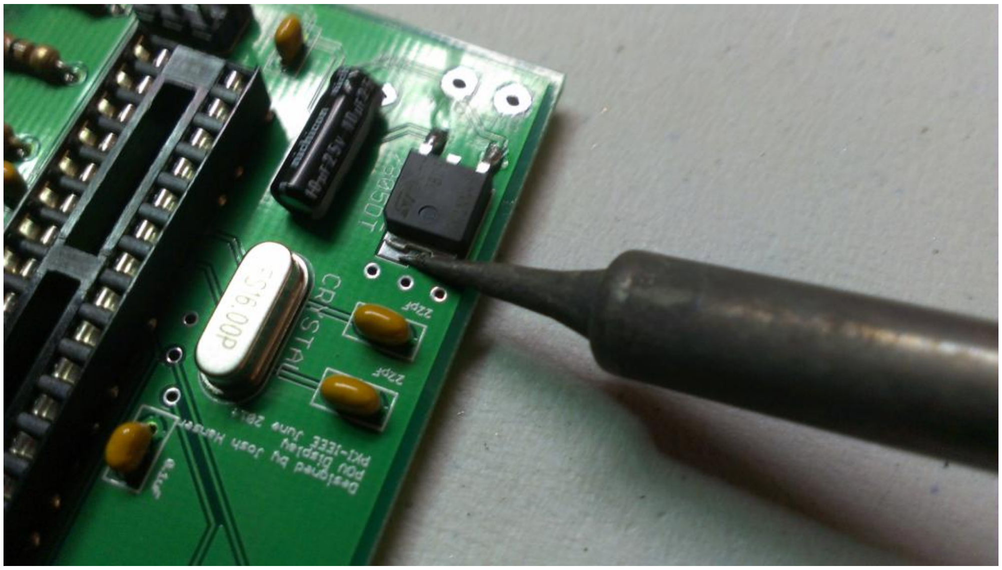


Lay the regulator on the PCB just above the solder you applied. Heat the solder into its liquid state and slide the regulator into place, moving the left lead of the regulator into the liquid solder. Do this quickly because the regulator will heat up and could begin to burn your finger. If it becomes too hot, let it cool and try again.

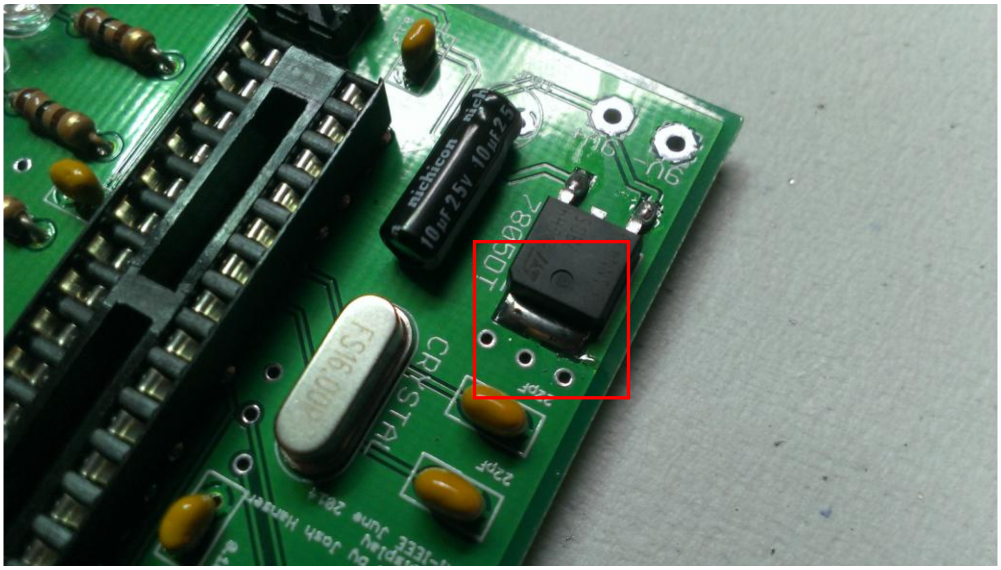




Once the left lead is soldered and the regulator is lined up correctly, solder the right lead. Next, place the tip of your soldering iron so that it touches both the top metal part of the regulator and the pad below it as shown in the picture.

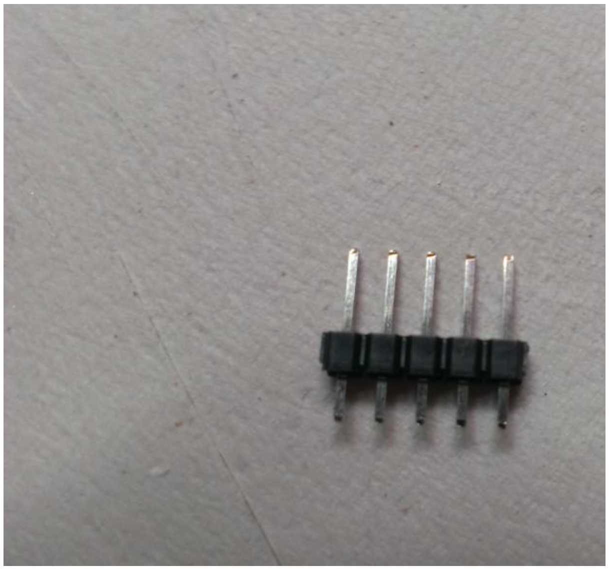


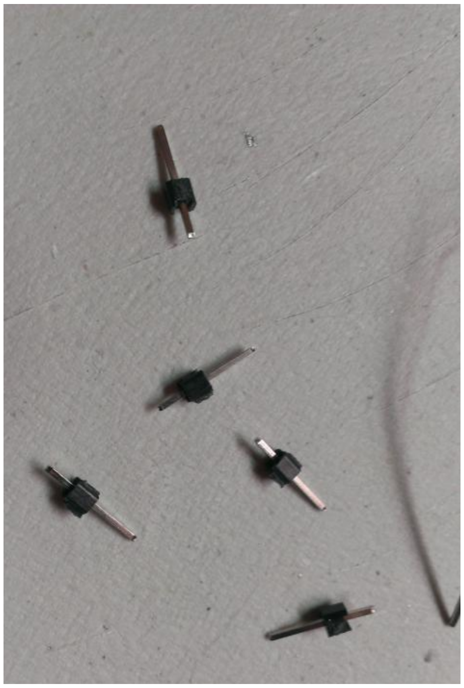
Let the soldering iron heat this area for a few seconds and then apply solder to the whole pad.



