

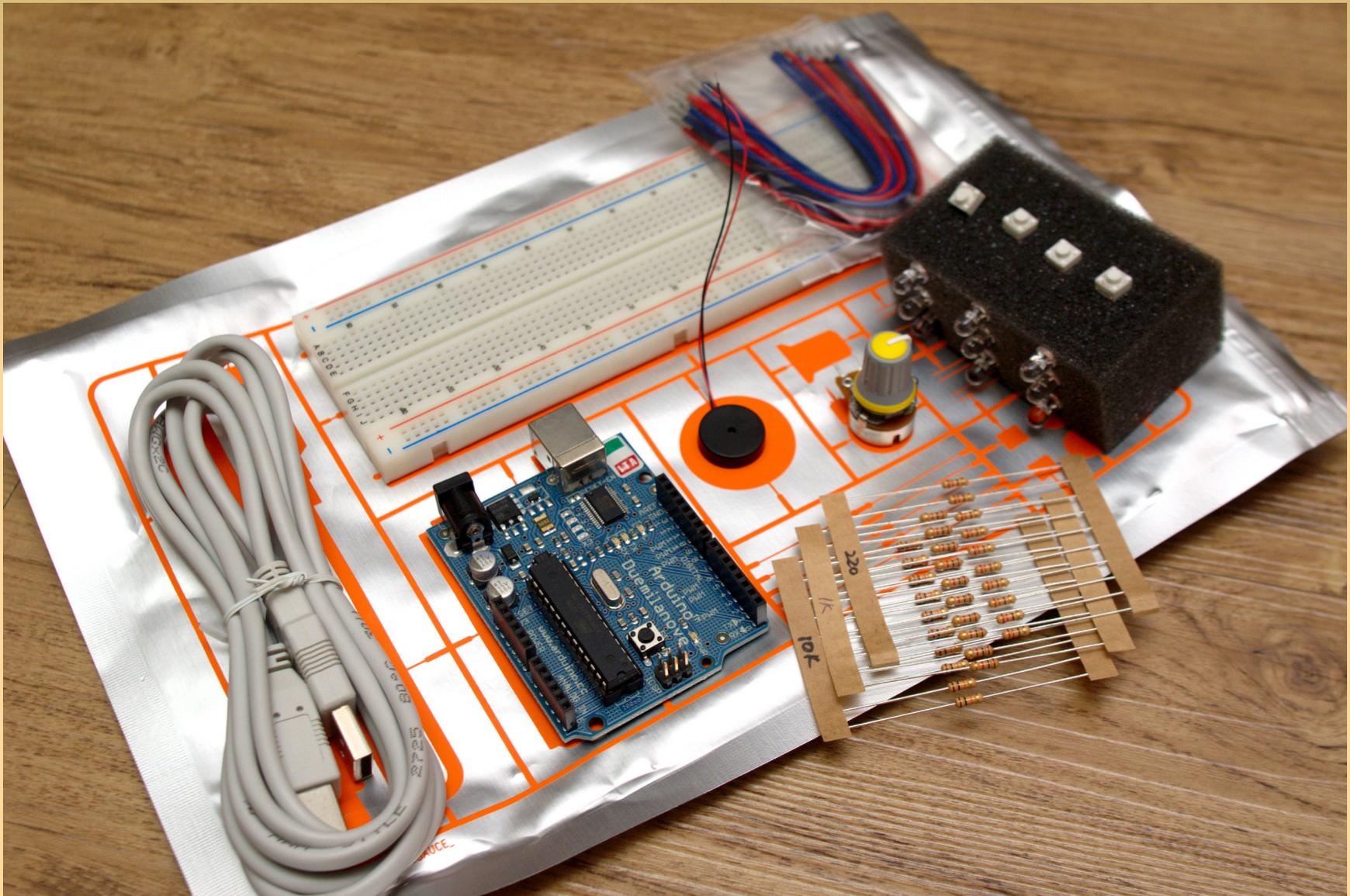
# 우분투리눅스로 임베디드 개발하기

이 호 민

# Host 로 우분투를 사용하는 이유

- 미려하고 편한 UI
- 진화를 멈추지 않은 cmdline – bash
- 안전한 OS
- 더 좋은 FileSystem
- 개발 도구 기본 제공
  - 컴파일러, 툴체인, 디버거, etc...
- 효율적인 패키징 (deb) 시스템

# 아두이노 소개



# 아두이노 소개 2/2

- AVR 프로세서를 사용한 마이컴
  - Open source HW
  - Self-Programming 을 위한 부트로더 내장
  - 프로토타입 개발에 편한 점퍼선 연결 GPIO
  - USB 케이블로 업로드 및 디버깅
- 아두이노 언어
  - C/CPP 에서 중복되는 부분을 빼서 간소화함
  - 스케치 - 아두이노 언어로 작성한 프로그램

