Week 4: Electronics Production

19.2. 2015

This week we will learn about electronics production – a new subject for me, at least.

The agenda:

PCB fabrication etching lithography, transfer ferric/cupric chloride, ammonium/sodium persulfate MSDS waste disposal machining tools 0.010 1/64 1/32 fixturing orientation zeroing lifetime deburring cleaning cutting printing plating sewing **PCB materials** rigid FR4 (epoxy glass) FR1 (phenolic paper) flex Kapton #1 epoxy film, #1126 copper tape high-frequency teflon glass copper 0.5 oz: 17.5 um 1.0 oz: 35 um 2.0 oz: 70 um **board houses** AP Circuits, Advanced, Sierra, Screaming Circuits, Gold Phoenix design rules width/spacing (15, 5 mils) layers 1, 1.5, 2, 4, N solder mask, silk screen vias blind, buried **components** through-hole surface-mount chip-scale **breadboards** **assembly** solder eutectic wetting flux wire, paste, bar manual, reflow, wave ROHS stuffing component orientation tacking down parts bottom to top, inside to outside fumes washing desoldering braid hot air gravity cutting traces, adding jumpers pick-and-place encapsulation **CAM** fab modules linewidth **assignment** make the FabISP in-circuit programmer David Andy Valentin hello.ISP.44.cad board components traces interior hello.ISP.44.res.cad board traces interior inventory microcontroller crystal USB connector ribbon connector Zener diode jumper firmware.zip USB power make clean make hex (sudo) make fuse (check programmer in Makefile, may need to repeat) (sudo) make program desolder SJ1 and SJ2 make IDC ISP cable, connecting header pin 1 to pin 1, check wires

The assignment for this week is to make the FabISP in-circuit programmer.

A new version of Antimony will come next week – follow the commit.

Fabmodules.org

Reviews:

Brazilian student who created a press-fit for a form.
Michael Schutz
Daniel Parilla
Patricio Ortiz
Gabriel Ewing
Guðbjörg
Þórarinn
Igor (Russia)
Yoshi Yamahara
Ben Mathes
Jasmin Cheng

This week:
Making of circuitboard

Programming of microcontroller.
A device to make a device.
60 dollara
We will make one and use it to program other things

To make a circuit board by etching
Ferric cubric chloride, sulphates

MSDS – hazardous materials

Waste disposal – to dispose of them in a safe way

Machining:
Miling machine to mill the board
Endmills used to cut
As the tools get smaller the easier they break, use the biggest ones possible

Double stick tape – holding the board in place and flat

Zero-ing the tools, setting it exactly on the top of the board, tapping it a bit, hold it to tighten.
A uniform cut is desired

Common irritating beginners mistake – set screw should be just snug, not too tight

Sacrificial layer – replacing needed every few months
If the bed is not clean and flat, the board moves a little, endmills get broken

The endmills have a lifetime… when the tool is new very fine shavings can be seen around the cut. But when the tool gets old rough cuts are visible. Burrs with a very new tool…

Washing the boards to prevent fingerprints

Research on printable materials
Research: e-broidery

PCB
FR4 Epoxy glass – kills the tools
FR1 phenolic papter: is not quite as strong and does not so well at high temperature, but cuts beautifully

Flexible circuites
Epoxy film tape –

High frequency – typically Teflon or glass
Copper – in ounces per area

Make a PCB

Board Houses – send to them – check list
Euro boards?
OSH Park – an electric ecosystem

1/64

One layer PCB, possible to make more layer PCBs

1.5 layer – 0 crossover

Cheapest electronics
Use as few layers as you can get away with
Set screen that does the writing on the board
Via is a connection between the top and the bottom

Components
Octopart – version of a microcontroller in the surface, through hole that goes through a hole in the board
Anchorage if you need a better connection

Solder paste –

Bread boards, then PBC – the electrical performance is terrible, causing instability.

Quickturn design in fab

Solder – lead tin, dip in the middle, typically a combination or two or more alloys, the flux helps to clean the join and attach the bits

Solder paste has a lifetime and has to be refrigerated

Stuffing – the boards. Attach the components. Board – chip. See picture.
Clean the tip. Get a bit of solder on the tip. When you bring in the iron, it helps the heat transfer. Heat on the junction of the solder iron and the bit and then bring in the solder.