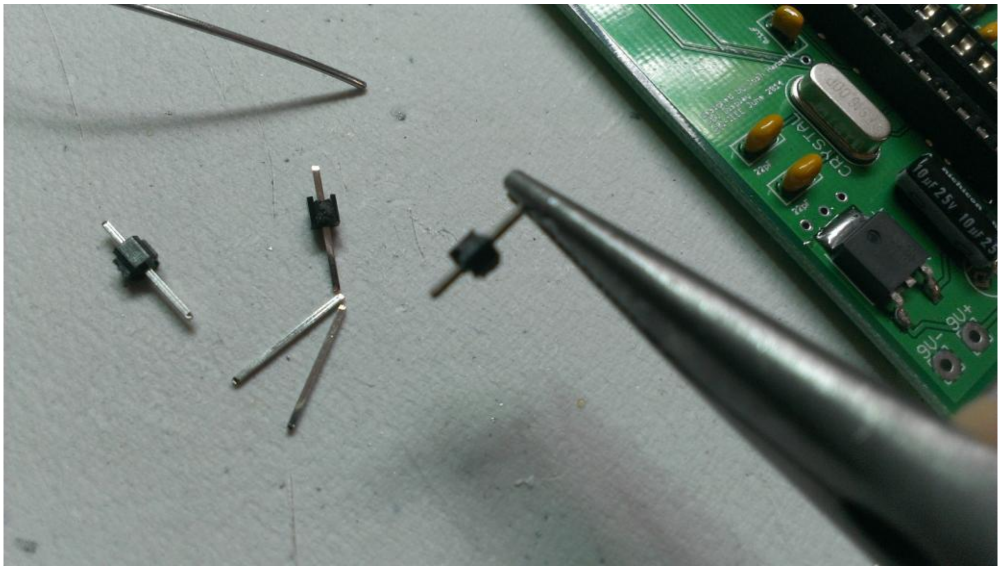
Soldering Step K - Accelerometer

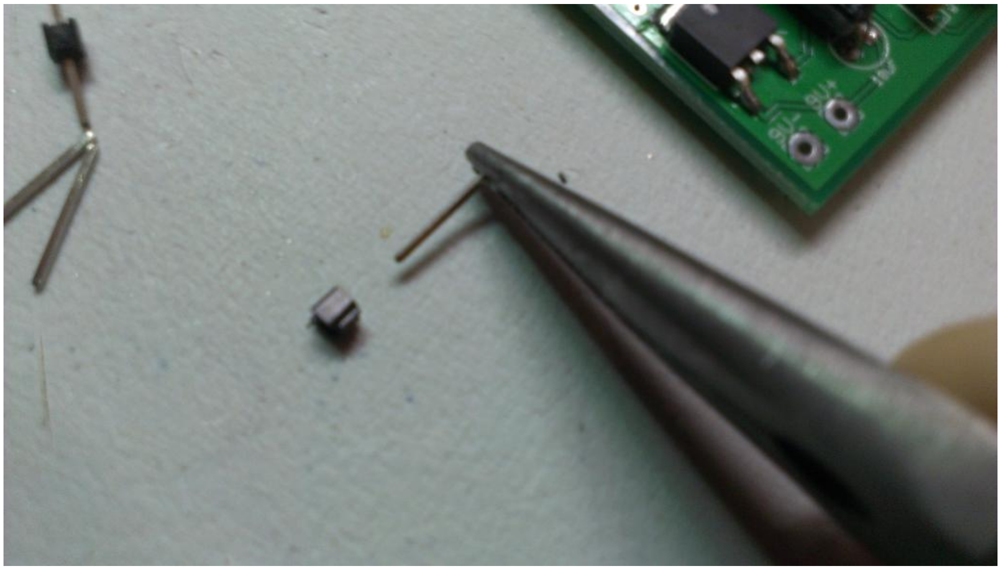
The accelerometer daughter board mounts to the PCB in the upper left corner. Begin by cutting the 5 pin header into 5 individual parts. Don't worry about damaging the plastic because all of it will be removed.