# 아두이노 개발환경 설치

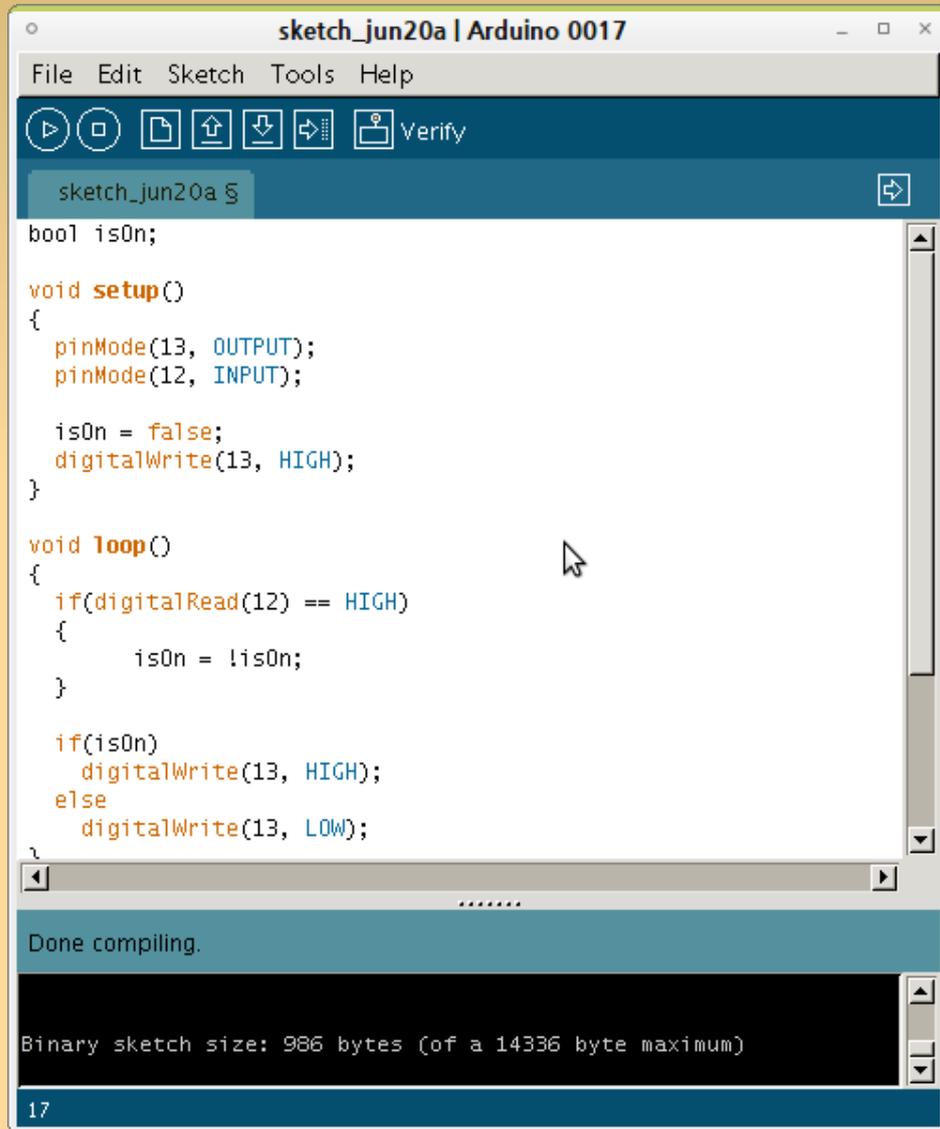
- `$ sudo add-apt-repository ppa:arduino-ubuntu-team`
  - 소스리스트에 아두이노 저장소 (ppa) 추가
- `$ sudo apt-get update`
  - 소스리스트 갱신
- `$ sudo apt-get install arduino`
  - 아두이노 패키지 설치

# 아두이노 패키지 구성

- 필요한 패키지들 (dependency) 이 같이 설치됨
  - avr-cross-toolchain
  - Java, etc...
- debian/control

```
Package: arduino
Architecture: all
Depends: ${misc:Depends}, gcc-avr, avr-libc,
        avrdude (>= 5.10-1ubuntu1),
        default-jre | java6-runtime, librx-tx-java (>=2.1.7r2-4ubuntu1)
Description: The Arduino libraries and the development environment
Arduino is an open-source electronics prototyping platform...
```

# 아두이노 IDE



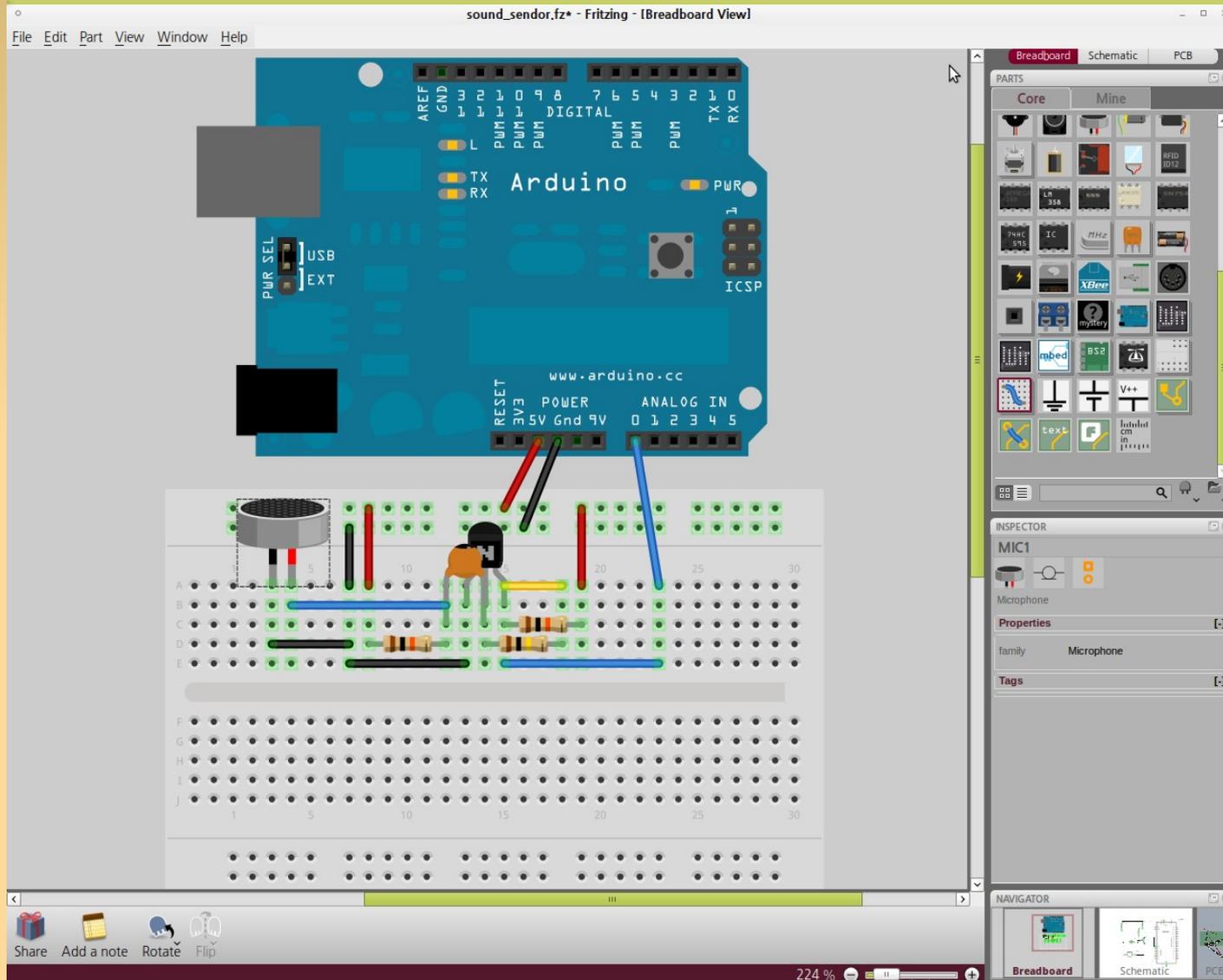
```
sketch_jun20a | Arduino 0017
File Edit Sketch Tools Help
[Icons] Verify
sketch_jun20a §
bool is0n;
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(12, INPUT);

  is0n = false;
  digitalWrite(13, HIGH);
}
void loop()
{
  if(digitalRead(12) == HIGH)
  {
    is0n = !is0n;
  }

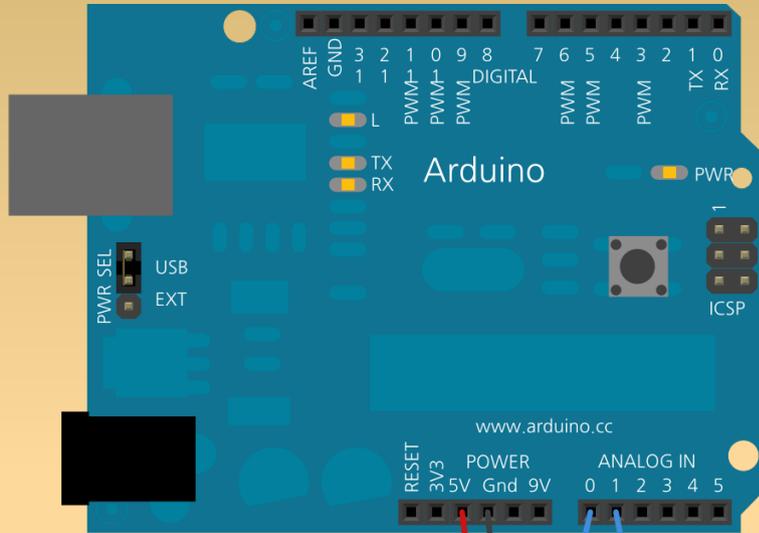
  if(is0n)
    digitalWrite(13, HIGH);
  else
    digitalWrite(13, LOW);
}
Done compiling.
Binary sketch size: 986 bytes (of a 14336 byte maximum)
17
```

