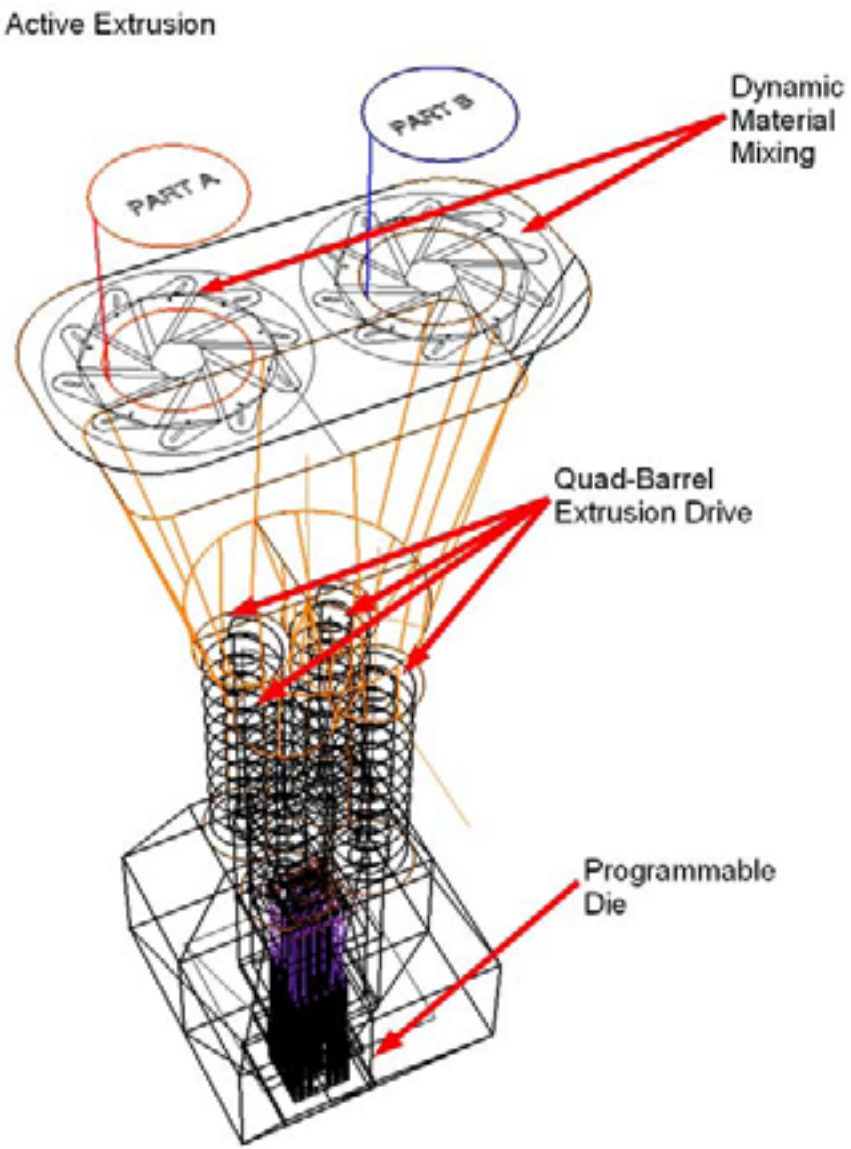
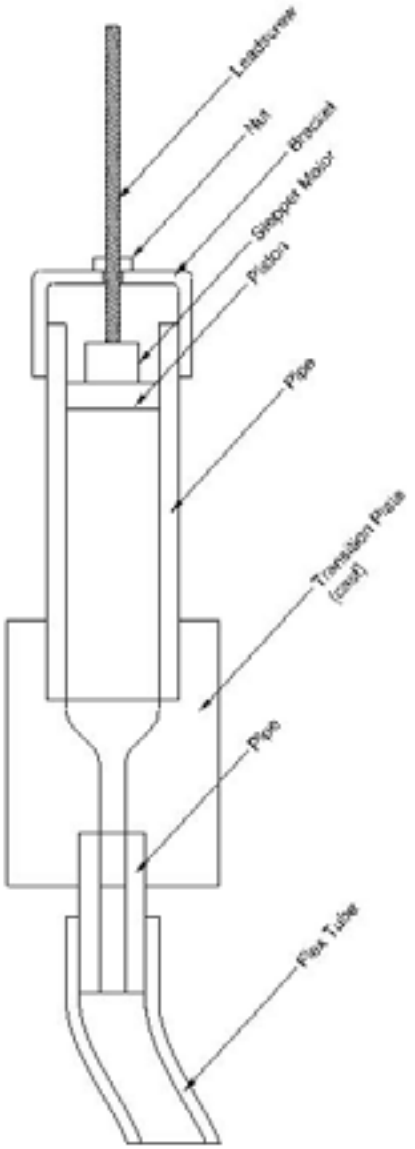
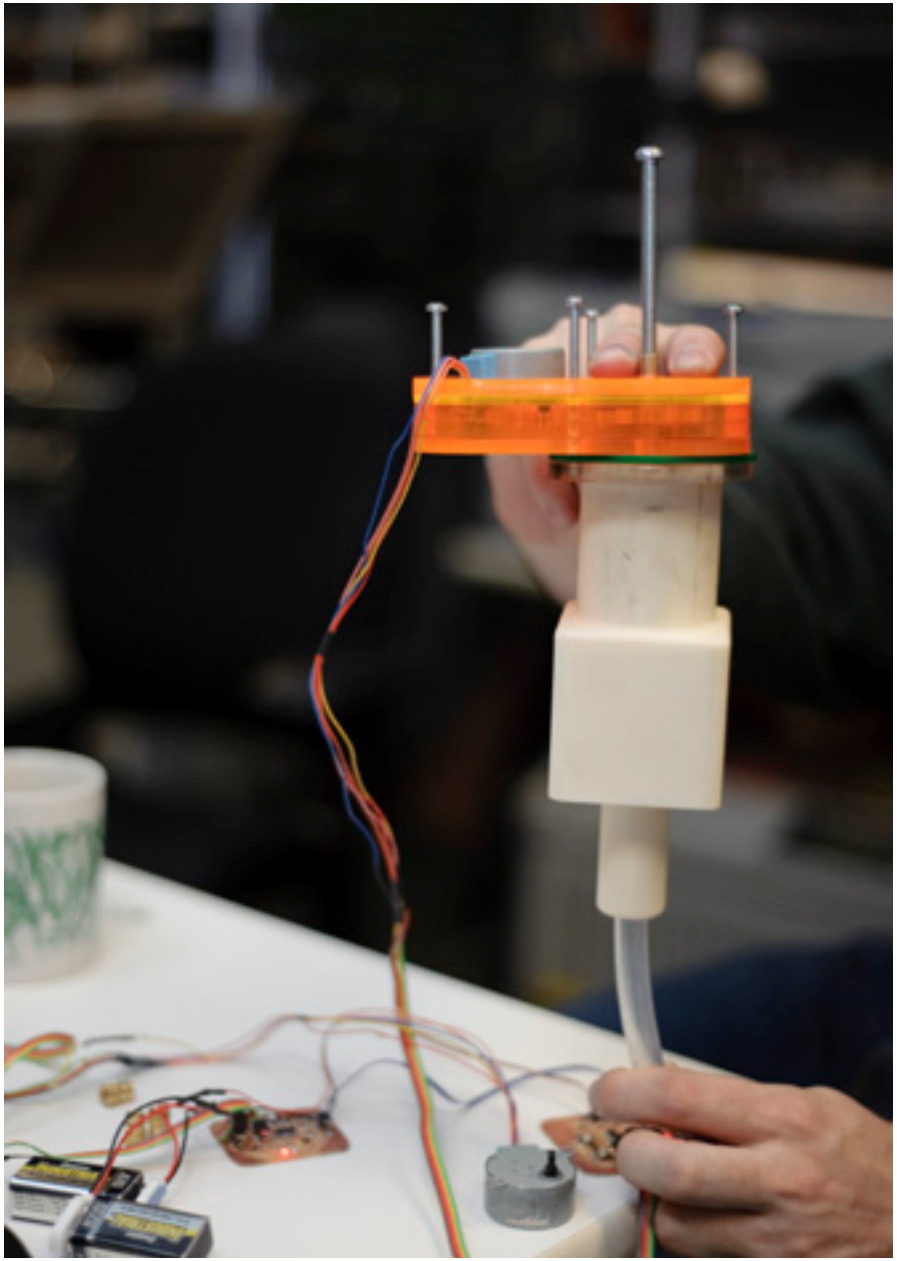