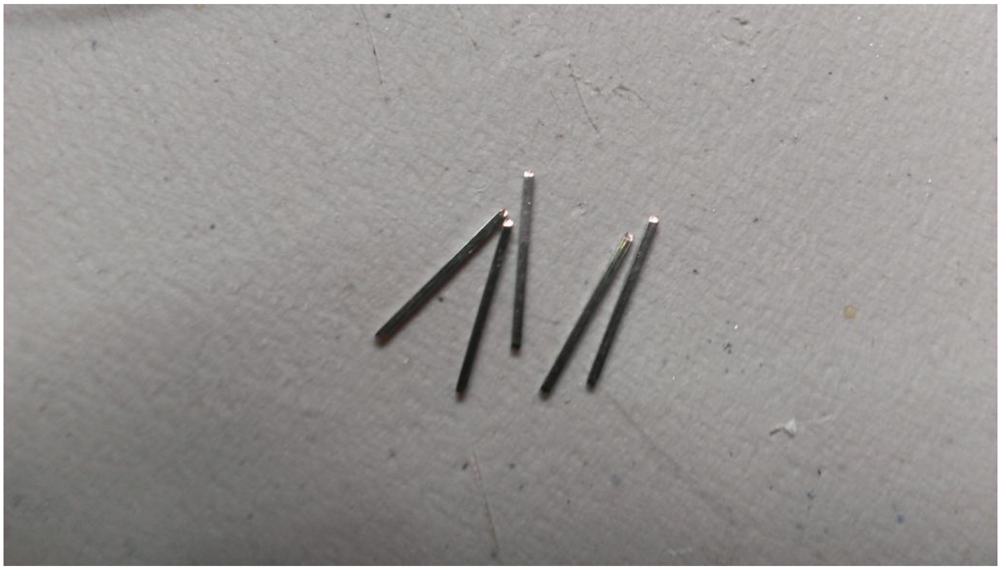




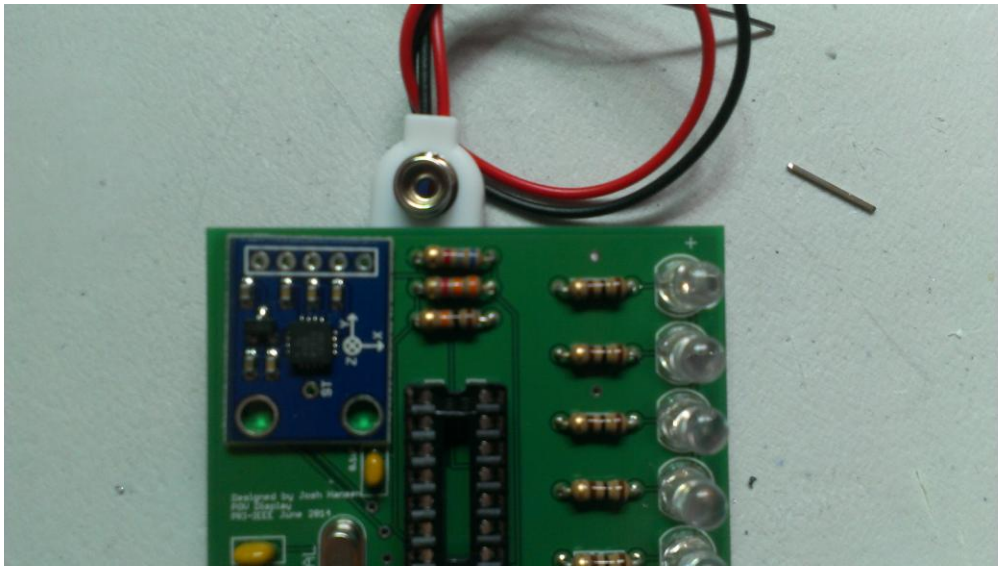
Next, hold one end of the individual header with a pair of pliers and slide the plastic off with your fingers.



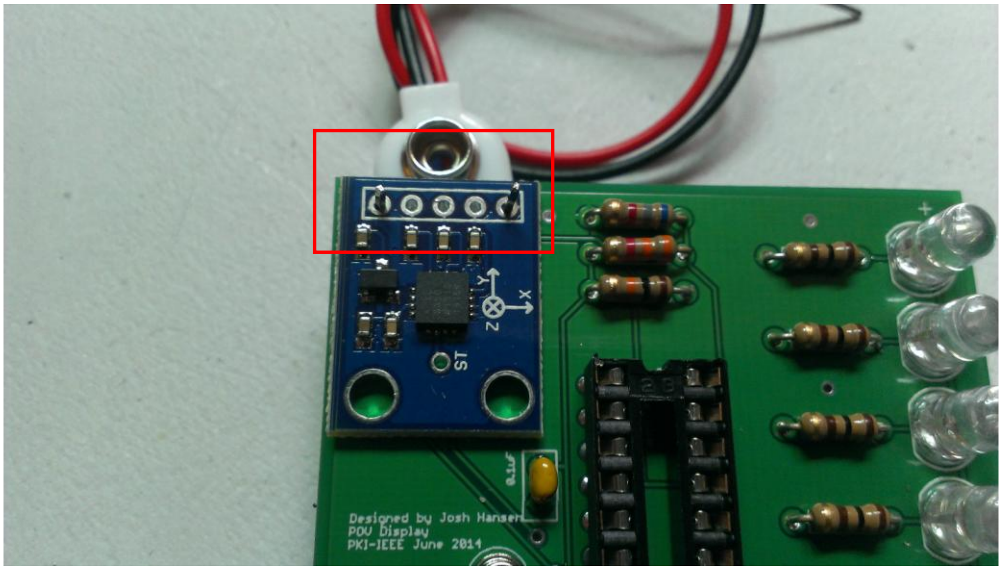

5V-
5V+



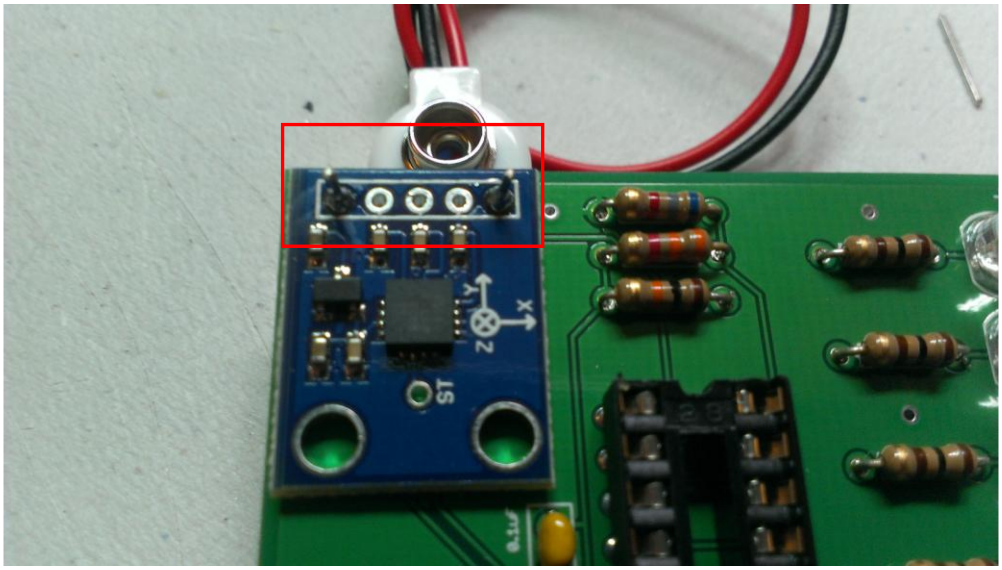
After you have all five headers separated and the plastic removed, prop the PCB up on the battery connector and line up the accelerometer board on the PCB as shown in the picture.



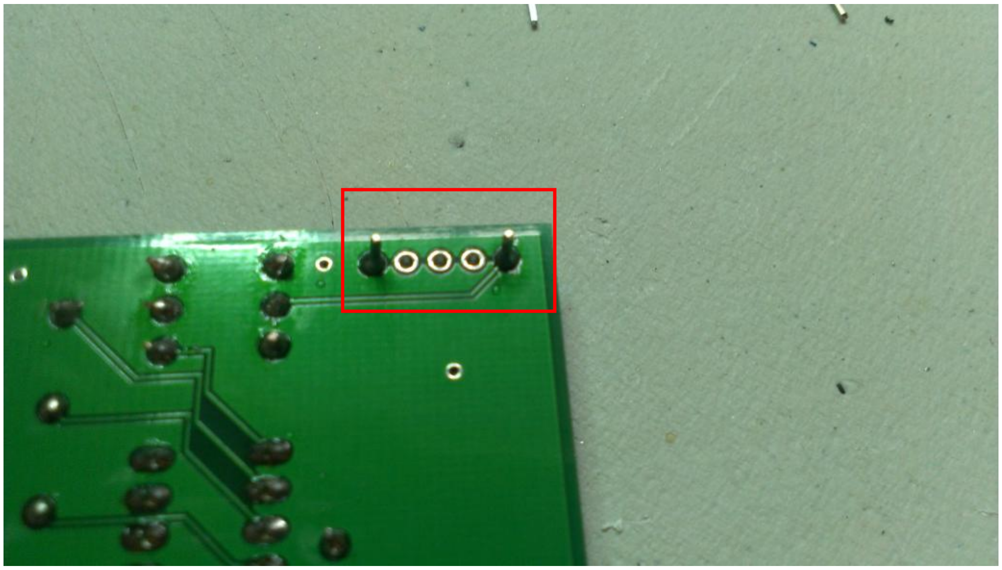
Insert two of the header pins into the right and left most holes of the accelerometer. Ensure that the pins go all the way through both the accelerometer board and the PCB and that you have enough pin sticking out of the back of the PCB to solder to.