- Verify 를 누르면 ..
  - 스케지에 살을 붙여 cpp 언어로 변경
  - avr-gcc 크로스컴파일
- Upload 를 누르면 ...
  - 이미지를 아두이노 보드에 올림

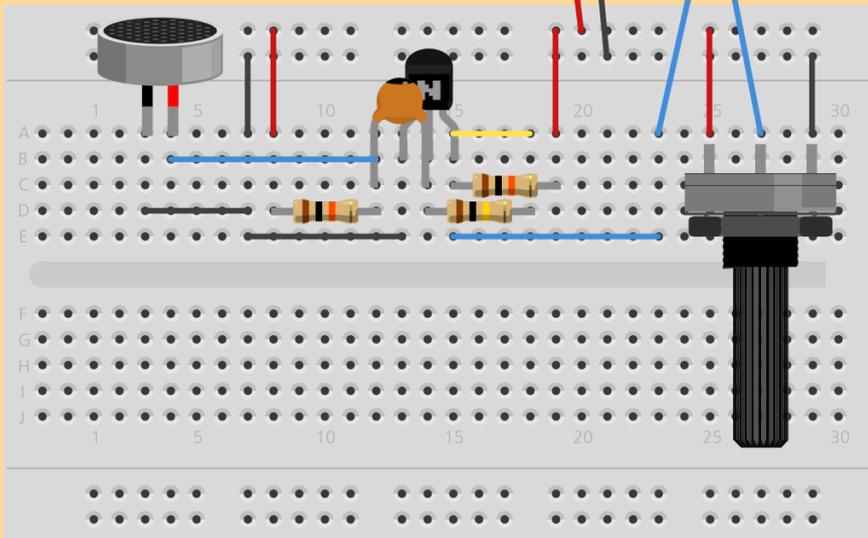
# Frizing - 소개



# Frizing - 브레드보드



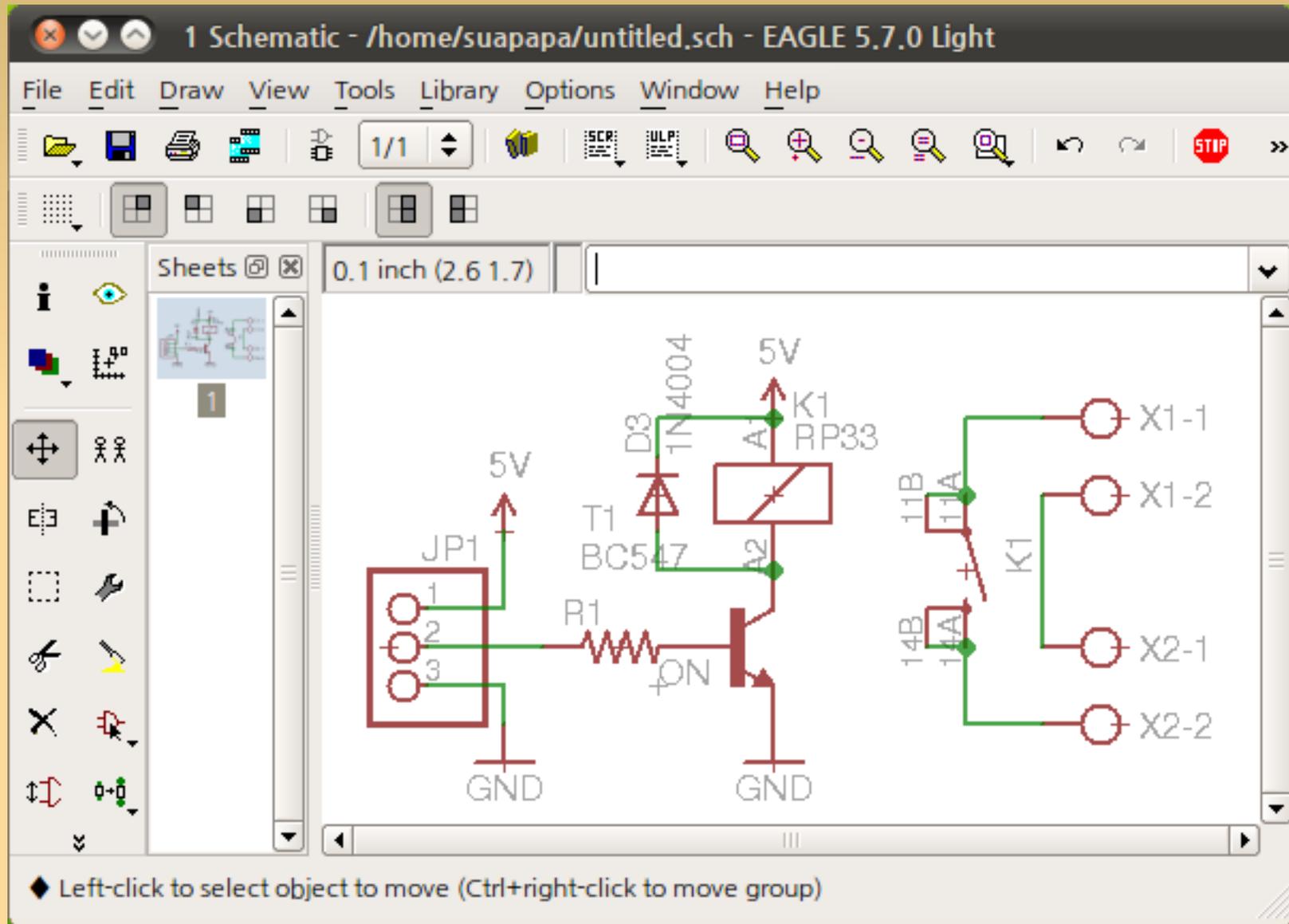
- <http://fritzing.org>
- 빠른 하드웨어 구성
- 적은 수정 비용
- 구멍들은 표준 간격
- 행 / 열의 구멍들은 서로 연결되어 있음
- 가운데 분리 영역 DIL 패키징의 칩 사용