Extrusion machines

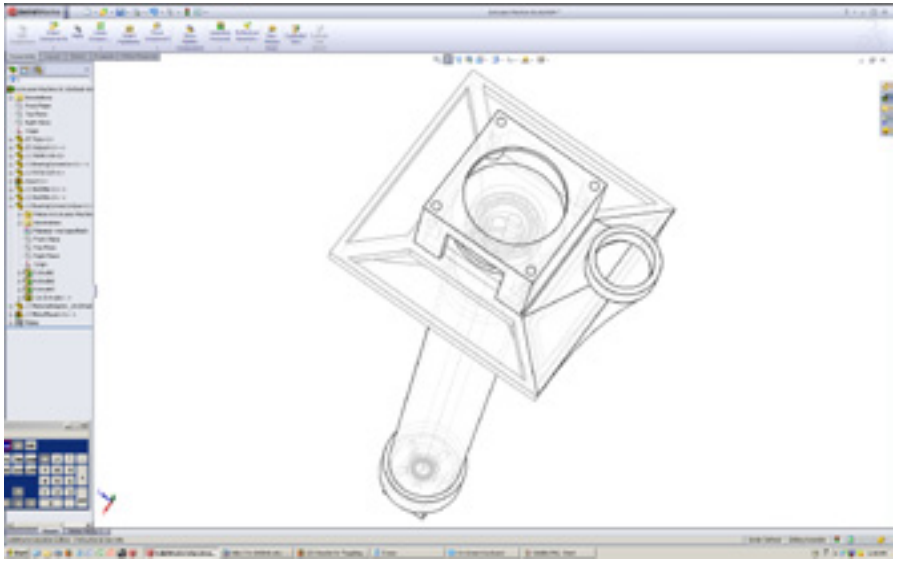


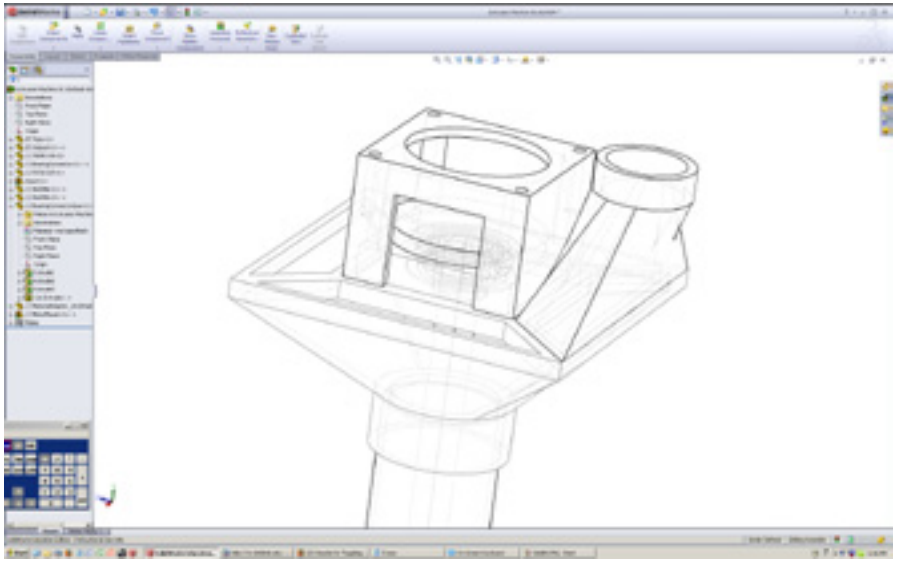


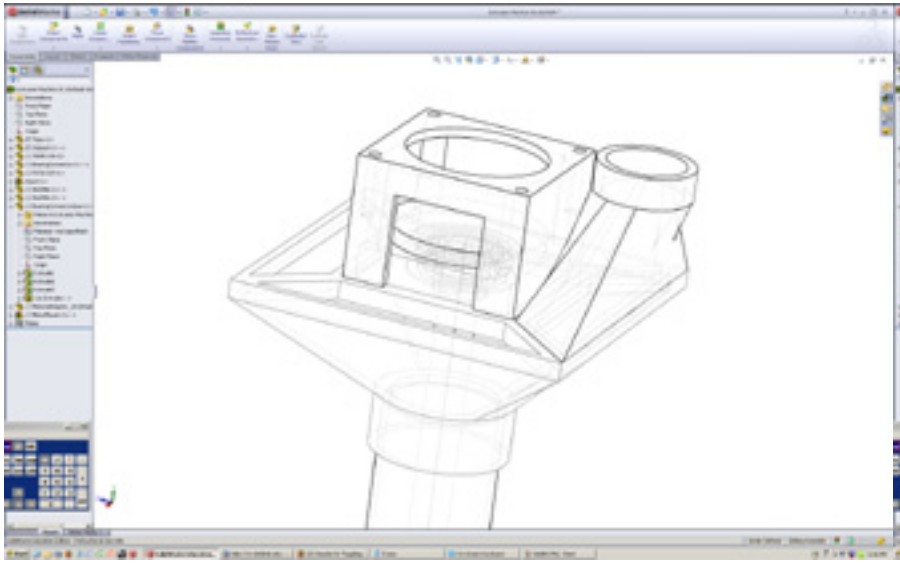
First designs

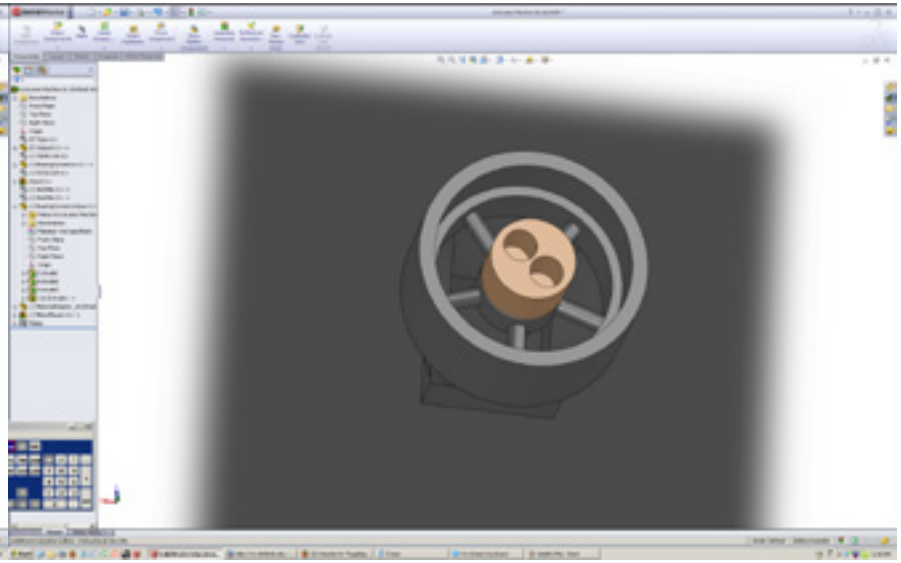


First prototypes

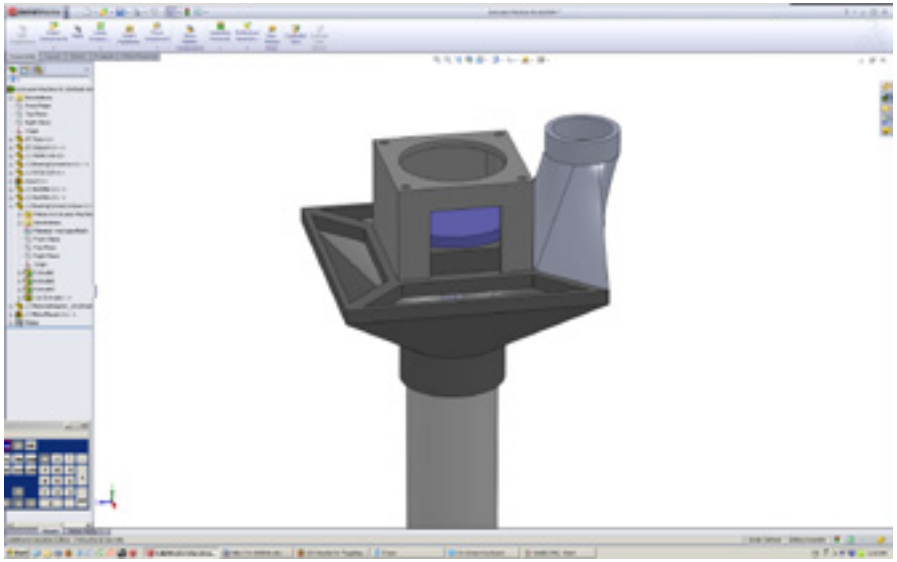


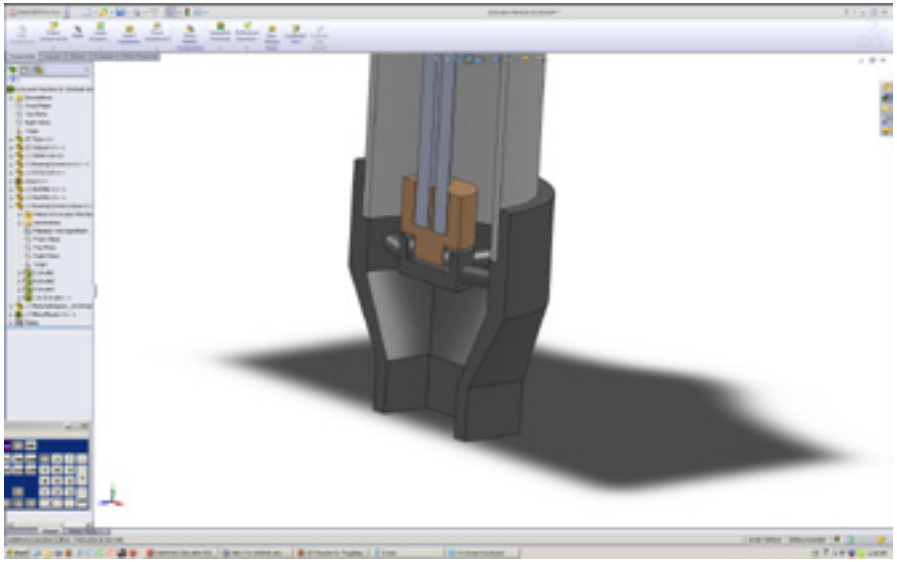


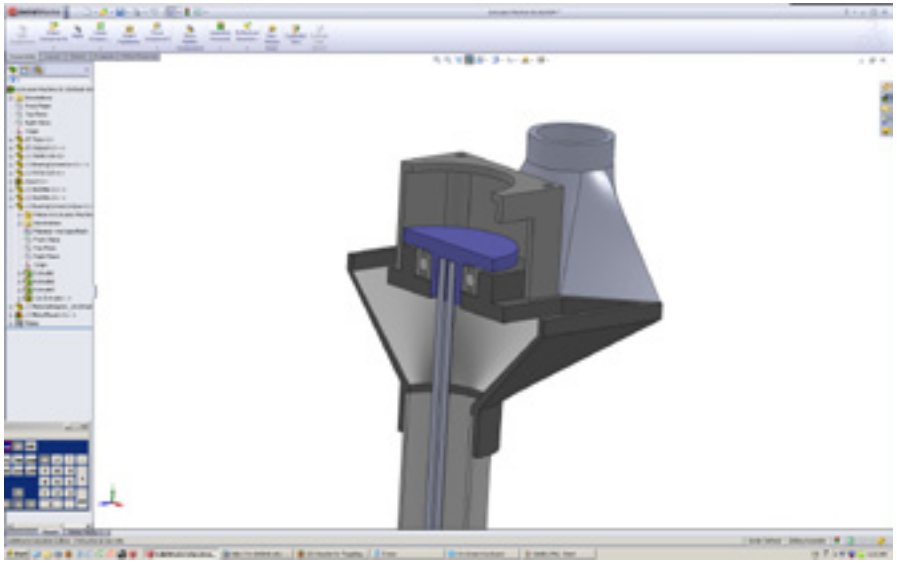




Final design / CAD model