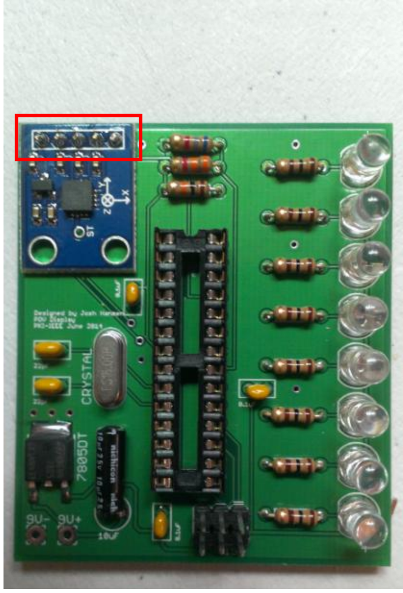
Once you have the pins positioned correctly and the accelerometer aligned, solder the two pins to the accelerometer board.



Flip the PCB over and solder one of the pins to the PCB. Ensure that the accelerometer board is laying flat against the PCB and that it is aligned parallel to the edge of the PCB. Once the alignment is set, solder the remaining pin to the PCB.

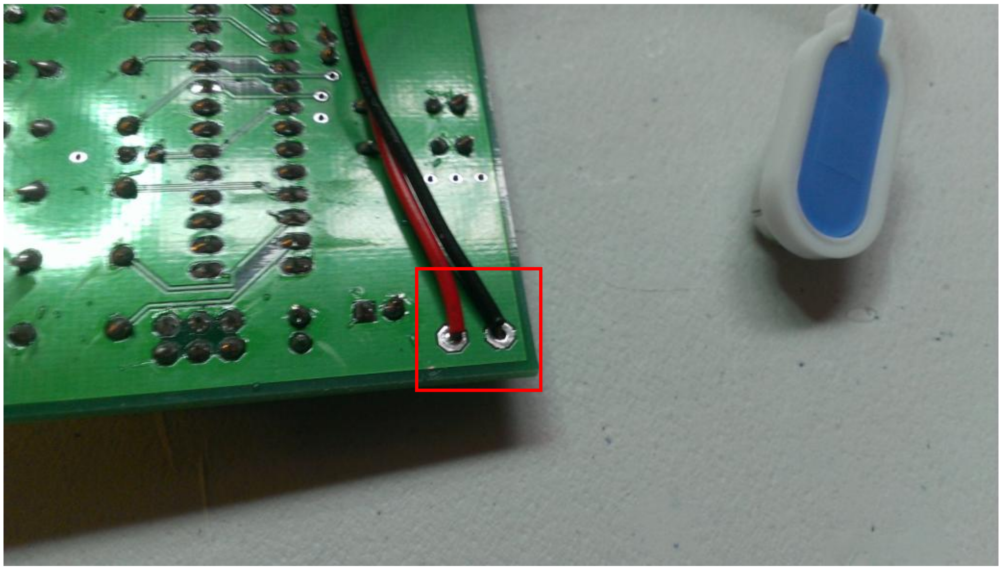


With the two pins soldered, insert and solder the remaining three center pins in the same way that the outer two pins were done.



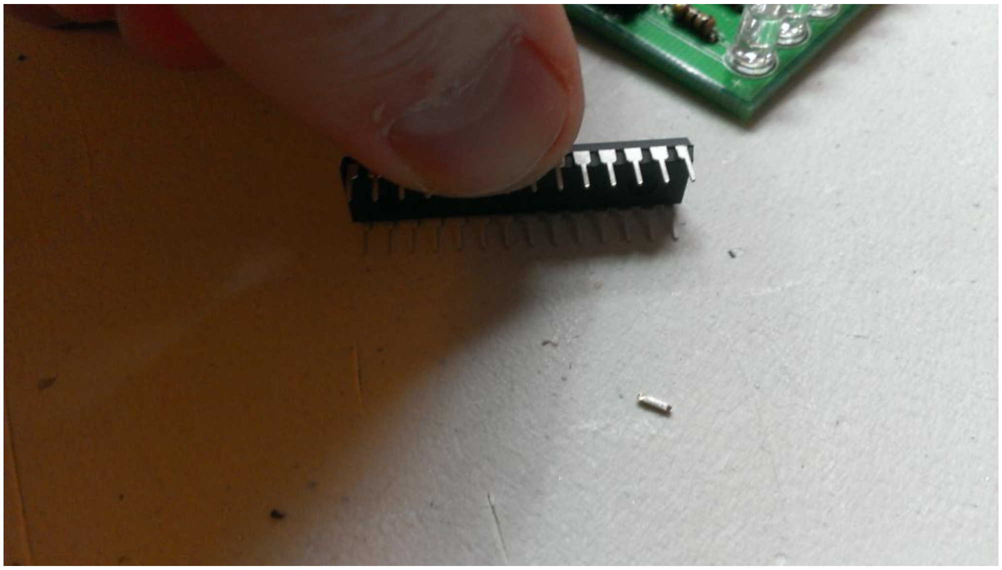
Soldering Step L - 9V Battery Connector

Insert the battery connector wires into the back of the PCB ensuing that the red wire goes through the 9V+ hole and the back wire goes through the 9V- hole. Once you are sure the wires are in their correct places, solder them from the front of the PCB.