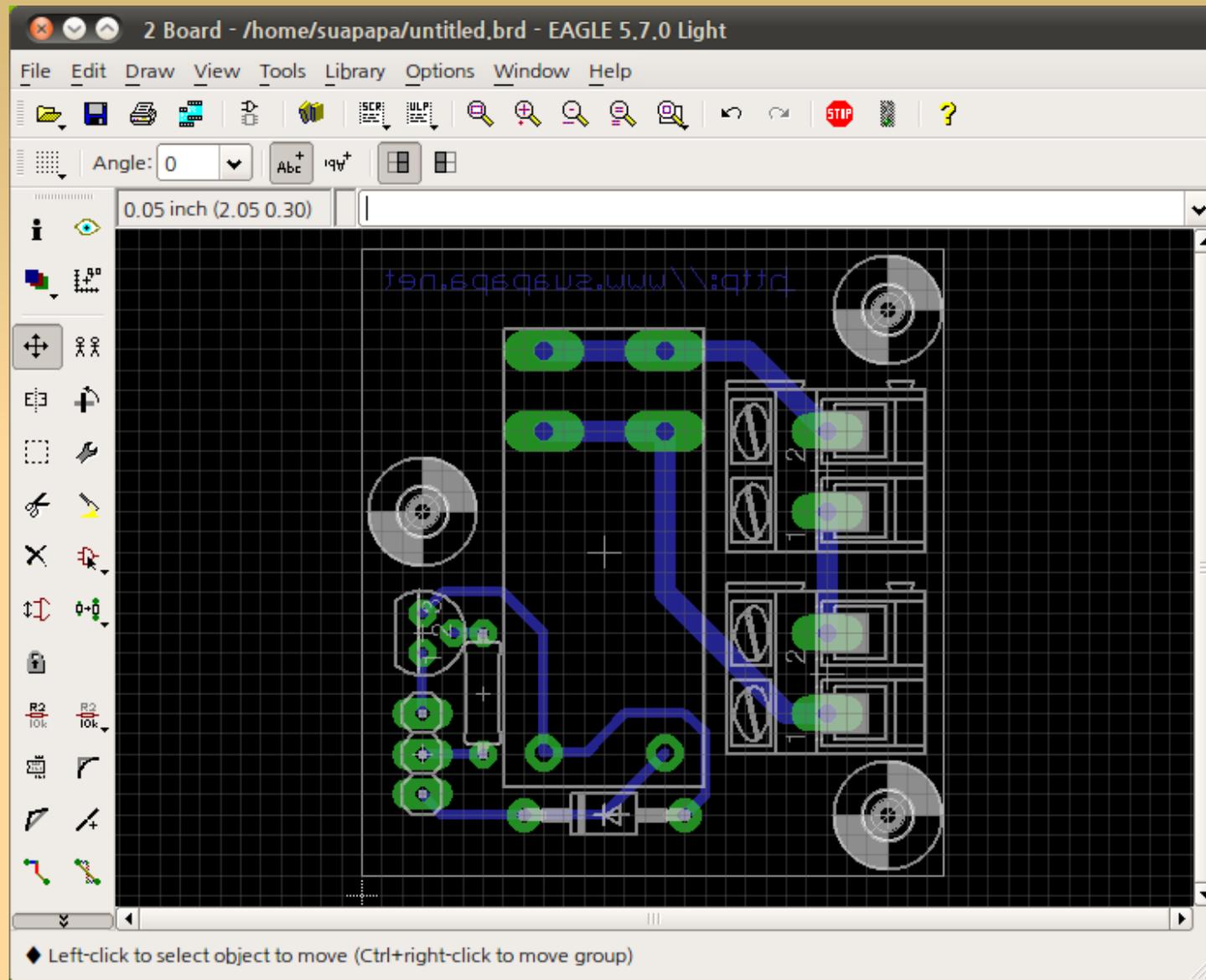
# EagleCad 1/3 - 소개

- <http://www.cadsoftusa.com/>
- 설치
  - `$ sudo apt-get install eaglecad`
- 회로설계 - sch
- 아트웍 - brd
  - gerber 파일로 출력 가능
- 라이브러리 - lbr