Solder the components. Orientation of the component is important. From the bottom to top. When you are stuffing the board an important trick is to … use the solder to tack it down.

When you solder there are fumes released from it. Exhaust fans to help.
All the components are designed to wash and that gets rid with the flux.

Mistakes:
Braid – add some solder to the braid, it wets the join and takes the rest away.

Scalpel knife can cut out mistakes, like traces

DDM Novastar – making of boards – DIY – machine building

If your board is going to be in the environment, it needs to be incapsulated

Board to make – board perimeter

Fabmodules.org
Png
PBC.traces
Roland MDX-20.rml
PBC.outline

Calculating of the outline?

Ribbon connector going into …
IDC – goes into housing
Pulling the ribbon and push twice

Firmware.zip

Mill the board and stuff them this week

Mac-version – Check Tutorials

Next week review – go through the steps needed, in the review to see successes and failure, talk project plans. All websites and project pages up.

`Make:` hello.ISP.44.res.cad

Recommended:
http://academy.cba.mit.edu/classes/embedded_programming/hello.ISP.44components.png - a microcontroller, that can do various tasks, like read sensors.

Before next meeting – check tutorials:

ISP = in circuit programming

Notes on using the Roland Modela milling machine – for milling PCBs:
Adam Harris's video: FabLab Using the Modela to Mill PCBs

1. Start with png images.
2. Run from fabmodules: run from command line, Ctrl – Alt, T, type fab, opens up fabmodule selector.
3. Insertimage, select traces png file – select Roland Modela rml – click make PNG.

4. A window pops up. Load the image you wish to cut (isp.png). Select width of the drill bit (1/64). At the bottom change the C depth (intensity) to 0.5 mm. Set the speed to 4 mm/sec. Set xmin to 2 and ymin to 2.
5. In view mode tape the copper board evenly and glue it on to a sacrificial layer, that can be cut into when the final cutting of the board will be done.
6. Move the tool to home and set the height, just on top of the material, by loosening the tool, move it down, turn it slightly and fasten the screw.
7. Now calculate the trences and move to xmin ymin.
8. Press send, to send the message from the computer to the milling machine.

Soldering

Programming

1. Get Crosspack AVR – download
2. Get Make, via download of Xcode from Apple
3. Download firmware: FabISP Firmware for MacOS 10.8.2
4. Open terminal navigate to the desktop:
5. cd ~/Desktop/
6. Unzip the firmware.zip directory (the directory will be "firmware.zip" if you downloaded the earlier version):
7. unzip fabISP_mac.0.8.2_firmware.zip
8. Move into the newly created firmware directory on your desktop
9. cd ~/Desktop/firmware

Edit the Make File

The Makefile is in the firmware directory that we downloaded. The Makefile is set up to work with the AVRISP2 by default. If you are using another programmer, you will need to edit the Makefile.

Open the Makefile with TextEdit.

Make Changes - All OS:

A window will open containing the Makefile. Go to the line that says:
#AVRDUDE = avrdude -c usbtiny -p $(DEVICE) # edit this line for your programmer
AVRDUDE = avrdude -c avrisp2 -P usb -p $(DEVICE) # edit this line for your programmer

- If using the USBtiny programmer or another FabISP
- Remove the "#" in front of the line with "usbtiny" in it
- Add a "#" to beginning the line with the "avrisp2" in it to comment it out.
- save the Makefile