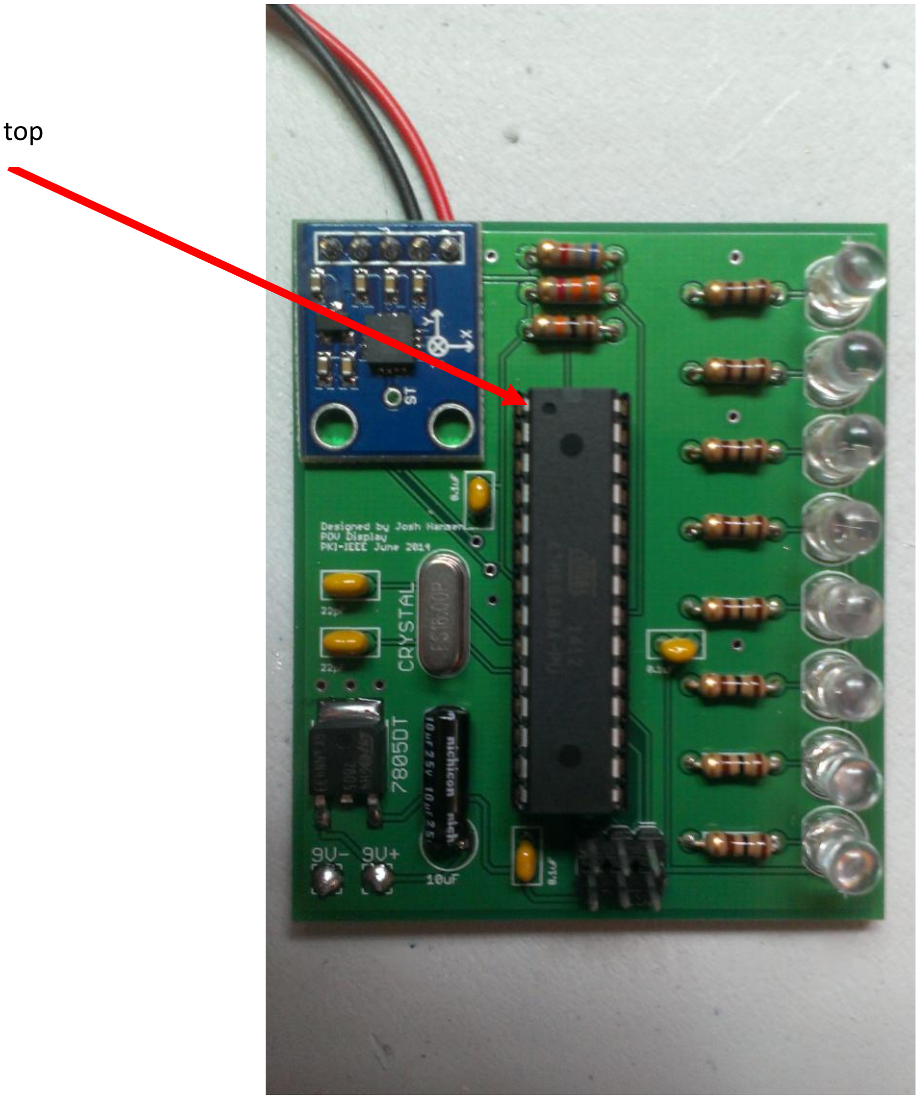


Step 3 - Inserting the Microcontroller

The pins of the microcontroller will not be bent at a 90 degree angle when you get them. One of the ways to bend the pins is to place a row of the pins against the table and push down on the microcontroller to bend the row of pins in. Do this to both sides.

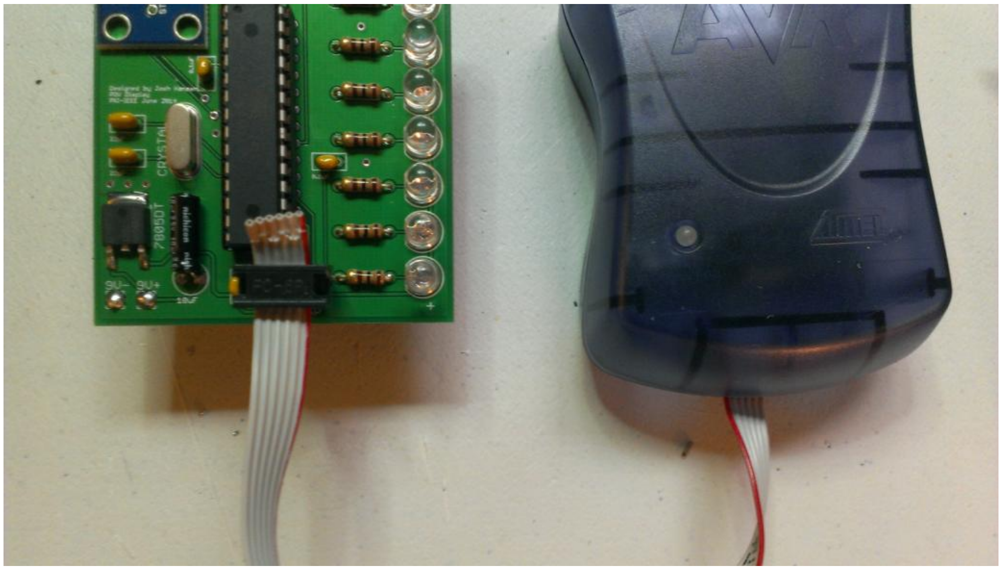


Once you have the pins bent in at 90 degrees, insert the microcontroller into the socket. Before pressing the microcontroller all the way in, ensure that all pins are aligned correctly and the microcontroller is facing the same way as in the picture.