# EagleCad 2/3 - 회로설계



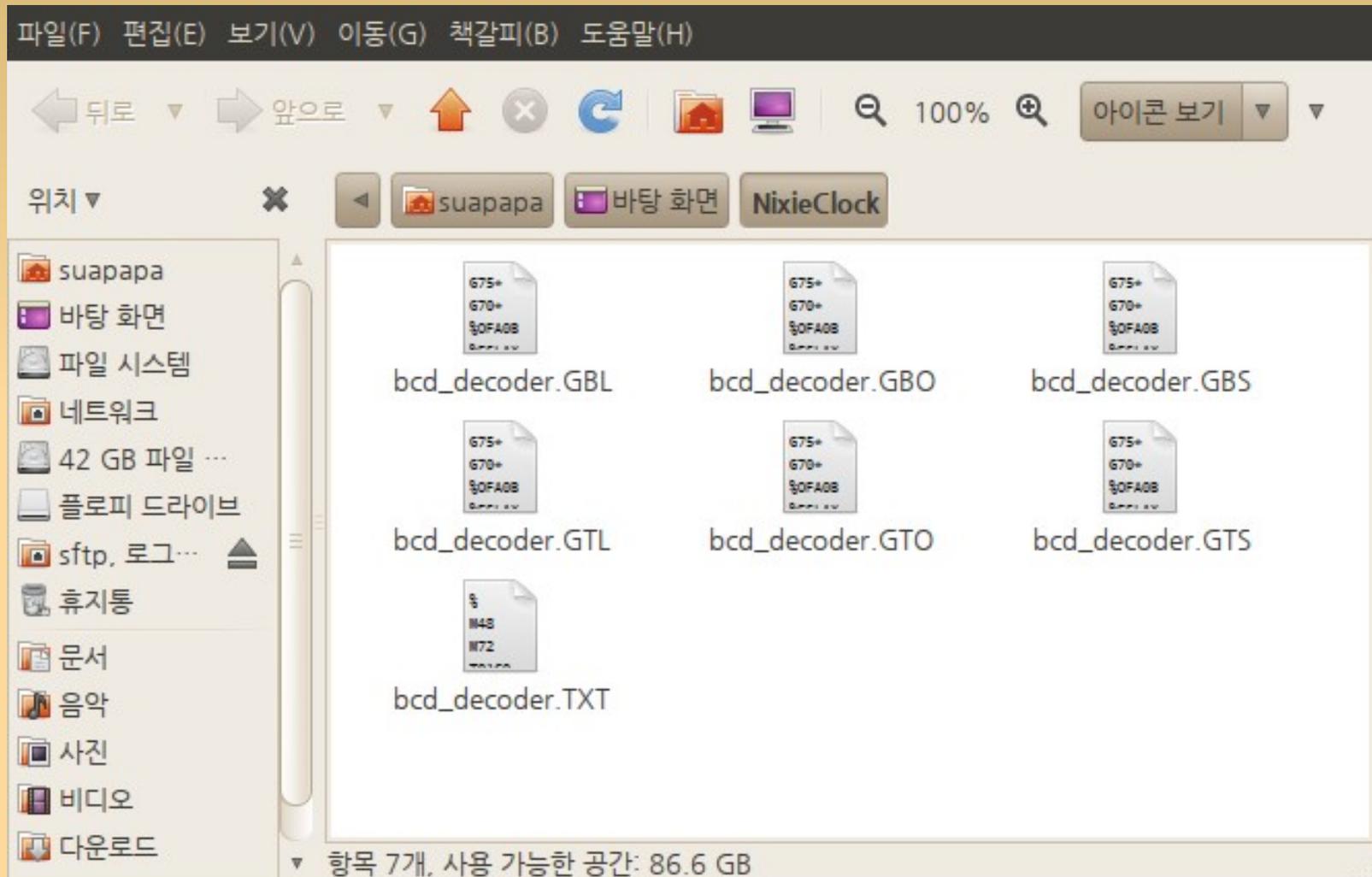
# EagleCad 3/3 – 아트웍



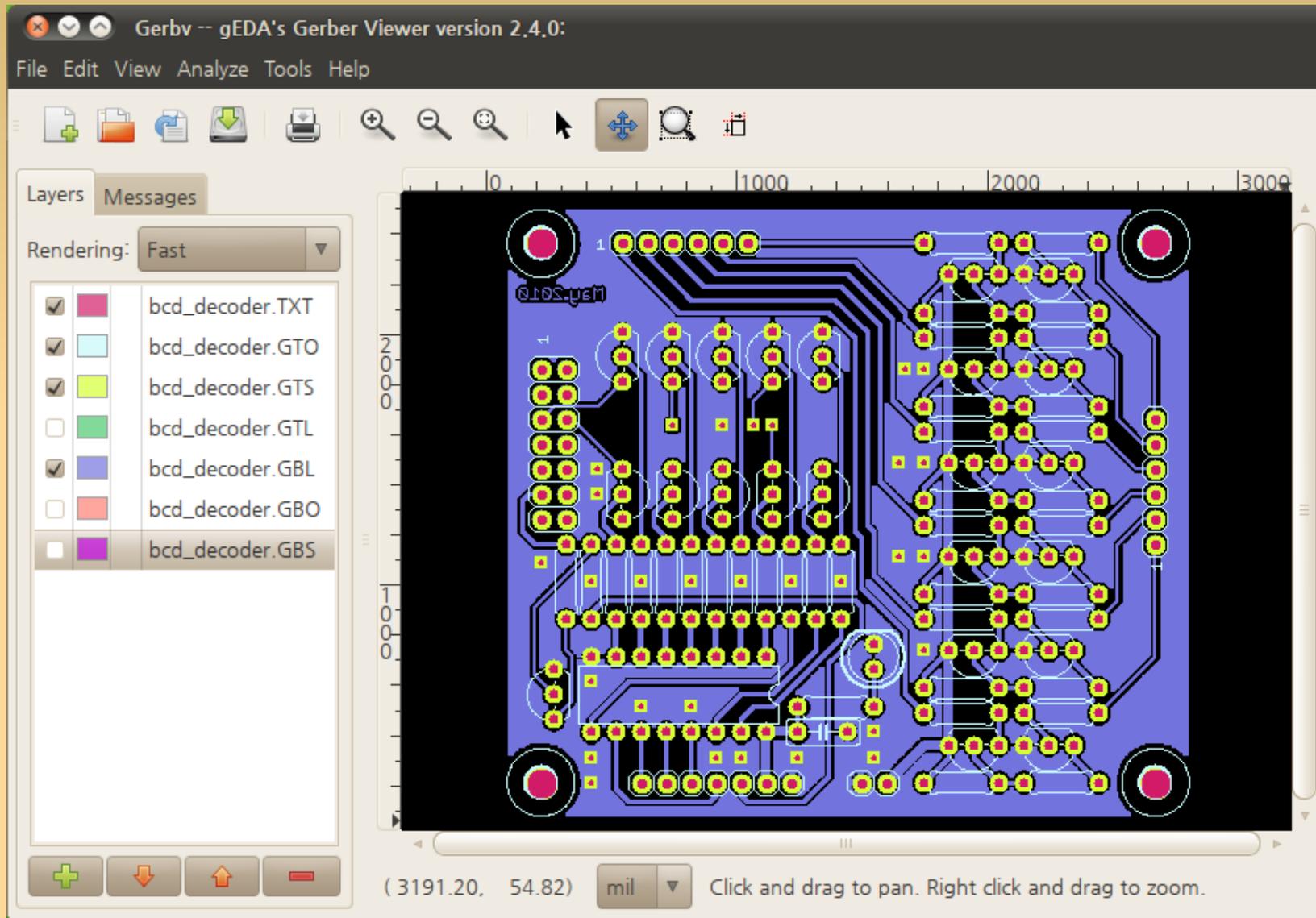
# Gerbv 1/3 – Gerber Viewer

- Gerber 는 회로도 의 PDF 파일
- 설치
  - `$ sudo apt-get install gerbv`
- 레이어 별로 파일 생성
  - GBL, GBO, GBS – bottom layer
  - GTL, GTO, GTS – top layer
  - TXT - drill
- 압축하여 PCB 제조 (fab) 업체에 주문