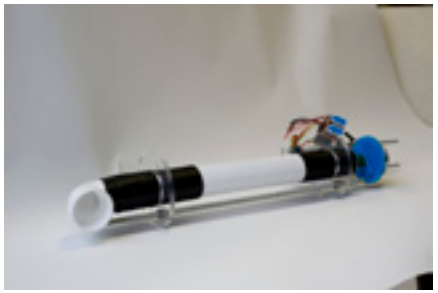




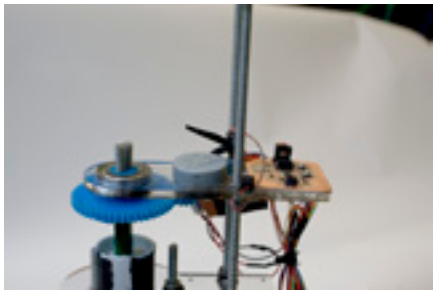
Final design / CAD model





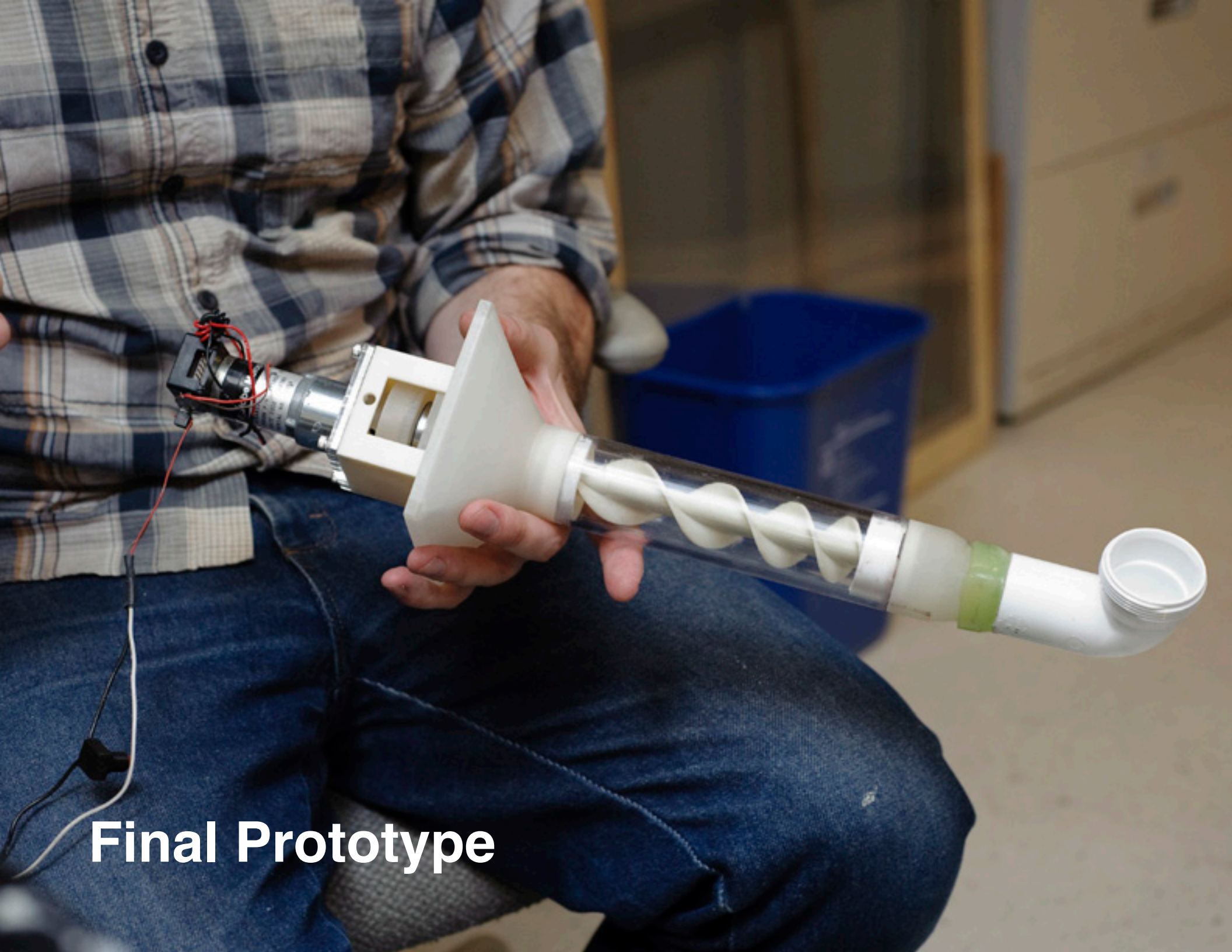






3D Prints and Assembly

Auger designs



Final Prototype


CERAMIC
CLAYSTONE
SELF-HARDENING CLAY
Betty Crocker
Potato Buds
100% REAL POTATOES
HODGSON MILL
AMACO
Vegetable
Corn Starch

MATERIAL