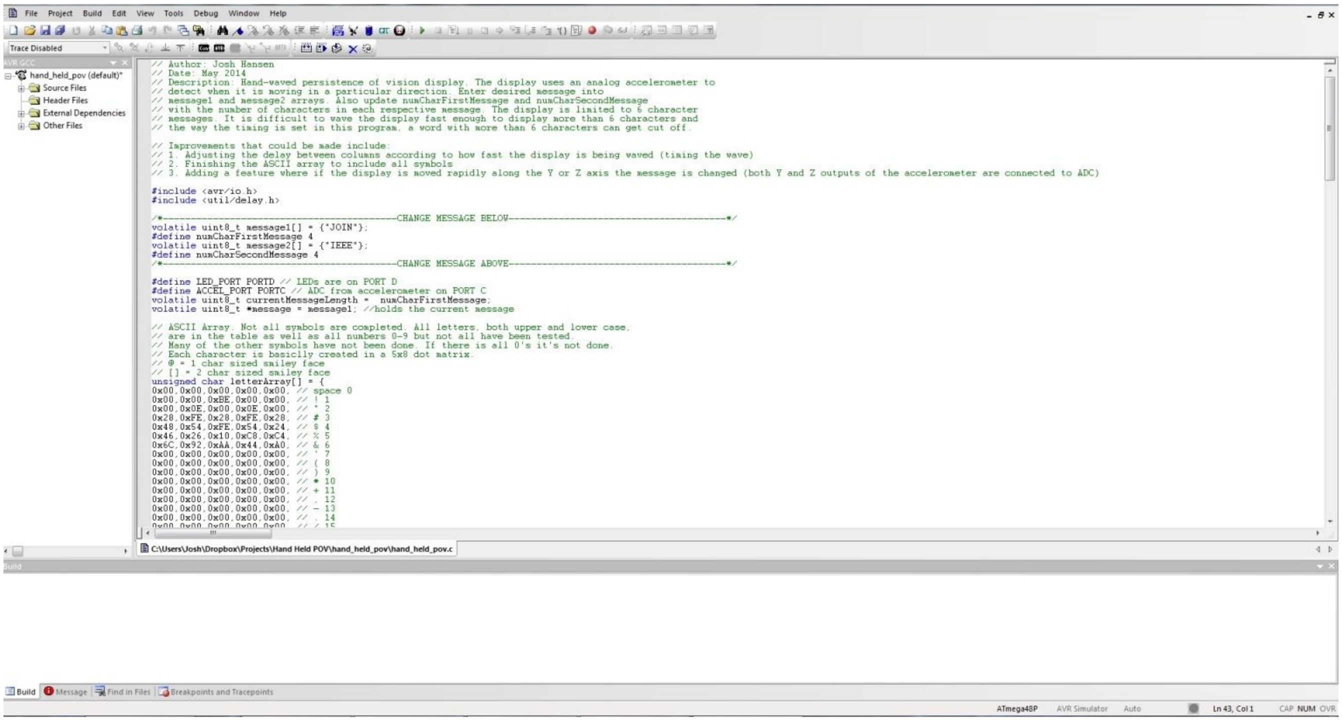

Pin 1 at top

Step 3 - Programming the Microcontroller (AVR Studio 4)

Begin by connecting the 9V battery to the display and then connecting the ISP programmer to the 6 pin header as shown in the picture.



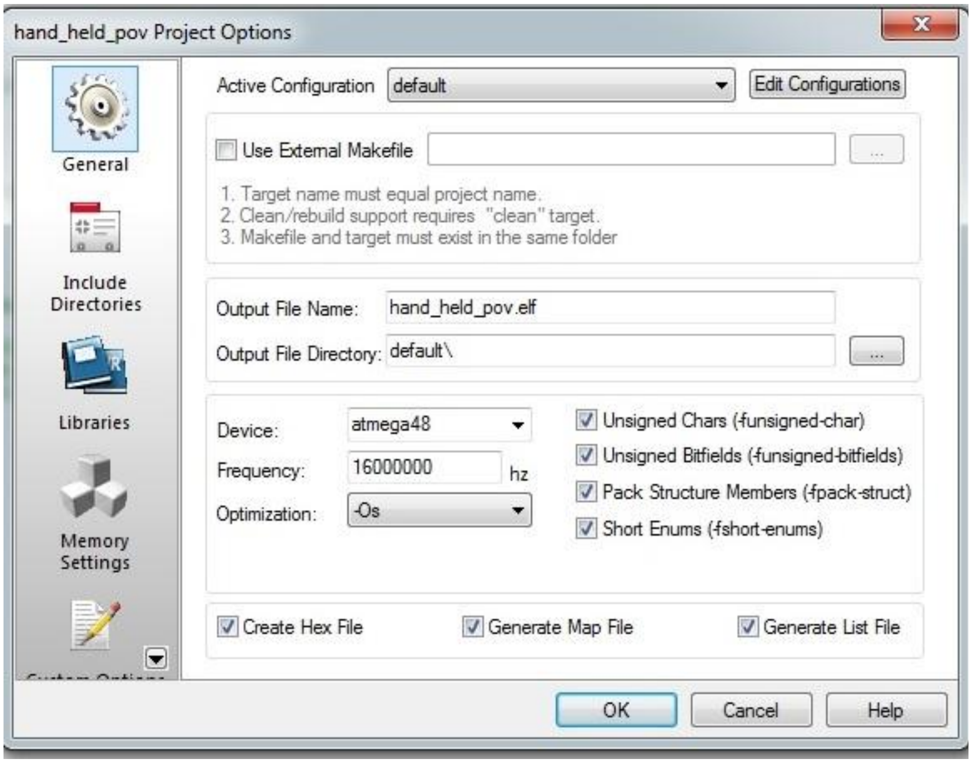
Open AVR Studio and start a new project selecting ATMEGA48 as the device. Copy and paste the code supplied with this document into the project.



```
// Author: Josh Hansen
// Date: May 2014
// Description: Hand-waved persistence of vision display. The display uses an analog accelerometer to
// detect when it is moving in a particular direction. Enter desired message into
// message1 and message2 arrays. Also update numCharFirstMessage and numCharSecondMessage
// with the number of characters in each respective message. The display is limited to 6 character
// messages. It is difficult to wave the display fast enough to display more than 6 characters and
// the way the timing is set in this program, a word with more than 6 characters can get cut off.

// Improvements that could be made include:
// 1. Adjusting the delay between columns according to how fast the display is being waved (timing the wave)
// 2. Finishing the ASCII array to include all symbols
// 3. Adding a feature where if the display is moved rapidly along the Y or Z axis the message is changed (both Y and Z outputs of the accelerometer are connected to ADC)

#include <avr/io.h>
#include <util/delay.h>

/*-----------------------------------------CHANGE MESSAGE BELOW--------------------------------------*/
volatile uint8_t message1[] = {"JOIN"};
#define numCharFirstMessage 4
volatile uint8_t message2[] = {"IEEE"};
#define numCharSecondMessage 4
/*-----------------------------------------CHANGE MESSAGE ABOVE--------------------------------------*/

#define LED_PORT PORTD // LEDs are on PORT D
#define ACCEL_PORT PORTC // ADC from accelerometer on PORT C
volatile uint8_t currentMessageLength =  numCharFirstMessage;
volatile uint8_t *message = message1; //holds the current message

// ASCII Array. Not all symbols are completed. All letters, both upper and lower case,
// are in the table as well as all numbers 0-9 but not all have been tested.
// Many of the other symbols have not been done. If there is all 0's it's not done.
// Each character is basiclly created in a 5x8 dot matrix.
// @ = 1 char sized smiley face
// [] = 2 char sized smiley face
unsigned char letterArray[] = {
0x00,0x00,0x00,0x00,0x00, // space 0
0x00,0x00,0xBE,0x00,0x00, // ! 1
0x00,0x0E,0x00,0x0E,0x00, // " 2
0x28,0xFE,0x28,0xFE,0x28, // # 3
0x48,0x54,0xFE,0x54,0x24, // $ 4
0x46,0x26,0x10,0xC8,0xC4, // % 5
0x6C,0x92,0xAA,0x44,0xA0, // & 6
0x00,0x00,0x00,0x00,0x00, // ' 7
0x00,0x00,0x00,0x00,0x00, // ( 8
0x00,0x00,0x00,0x00,0x00, // ) 9
0x00,0x00,0x00,0x00,0x00, // * 10
0x00,0x00,0x00,0x00,0x00, // + 11
0x00,0x00,0x00,0x00,0x00, // , 12
0x00,0x00,0x00,0x00,0x00, // - 13
0x00,0x00,0x00,0x00,0x00, // . 14
0x00,0x00,0x00,0x00,0x00, // / 15
```