Program the fabISP board

Open Terminal – see log:

```
Last login: Fri Feb 13 17:33:48 on console
Skulinas-MBP:~ skulina$ cd ~/Desktop/
Skulinas-MBP:Desktop skulina$ unzip fabISP_mac.0.8.2_firmware.zip
Archive:  fabISP_mac.0.8.2_firmware.zip
   creating: fabISP_mac.0.8.2_firmware/
  inflating: fabISP_mac.0.8.2_firmware/.DS_Store
   creating: __MACOSX/
   creating: __MACOSX/fabISP_mac.0.8.2_firmware/
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/._.DS_Store
  inflating: fabISP_mac.0.8.2_firmware/main.c
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/._main.c
  inflating: fabISP_mac.0.8.2_firmware/main.elf
  inflating: fabISP_mac.0.8.2_firmware/main.hex
  inflating: fabISP_mac.0.8.2_firmware/main.o
  inflating: fabISP_mac.0.8.2_firmware/Makefile
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/._Makefile
  inflating: fabISP_mac.0.8.2_firmware/usbconfig.h
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/._usbconfig.h
   creating: fabISP_mac.0.8.2_firmware/usbdrv/
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/.DS_Store
   creating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._.DS_Store
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/asmcommon.inc
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._asmcommon.inc
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/Changelog.txt
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._Changelog.txt
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/CommercialLicense.txt
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._CommercialLicense.txt
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/License.txt
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._License.txt
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/oddebug.c
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._oddebug.c
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/oddebug.h
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._oddebug.h
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/oddebug.o
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/Readme.txt
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._Readme.txt
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/USB-ID-FAQ.txt
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._USB-ID-FAQ.txt
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/USB-IDs-for-free.txt
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._USB-IDs-for-free.txt
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbconfig-prototype.h
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbconfig-prototype.h
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrv.c
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbdrv.c
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrv.h
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbdrv.h
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrv.o
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrvasm.asm
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbdrvasm.asm
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrvasm.o
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrvasm.S
  inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbdrvasm.S
  inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrvasm12.inc
```

```
 inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbdrvasm12.inc
 inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrvasm128.inc
 inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbdrvasm128.inc
 inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrvasm15.inc
 inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbdrvasm15.inc
 inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrvasm16.inc
 inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbdrvasm16.inc
 inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrvasm165.inc
 inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbdrvasm165.inc
 inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrvasm18-crc.inc
 inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbdrvasm18-crc.inc
 inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbdrvasm20.inc
 inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbdrvasm20.inc
 inflating: fabISP_mac.0.8.2_firmware/usbdrv/USBID-License.txt
 inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._USBID-License.txt
 inflating: fabISP_mac.0.8.2_firmware/usbdrv/usbportability.h
 inflating: __MACOSX/fabISP_mac.0.8.2_firmware/usbdrv/._usbportability.h
Skulinas-MBP:Desktop skulina$ cd ~/Desktop/firmware
-bash: cd: /Users/skulina/Desktop/firmware: No such file or directory
Skulinas-MBP:Desktop skulina$ cd ~/Desktop/firmware
-bash: cd: /Users/skulina/Desktop/firmware: No such file or directory
Skulinas-MBP:Desktop skulina$ cd ~/Desktop/fabISP_mac.0.8.2_firmware
Skulinas-MBP:fabISP_mac.0.8.2_firmware skulina$ avrdude
Usage: avrdude [options]
Options:
 -p <partno>            Required. Specify AVR device.
 -b <baudrate>          Override RS-232 baud rate.
 -B <bitclock>          Specify JTAG/STK500v2 bit clock period (us).
 -C <config-file>       Specify location of configuration file.
 -c <programmer>        Specify programmer type.
 -D                     Disable auto erase for flash memory
 -i <delay>             ISP Clock Delay [in microseconds]
 -P <port>              Specify connection port.
 -F                     Override invalid signature check.
 -e                     Perform a chip erase.
 -O                     Perform RC oscillator calibration (see AVR053).
 -U <memtype>:r|w|v:<filename>[:format]
                        Memory operation specification.
                        Multiple -U options are allowed, each request
                        is performed in the order specified.
 -n                     Do not write anything to the device.
 -V                     Do not verify.
 -u                     Disable safemode, default when running from a script.
 -s                     Silent safemode operation, will not ask you if
                        fuses should be changed back.
 -t                     Enter terminal mode.
 -E <exitspec>[,<exitspec>] List programmer exit specifications.
 -x <extended_param>    Pass <extended_param> to programmer.
 -y                     Count # erase cycles in EEPROM.
 -Y <number>            Initialize erase cycle # in EEPROM.
 -v                     Verbose output. -v -v for more.
 -q                     Quell progress output. -q -q for less.
 -l logfile             Use logfile rather than stderr for diagnostics.
 -?                     Display this usage.

avrdude version 6.0.1, URL: <http://savannah.nongnu.org/projects/avrdude/>
Skulinas-MBP:fabISP_mac.0.8.2_firmware skulina$ make clean
```

Agreeing to the Xcode/iOS license requires admin privileges, please re-run as root via sudo.