All tested materials except wheat flour.

Some results.

1. **GOAL = Finding cheap, easy to extrude material!**
2. **Additional Goal = Controlled curved extrusion.**

1. Plaster / Stone

- Gypsum
- Drystone
- Hydrostone Super X
- Tuf Stone

Instights:

- Separation between material and water
  --> not useful for extrusion processes

2. Clay

Clay Flour:

Attributes:

- Easy to mix
- Very soft like toothpaste
- Stays in shape

Goal material! - Unfortunately toxic !

2. Clay

Terra Cotta:

Attributes:

- Hard
- Brittle
- Not easy to mix with oil

2. Clay

Terra Cotta:

Attributes:

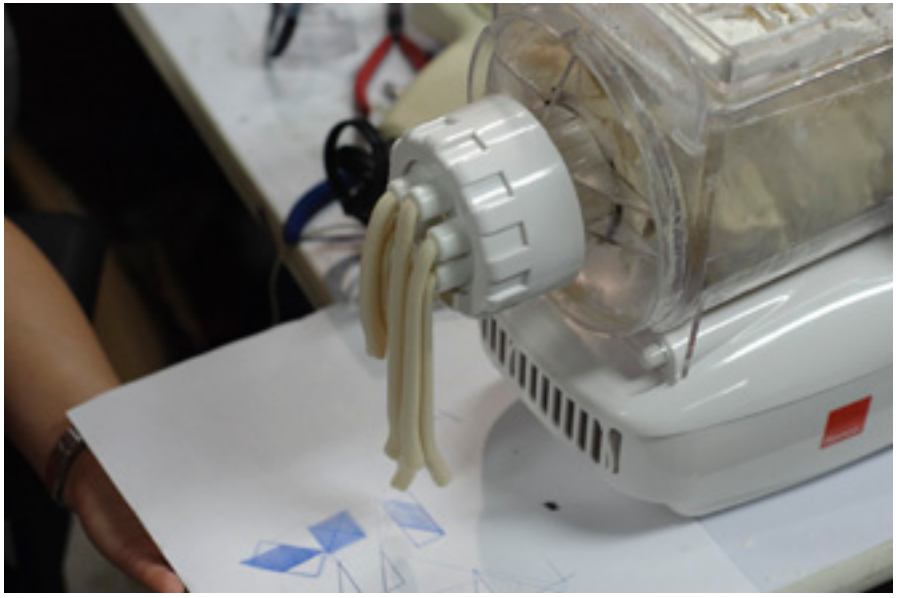
- Self-Hardened !!!
- Easy to mix with oil
- Can be mix to quite soft material
- Perfekt for curved extrusion