C:\Users\Josh\Dropbox\Projects\Hand Held POV\hand_held_pov\hand_held_pov.c

Next go to Project/Configuration Options. Under the General settings, select ATMEGA48 as the device and enter 16000000 for the frequency then click OK.



hand_held_pov Project Options

Active Configuration: default

□ Use External Makefile

1. Target name must equal project name.
2. Clean/rebuild support requires "clean" target.
3. Makefile and target must exist in the same folder

Output File Name: hand_held_pov.elf

Output File Directory: default\

Device: atmega48

Frequency: 16000000 hz

Optimization: -Os

☑ Unsigned Chars (-funsigned-char)

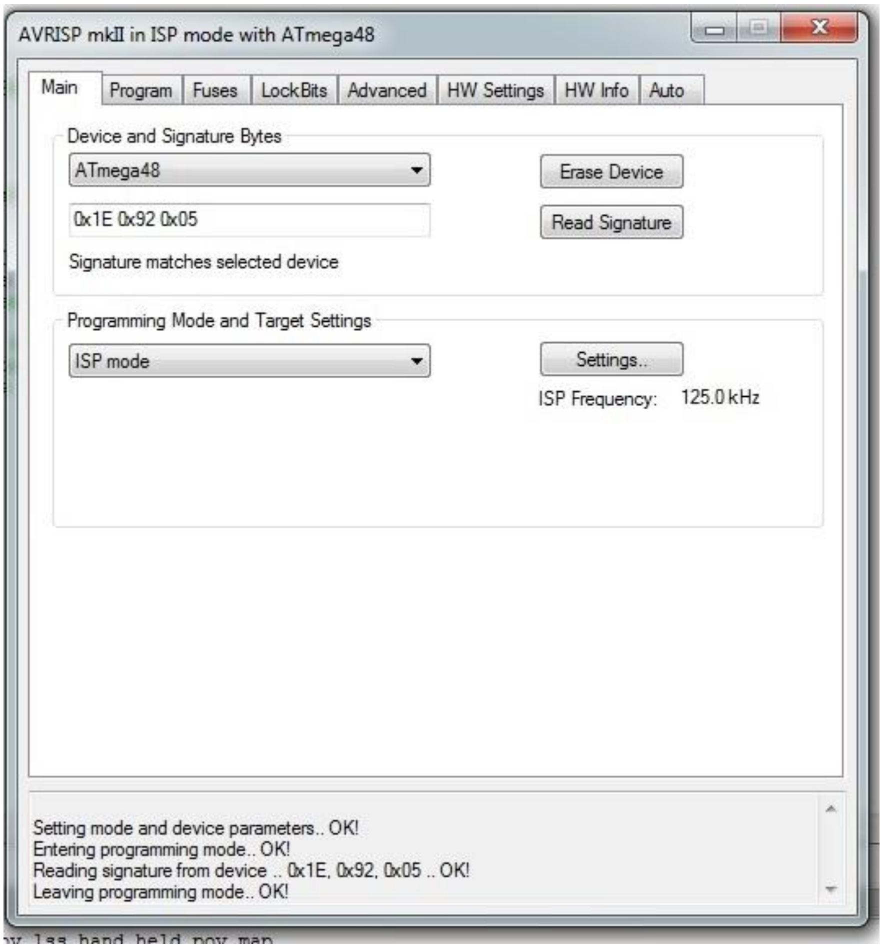
☑ Unsigned Bitfields (-funsigned-bitfields)

☑ Pack Structure Members (-fpack-struct)

☑ Short Enums (-fshort-enums)

☑ Create Hex File ☑ Generate Map File ☑ Generate List File

Next, change the display message by following the comments in the code and then build the project by pressing F7. After ensuring your display is connected to your ISP programmer and the battery is also connected, go to Tools/Program AVR/Auto Connect. Go to the main tab and check that atmega48 is selected as the device. Click Read Signature and ensure that you get the "Signature matches selected device" message before continuing.



AVRISP mkII in ISP mode with ATmega48
Main
Program
Fuses
LockBits
Advanced
HW Settings
HW Info
Auto
Device and Signature Bytes
Erase Device
Read Signature
Programming Mode and Target Settings
Settings..

ATmega48

0x1E 0x92 0x05

Signature matches selected device

ISP mode

ISP Frequency: 125.0 kHz

```
Setting mode and device parameters.. OK!
Entering programming mode.. OK!
Reading signature from device .. 0x1E, 0x92, 0x05 .. OK!
Leaving programming mode.. OK!
```

Next go to the Fuses Tab and enter the following hex values:

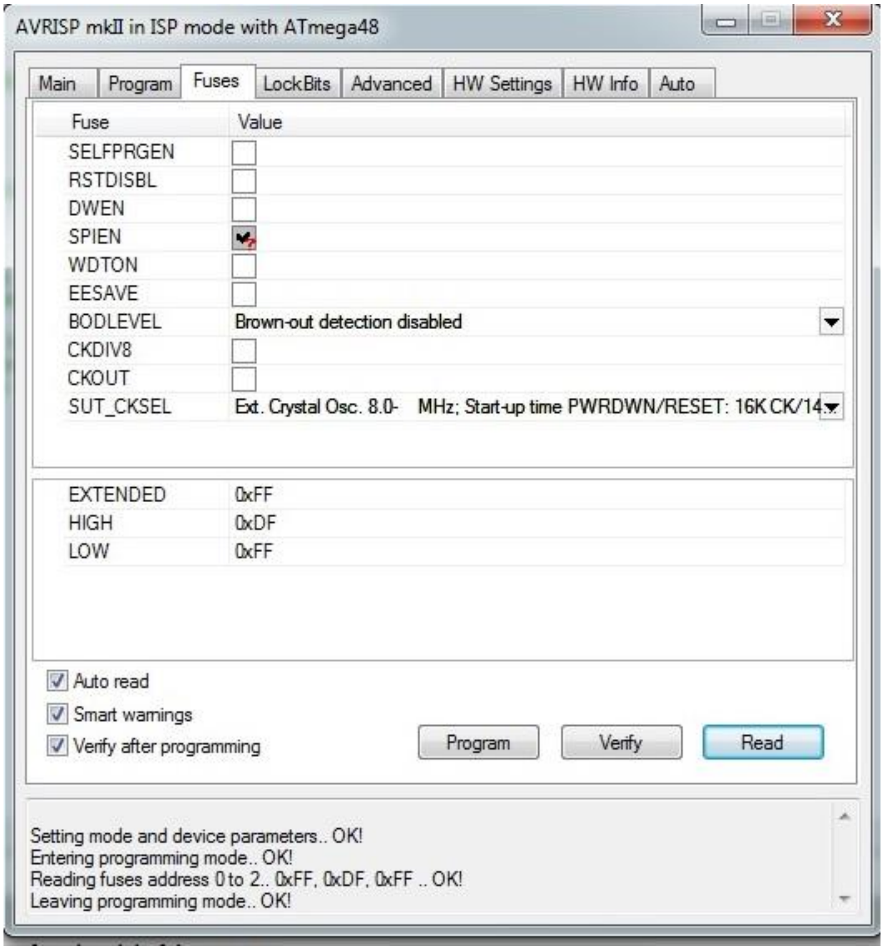
Extended: 0xFF

HIGH: 0xDF

LOW: 0xFF

YOU MUST ENSURE THESE VALUES ARE CORRECT. If you make a mistake and enter the wrong values it is possible that your microcontroller will become unusable.

After you are sure the values are correct, click the program button.



Fuse	Value
SELFPRGEN	□
RSTDISBL	□
DWEN	□
SPIEN	☑
WDTON	□
EESAVE	□
BODLEVEL	Brown-out detection disabled
CKDIV8	□
CKOUT	□
SUT_CKSEL	Ext. Crystal Osc. 8.0- MHz; Start-up time PWRDWN/RESET: 16K CK/14

EXTENDED	0xFF
HIGH	0xDF
LOW	0xFF

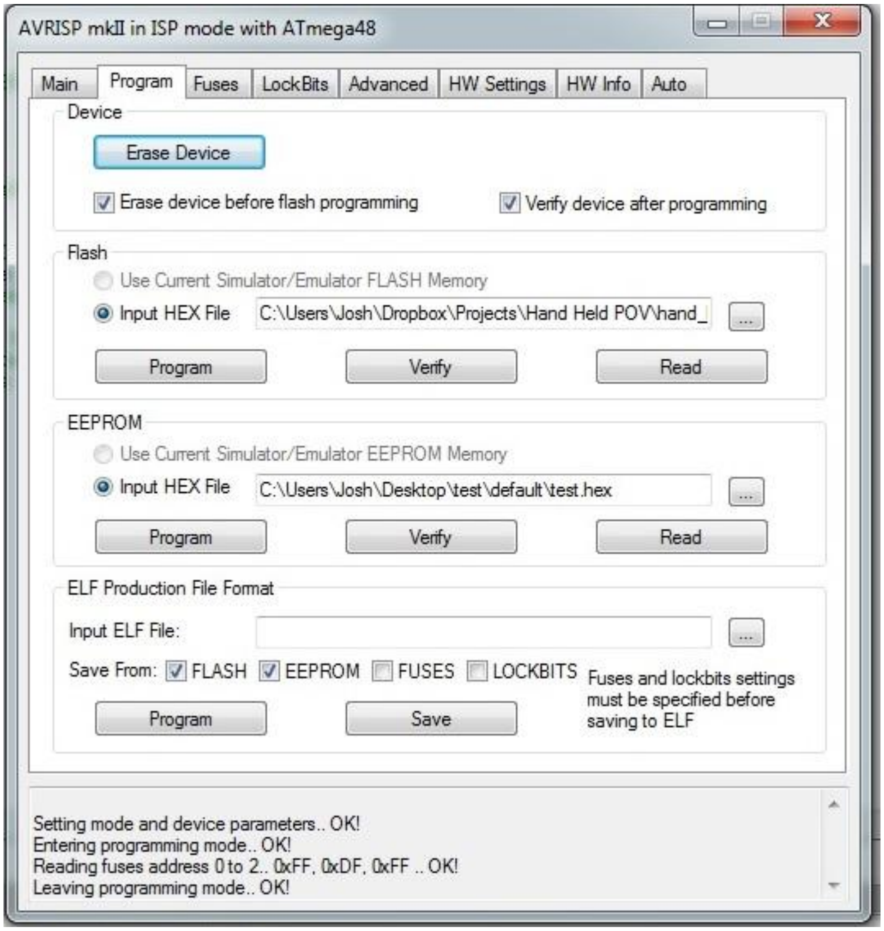
☑ Auto read

☑ Smart warnings

☑ Verify after programming

```
Setting mode and device parameters.. OK!
Entering programming mode.. OK!
Reading fuses address 0 to 2.. 0xFF, 0xDF, 0xFF .. OK!
Leaving programming mode.. OK!
```

Once you have programmed the fuses, go to the Program tab. Under the Flash section, browse to the location your project is saved at and select the .hex file. Once the .hex file location is selected, click the Program button. The microcontroller should then be programmed.



AVRISP mkII in ISP mode with ATmega48

☑ Erase device before flash programming ☑ Verify device after programming

○ Use Current Simulator/Emulator FLASH Memory

● Input HEX File C:\Users\Josh\Dropbox\Projects\Hand Held POV\hand_

○ Use Current Simulator/Emulator EEPROM Memory

● Input HEX File C:\Users\Josh\Desktop\test\default\test.hex

Input ELF File:

Save From: ☑ FLASH ☑ EEPROM □ FUSES □ LOCKBITS

Fuses and lockbits settings must be specified before saving to ELF

```
Setting mode and device parameters.. OK!
Entering programming mode.. OK!
Reading fuses address 0 to 2.. 0xFF, 0xDF, 0xFF .. OK!
Leaving programming mode.. OK!
```

Disconnect the ISP programmer from the header on the display. You should now be able to wave the display and see the message you programmed appear.