# Gerbv 2/3



# Gerbv 3/3



# Home – fab 1/3



- Export layer
  - Bottom or Top Layer
  - PAD, VIA, Route, Dimention
- 색반전 후 출력
  - PressNPeel 필름
- 다리미로 동판에 패턴 인쇄

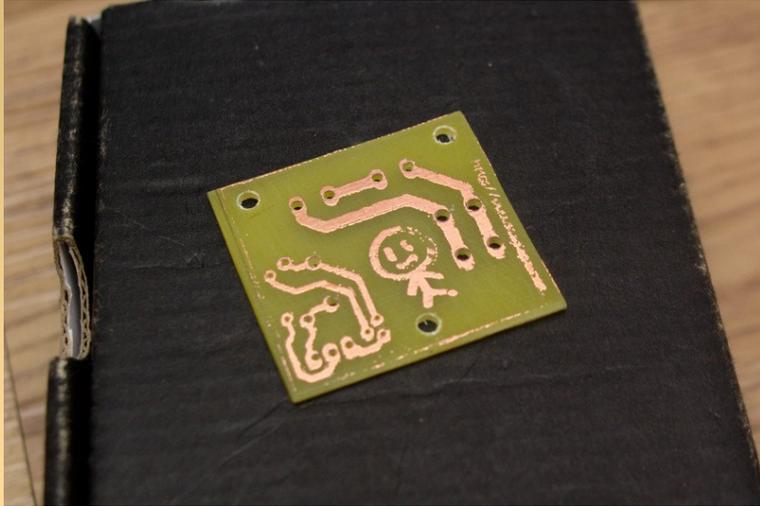
# Home – fab 2/3



- 필름을 벗겨낸 후
- 유성팬으로 패턴 수정
- 에칭액을 사용해 에칭



# Home – fab 3/3

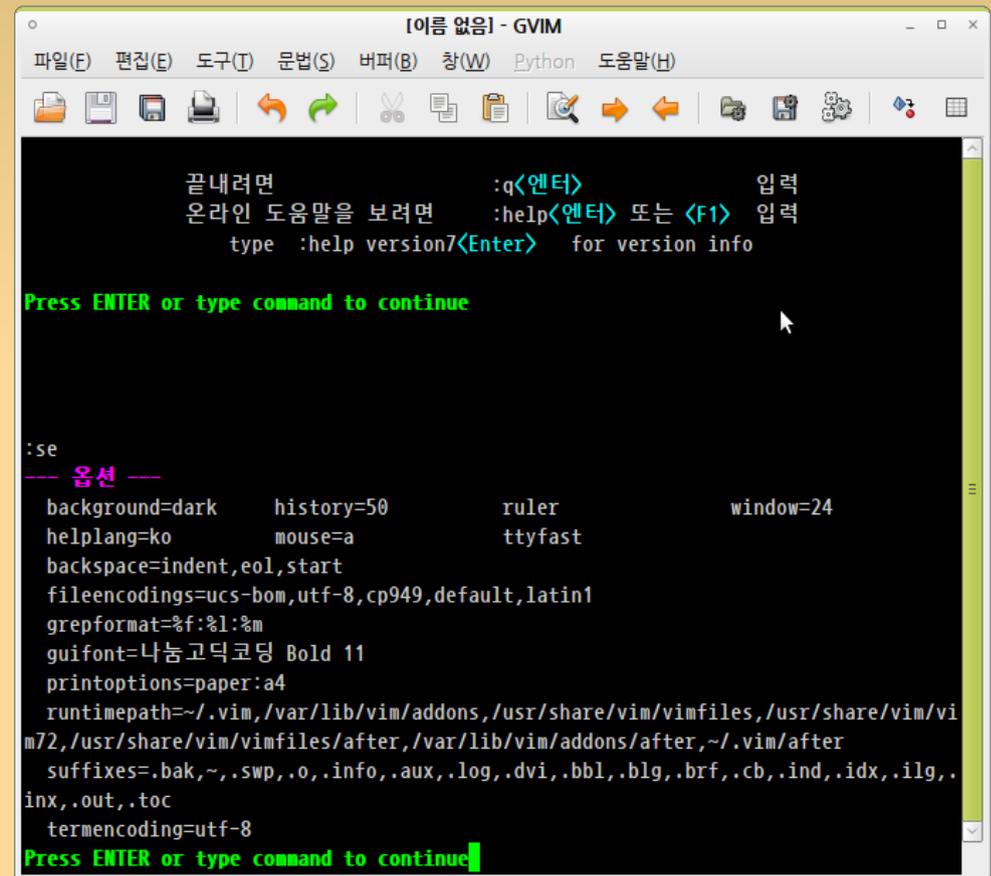


- 마스크를 벗겨냄
- 드릴링
- 전자부품을 조립 / 완성



# Vim - 소개

- 설치 (gvim)
  - \$ sudo apt-get install vim-gnome
- GUI 메뉴로 폰트 변경 후 현재 설정 확인
  - :set
- 기본 폰트 설정
  - ~/.vimrc
  - set gfn= 나눔고딕코딩 \ Bold\ 10



The screenshot shows the GVIM GUI window titled "[이름 없음] - GVIM". The menu bar includes "파일(F)", "편집(E)", "도구(T)", "문법(S)", "버퍼(B)", "창(W)", "Python", and "도움말(H)". The toolbar contains various icons for file operations and editing. The main editor area has a black background with green text. It displays instructions for exiting (:q<Enter>) and viewing help (:help<Enter> or <F1>). Below this, it shows the output of the :set command, listing various options such as background=dark, history=50, ruler, window=24, and guifont=나눔고딕코딩 Bold 11. The prompt "Press ENTER or type command to continue" is visible at the bottom.

```
[이름 없음] - GVIM
파일(F) 편집(E) 도구(T) 문법(S) 버퍼(B) 창(W) Python 도움말(H)
[Icons]
끝내려면 :q<엔터> 입력
온라인 도움말을 보려면 :help<엔터> 또는 <F1> 입력
type :help version7<Enter> for version info

Press ENTER or type command to continue

:set
--- 옵션 ---
background=dark    history=50          ruler                window=24
helplang=ko        mouse=a            ttyfast
backspace=indent,eol,start
fileencodings=ucs-bom,utf-8,cp949,default,latin1
grepformat=%f:%l:%m
guifont=나눔고딕코딩 Bold 11
printoptions=paper:a4
runtimepath=~/.vim,/var/lib/vim/addons,/usr/share/vim/vimfiles,/usr/share/vim/vi
m72,/usr/share/vim/vimfiles/after,/var/lib/vim/addons/after,~/.vim/after
suffixes=.bak,~, .swp,.o,.info,.aux,.log,.dvi,.bbl,.blg,.brf,.cb,.ind,.idx,.ilg,.
inx,.out,.toc
termencoding=utf-8
Press ENTER or type command to continue
```