Skulinas-MBP:fabISP_mac.0.8.2_firmware skulina$ sudo make clean

WARNING: Improper use of the sudo command could lead to data loss
or the deletion of important system files. Please double-check your
typing when using sudo. Type "man sudo" for more information.

To proceed, enter your password, or type Ctrl-C to abort.

Password:

You have not agreed to the Xcode license agreements. You must agree to both license agreements below in order to use Xcode.

Hit the Enter key to view the license agreements at
'/Applications/Xcode.app/Contents/Resources/English.lproj/License.rtf'

IMPORTANT: BY USING THIS SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE FOLLOWING APPLE TERMS:

A. MAC SDK AND XCODE AGREEMENT
B. iOS SDK AGREEMENT

APPLE INC.
MAC SDK AND XCODE AGREEMENT

PLEASE READ THIS MAC SDK AND XCODE AGREEMENT ("LICENSE") CAREFULLY BEFORE USING THE DEVELOPER SOFTWARE (DEFINED BELOW). BY USING THE DEVELOPER SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS LICENSE. IF YOU ARE ACCESSING THE DEVELOPER SOFTWARE ELECTRONICALLY, SIGNIFY YOUR AGREEMENT TO BE BOUND BY THE TERMS OF THIS LICENSE BY CLICKING THE "AGREE " BUTTON. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, DO NOT USE THE DEVELOPER SOFTWARE AND CLICK “DISAGREE”.

IMPORTANT NOTE: To the extent that this software may be used to reproduce materials, it is licensed to you only for reproduction of non-copyrighted materials, materials in which you own the copyright, or materials you are authorized or legally permitted to reproduce. If you are uncertain about your right to copy any material, you should contact your legal advisor.

1. General.

By typing 'agree' you are agreeing to the terms of the software license agreements. Type 'print' to print them or anything else to cancel, [agree, print, cancel] agree

You can view the license agreements in Xcode's About Box, or at
/Applications/Xcode.app/Contents/Resources/English.lproj/License.rtf