2. Clay

Self-made Clay:

Materials:

1. Flour
    - Makes material sticky
    - Encourages togetherness
2. Salt
    - Produces clay character
3. Water
    - Main matrix
4. Oil
    - Makes material smoother



3. Doug

Rice Flour:

Attributes:

- Not sticky

Attributes:

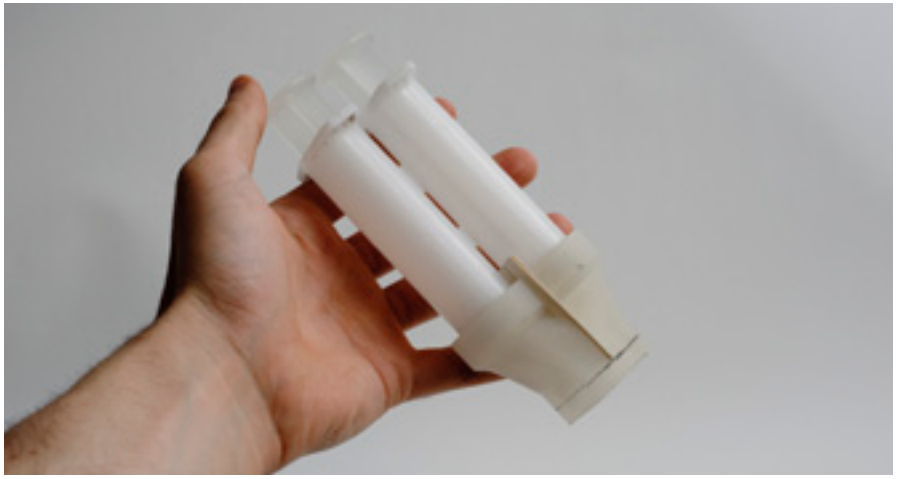
- Grainy
- Expands after extrusion

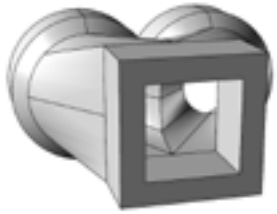
4. Mashed Potatos

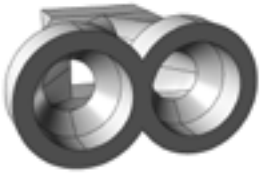
Attributes:

- Very soft and smooth
- Very light
- Very easy to extrude
- Retains shape
- Used in active die

Goal material!