# Vim - 사용

- 내장 도움말
  - :help
- Graphical vi-vim Cheat Sheet and Tutorial
  - [http://www.viemu.com/a\\_vi\\_vim\\_graphical\\_cheat\\_sheet\\_tutorial.html](http://www.viemu.com/a_vi_vim_graphical_cheat_sheet_tutorial.html)

# ctags & cscope

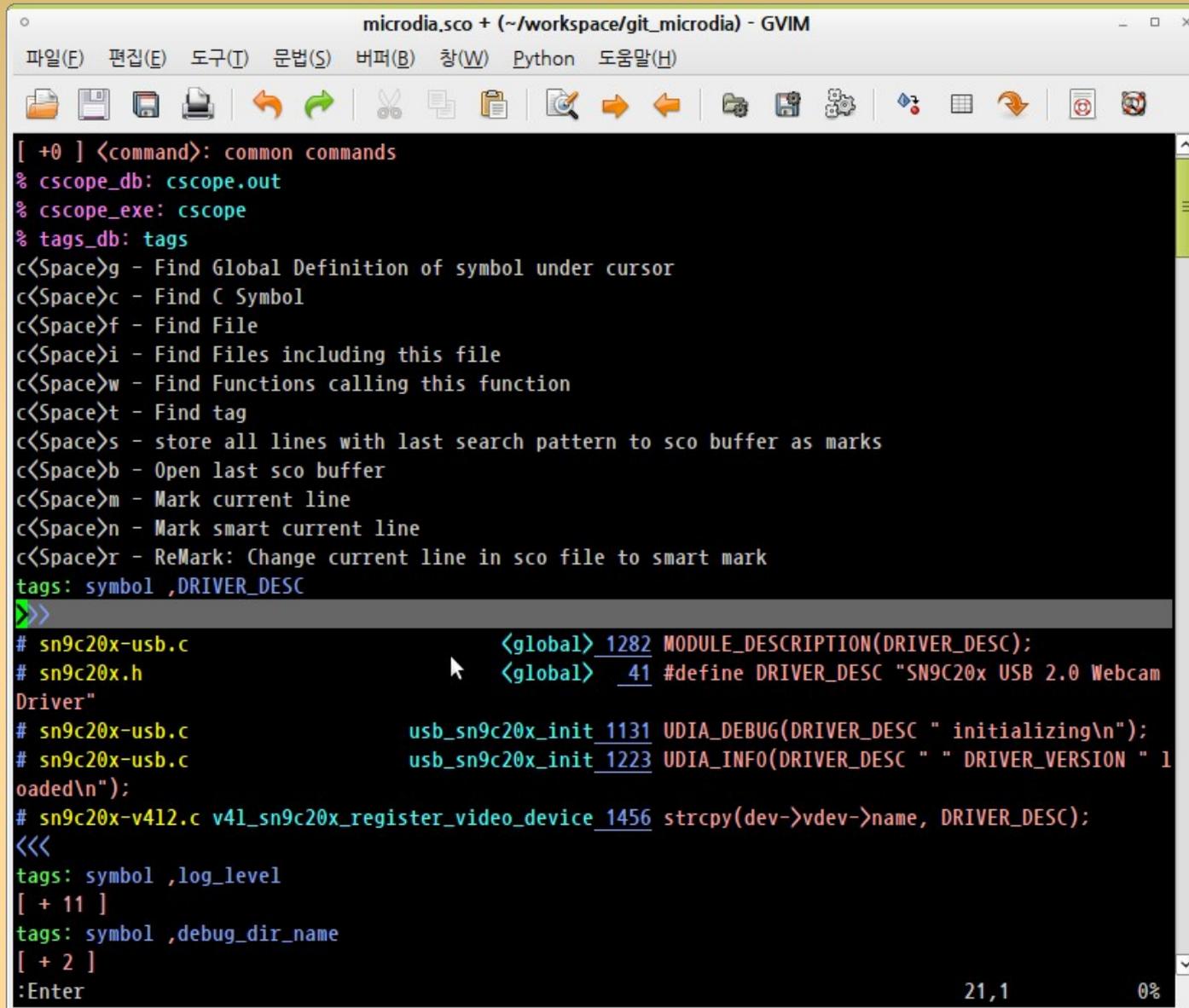
- ctags
  - <http://ctags.sourceforge.net/>
  - 소스의 심볼들을 빠르게 검색
- cscope
  - <http://cscope.sourceforge.net/>
  - 함수 호출 검색
  - 정규식으로 파일 검색
  - 파일을 포함한 파일 , etc...

# ctags & cscope – vim 연동 1/2

- SourceCodeObedience 플러그인 설치
  - [http://www.vim.org/scripts/script.php?script\\_id=1638](http://www.vim.org/scripts/script.php?script_id=1638)
  - 설치 - ~/.vim/plugin 폴더에 압축 해제
- ctags, cscope DB 생성

```
#!/bin/bash
SOURCE_LIST=source.list
rm -rf cscope.out tags
find . \( -name '*.c' -o -name '*.cpp' -o -name '*.cc' -o -name '*.h' -o
-name '*.s' -o -name '*.S' \) -print > $SOURCE_LIST
cscope -b -i $SOURCE_LIST
ctags -L $SOURCE_LIST
```

# ctags & cscope - vim 연동 2/2



```
microdia.sco + (~/.workspace/git_microdia) - GVIM
파일(F) 편집(E) 도구(T) 문법(S) 버퍼(B) 창(W) Python 도움말(H)
[ +0 ] <command>: common commands
% cscope_db: cscope.out
% cscope_exe: cscope
% tags_db: tags
c<Space>g - Find Global Definition of symbol under cursor
c<Space>c - Find C Symbol
c<Space>f - Find File
c<Space>i - Find Files including this file
c<Space>w - Find Functions calling this function
c<Space>t - Find tag
c<Space>s - store all lines with last search pattern to sco buffer as marks
c<Space>b - Open last sco buffer
c<Space>m - Mark current line
c<Space>n - Mark smart current line
c<Space>r - ReMark: Change current line in sco file to smart mark
tags: symbol ,DRIVER_DESC
>>
# sn9c20x-usb.c          <global> 1282 MODULE_DESCRIPTION(DRIVER_DESC);
# sn9c20x.h              <global> 41 #define DRIVER_DESC "SN9C20x USB 2.0 Webcam
Driver"
# sn9c20x-usb.c          usb_sn9c20x_init 1131 UDIA_DEBUG(DRIVER_DESC " initializing\n");
# sn9c20x-usb.c          usb_sn9c20x_init 1223 UDIA_INFO(DRIVER_DESC " " DRIVER_VERSION " 1
oaded\n");
# sn9c20x-v412.c v41_sn9c20x_register_video_device 1456 strcpy(dev->vdev->name, DRIVER_DESC);
<<<
tags: symbol ,log_level
[ + 11 ]
tags: symbol ,debug_dir_name
[ + 2 ]
:Enter
21,1 0%
```