rm -f main.hex main.lst main.obj main.cof main.list main.map main.eep.hex main.elf *.o usbdrv/*.o main.s usbdrv/oddebug.s usbdrv/usbdrv.s
Skulinas-MBP:fabISP_mac.0.8.2_firmware skulina$ make clean
rm -f main.hex main.lst main.obj main.cof main.list main.map main.eep.hex main.elf *.o usbdrv/*.o main.s usbdrv/oddebug.s usbdrv/usbdrv.s

```
Skulinas-MBP:fabISP_mac.0.8.2_firmware skulina$ make hex
avr-gcc -Wall -Os -DF_CPU=20000000       -Iusbdrv -I. -DDEBUG_LEVEL=0 -mmcu=attiny44 -c
usbdrv/usbdrv.c -o usbdrv/usbdrv.o
avr-gcc -Wall -Os -DF_CPU=20000000       -Iusbdrv -I. -DDEBUG_LEVEL=0 -mmcu=attiny44 -x
assembler-with-cpp -c usbdrv/usbdrvasm.S -o usbdrv/usbdrvasm.o
avr-gcc -Wall -Os -DF_CPU=20000000       -Iusbdrv -I. -DDEBUG_LEVEL=0 -mmcu=attiny44 -c
usbdrv/oddebug.c -o usbdrv/oddebug.o
avr-gcc -Wall -Os -DF_CPU=20000000       -Iusbdrv -I. -DDEBUG_LEVEL=0 -mmcu=attiny44 -c main.c -o
main.o
main.c:88:13: warning: always_inline function might not be inlinable [-Wattributes]
 static void delay ( void )
             ^
avr-gcc -Wall -Os -DF_CPU=20000000       -Iusbdrv -I. -DDEBUG_LEVEL=0 -mmcu=attiny44 -o main.elf
usbdrv/usbdrv.o usbdrv/usbdrvasm.o usbdrv/oddebug.o main.o
rm -f main.hex main.eep.hex
avr-objcopy -j .text -j .data -O ihex main.elf main.hex
avr-size main.hex
   text    data     bss     dec     hex filename
      0    2002       0    2002    7d2 main.hex
Skulinas-MBP:fabISP_mac.0.8.2_firmware skulina$ make fuse
avrdude -c usbtiny -p attiny44  -U hfuse:w:0xDF:m -U lfuse:w:0xFF:m

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e9207
avrdude: reading input file "0xDF"
avrdude: writing hfuse (1 bytes):

Writing | ################################################## | 100% 0.00s

avrdude: 1 bytes of hfuse written
avrdude: verifying hfuse memory against 0xDF:
avrdude: load data hfuse data from input file 0xDF:
avrdude: input file 0xDF contains 1 bytes
avrdude: reading on-chip hfuse data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of hfuse verified
avrdude: reading input file "0xFF"
avrdude: writing lfuse (1 bytes):

Writing | ################################################## | 100% 0.00s

avrdude: 1 bytes of lfuse written
avrdude: verifying lfuse memory against 0xFF:
avrdude: load data lfuse data from input file 0xFF:
avrdude: input file 0xFF contains 1 bytes
avrdude: reading on-chip lfuse data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of lfuse verified
```

```
avrdude: safemode: Fuses OK (H:FF, E:DF, L:FF)

avrdude done.  Thank you.

Skulinas-MBP:fabISP_mac.0.8.2_firmware skulina$ make program
avrdude -c usbtiny -p attiny44  -U flash:w:main.hex:i

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e9207
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
       To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "main.hex"
avrdude: writing flash (2002 bytes):

Writing | ################################################## | 100% 2.48s

avrdude: 2002 bytes of flash written
avrdude: verifying flash memory against main.hex:
avrdude: load data flash data from input file main.hex:
avrdude: input file main.hex contains 2002 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 3.56s

avrdude: verifying ...
avrdude: 2002 bytes of flash verified

avrdude: safemode: Fuses OK (H:FF, E:DF, L:FF)

avrdude done.  Thank you.

avrdude -c usbtiny -p attiny44  -U hfuse:w:0xDF:m -U lfuse:w:0xFF:m

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e9207
avrdude: reading input file "0xDF"
avrdude: writing hfuse (1 bytes):

Writing | ################################################## | 100% 0.00s

avrdude: 1 bytes of hfuse written
avrdude: verifying hfuse memory against 0xDF:
avrdude: load data hfuse data from input file 0xDF:
avrdude: input file 0xDF contains 1 bytes
avrdude: reading on-chip hfuse data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
```

```
avrdude: 1 bytes of hfuse verified
avrdude: reading input file "0xFF"
avrdude: writing lfuse (1 bytes):

Writing | ################################################## | 100% 0.00s

avrdude: 1 bytes of lfuse written
avrdude: verifying lfuse memory against 0xFF:
avrdude: load data lfuse data from input file 0xFF:
avrdude: input file 0xFF contains 1 bytes
avrdude: reading on-chip lfuse data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of lfuse verified

avrdude: safemode: Fuses OK (H:FF, E:DF, L:FF)

avrdude done.  Thank you.

Skulinas-MBP:fabISP_mac.0.8.2_firmware skulina$ make program
```

Verify the results:

Go to the System Profiler > Hardware > USB > Hub:

1. Click the "apple" menu in your main toolbar
2. Select "about this mac"
3. Select "more info"
4. Under the "Contents" menu in the left hand navigation
   - Click "Hardware" to expand the hardware menu (if not already expanded)
   - Click "USB"
   - Under the "USB Device Tree"
   - Click "Hub" to expand the hub menu (if not already expanded)
   - "FabISP" should be listed in the hub menu
5. Your FabISP device has been successfully programmed and is recognized by your computer.

When the programming is done:

Remove the 0 ohm resistor and solder bridge as shown in the picture below. Now you can use it as a programmer to program other boards.