# arm-cross-toolchain

- 크로스 툴 체인 빌드
  - <http://www.kegel.com/crosstool/>
- 기 빌드된 툴 체인 설치
  - CodeSourcery, etc...
  - 적당한 곳에 압축 해제 후 사용
- 다음과 같이 설치 위치 PATH 추가
  - `$ vi ~/.bashrc`
  - `PATH=$PATH:/usr/local/arm/arm-2009q3/bin`

# Arm 타겟 으로 포팅하기 1/3

- 기존 target → Makefile 수정

```
$ make
cc -c mkdosfs.c -o mkdosfs.o
cc mkdosfs.o -o mkdosfs
$ file mkdosfs
mkdosfs: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.15, not
stripped
```

```
$ vi Makefile
CC=arm-none-linux-gnueabi-gcc
CFLAGS=--static
...
```

# Arm 타겟으로 포팅하기 2/3

```
$ make
```

```
arm-none-linux-gnueabi-gcc --static -c mkdosfs.c -o mkdosfs.o
```

```
arm-none-linux-gnueabi-gcc mkdosfs.o -o mkdosfs
```

```
$ file mkdosfs
```

```
mkdosfs: ELF 32-bit LSB executable, ARM, version 1 (SYSV),  
dynamically
```

```
linked (uses shared libs), for GNU/Linux 2.6.16, not stripped
```

# Arm 타겟으로 포팅하기 3/3

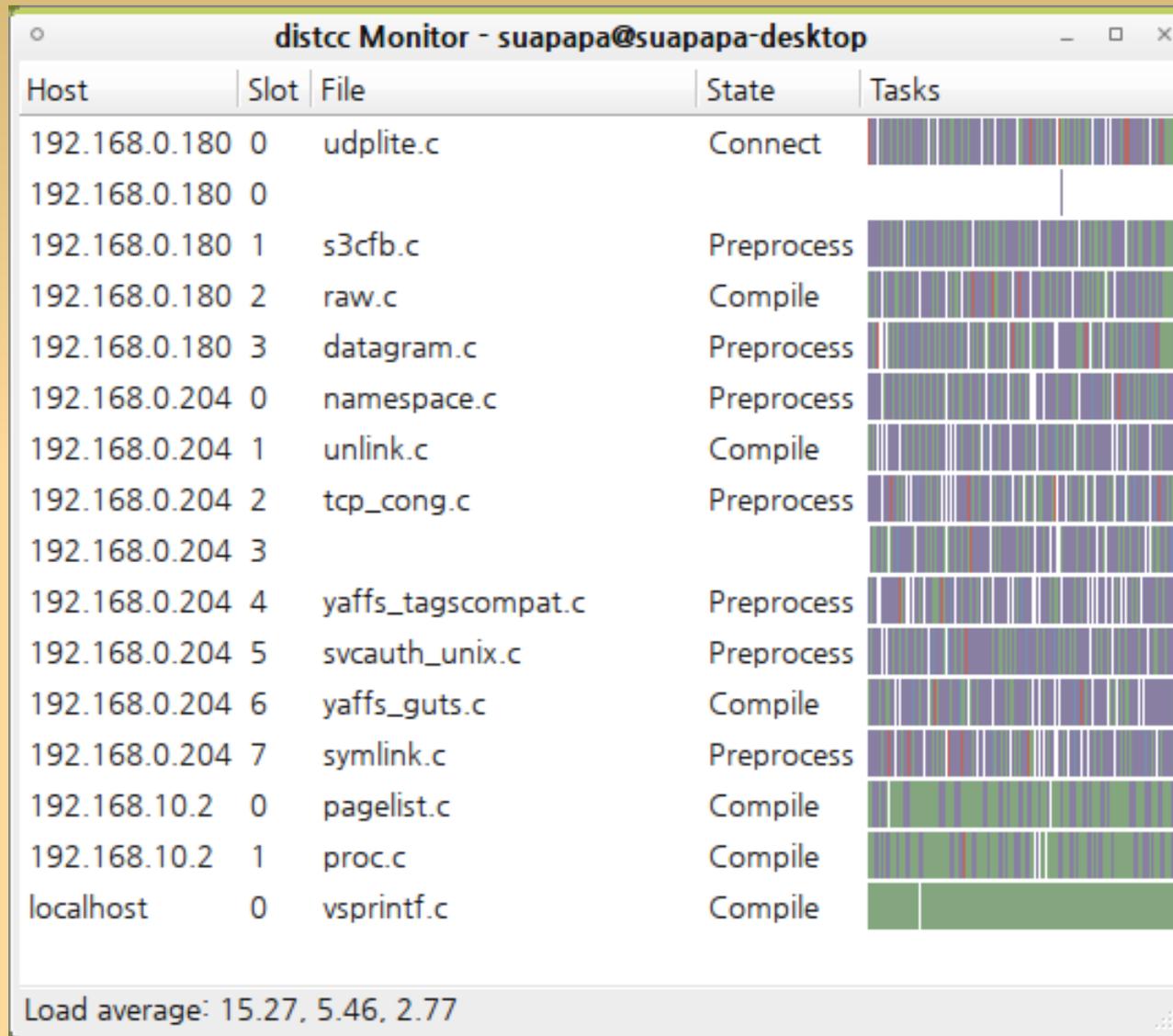
- 리눅스 커널의 Makefile 중 ...
  - CROSS\_COMPILE 변수로 틀체인 설정
  - \$ make ARCH=arm \  
CROSS\_COMPILE="arm-none-eabi-" -j3

```
# Make variables (CC, etc...)
AS      = $(CROSS_COMPILE)as
LD      = $(CROSS_COMPILE)ld
CC      = $(CROSS_COMPILE)gcc
CPP     = $(CC) -E
AR      = $(CROSS_COMPILE)ar
...
```

# Distcc - 분산 빌드 1/2

- 설치
  - `$ sudo apt-get install distcc distccmon-gnome`
- 서버 설정
  - `/etc/default/distcc, /etc/init.d/distcc`
- distcc 서버들 사용하기
  - `$ DISTCC_HOSTS="192.168.10.2/3 localhost/2"`  
`make -j 16`
  - `-j` 옵션으로 (프로세스 개수 \* 3 + 1)

# Distcc - 분산빌드 2/2



# Q&A

감사합니다 .