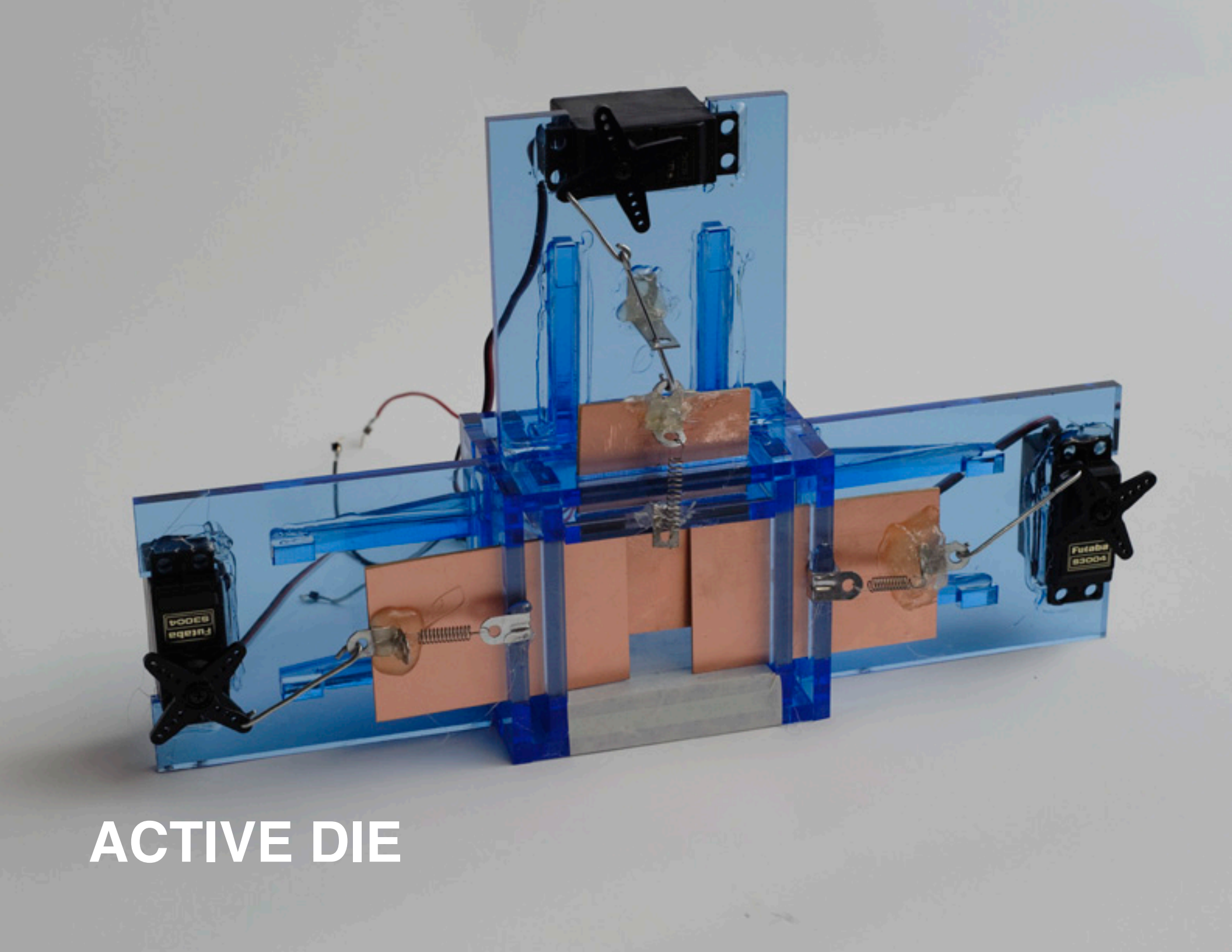




2. Additional goal

Curved extrusion through varied pressure units.

Material: Self-hardened Clay!



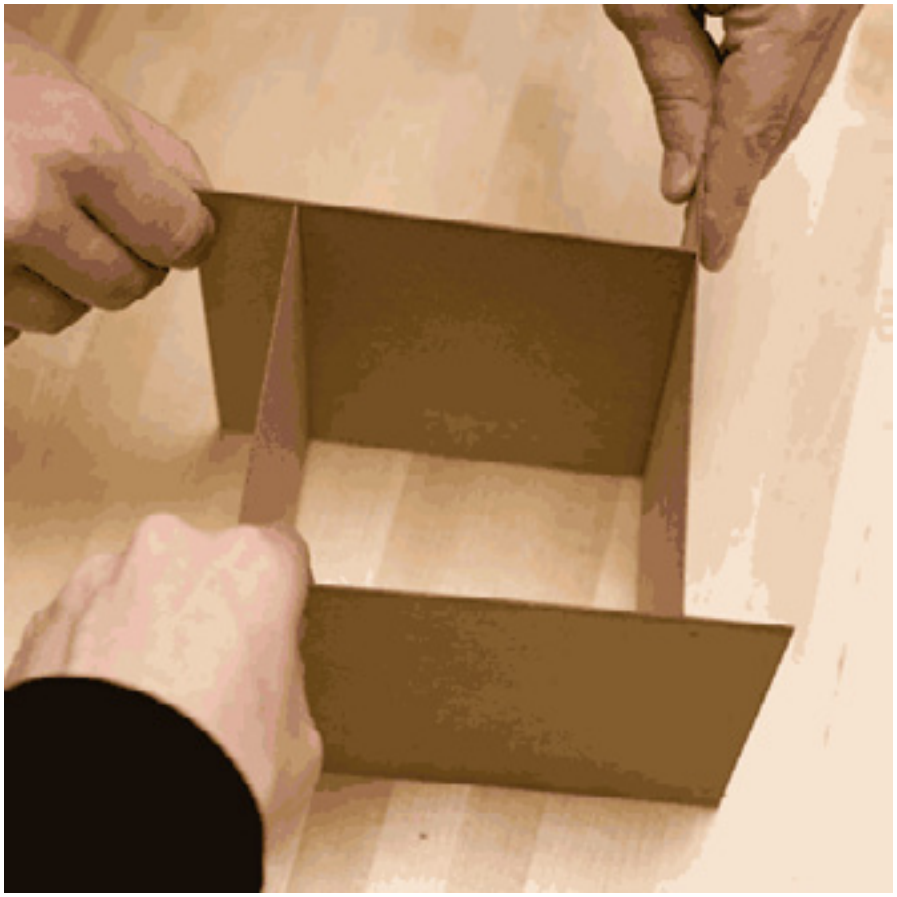
ACTIVE DIE

1. GOAL: Controlled, customized shaped extrusion!

2. Additional Goal: User interface!

Horizontal aperture:

- Strong pressure on blades
- Easy to connect



Vertical aperture:

- Less pressure on blades
- Difficult connection

Arduino code to serial port

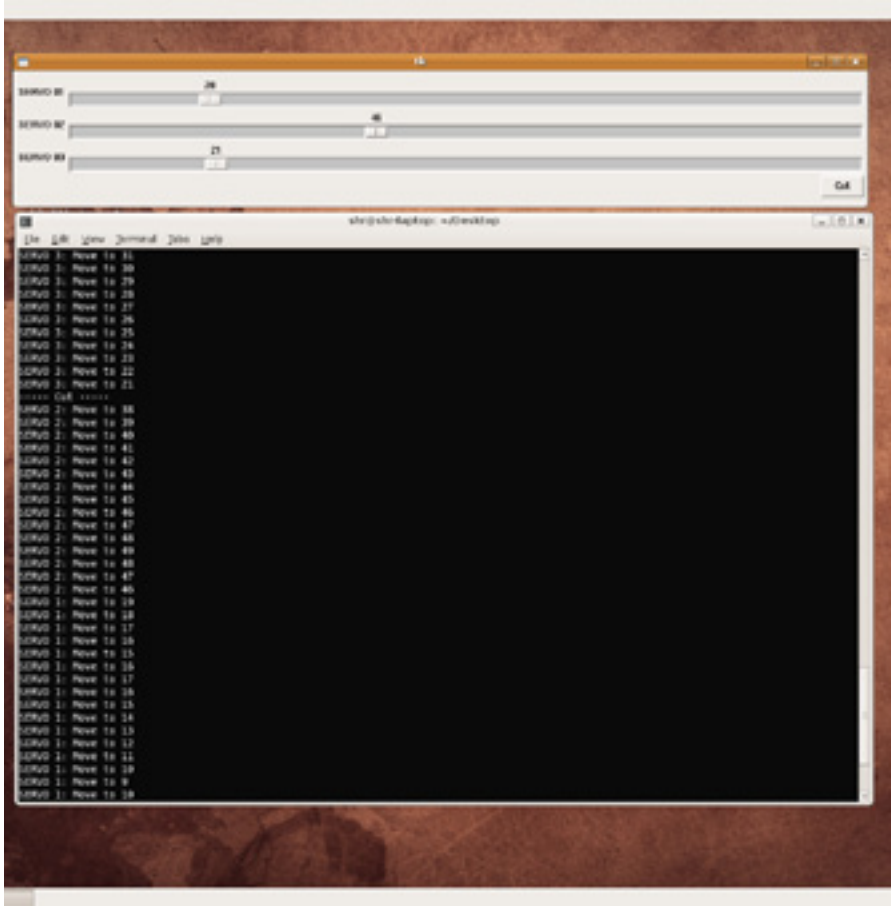
```
/*
* MultipleServos
* --------------
* Arduino servo control from a PC
*
* Created: 2 April 2008
* Author:  Brian D. Wendt
*   http://principialabs.com/
* License: GPLv3, copyleft 2008
*   http://www.fsf.org/licensing/
*
* Adapted from code by Tom Igoe
*   http://itp.nyu.edu/physcomp/Labs/Servo
*/

/** Adjust these values for your servo and setup, if necessary **/
int pinArray[4] = {2, 3, 4, 5}; // digital pins for the servos
int minPulse = 600;            // minimum servo position
int maxPulse = 2400;           // maximum servo position
int refreshTime =  20;         // time (ms) between pulses (50Hz)

/** The Arduino will calculate these values for you **/
int i;              // iterator
int servoPin;       // control pin for current servo
int userInput[3];   // raw input from serial buffer, 3 bytes
int pulseWidth;     // servo pulse width
int servoPosition;  // commanded servo position, 0-180 degrees
int pulseRange;     // maxPulse - minPulse
int centerServo;    // servo starting point
long lastPulse = 0; // recorded time (ms) of the last pulse
int servo;          // which servo to pulse? 1-4
int servo1[2];      // servo #1 array{pin, pulsewidth}
int servo2[2];      // servo #2 array{pin, pulsewidth}
int servo3[2];      // servo #3 array{pin, pulsewidth}
int servo4[2];      // servo #4 array{pin, pulsewidth}
int pin;            // digital pin for pulse() function
int puls;           // pulsewidth for pulse() function
int startbyte;      // start byte, begin reading input

void setup() {
 // loop through all 4 servo pins
 // and set them as OUTPUT
 for (i=0;i<4;i++) {
   pinMode(pinArray[i], OUTPUT);
 }
 // servo starting point (center)
 pulseRange  = maxPulse - minPulse;
 centerServo = maxPulse - ((pulseRange)/2);
 pulseWidth  = centerServo;
 // map pins to servos
 servo1[0] = pinArray[0];  // servo #1 is pin 2
 servo2[0] = pinArray[1];  // servo #2 is pin 3
 servo3[0] = pinArray[2];  // servo #3 is pin 4
 servo4[0] = pinArray[3];  // servo #4 is pin 5
 // center all servos
 servo1[1] = pulseWidth;
 servo2[1] = pulseWidth;
 servo3[1] = pulseWidth;
 servo4[1] = pulseWidth;
 // open serial connection
 Serial.begin(9600);
}

void loop() {
 // wait for serial input (min 3 bytes in buffer)
 if (Serial.available() > 2) {
   //read the first byte
   startbyte = Serial.read();
   // if it's really the startbyte (255)
   if (startbyte == 255) {
     // then get the next two bytes
     for (i=0;i<2;i++) {
       userInput[i] = Serial.read();
     }
     // first byte = servo to move?
     servo = userInput[0];
     // second byte = which position?
     servoPosition = userInput[1];
     // packet check
     if (servoPosition == 255) { servo = 255; }
     // compute pulseWidth from servoPosition
     pulseWidth = minPulse + (servoPosition * (pulseRange/180));
     // stop servo pulse at min and max
     if (pulseWidth > maxPulse) { pulseWidth = maxPulse; }
     if (pulseWidth < minPulse) { pulseWidth = minPulse; }
     // assign new pulsewidth to appropriate servo
     switch (servo) {
       case 1:
         servo1[1] = pulseWidth;
         break;
       case 2:
         servo2[1] = pulseWidth;
         break;
       case 3:
         servo3[1] = pulseWidth;
         break;
       case 4:
         servo4[1] = pulseWidth;
         break;
     }
   }
 }
 // pulse each servo
 if (millis() - lastPulse >= refreshTime) {
   pulse(servo1[0], servo1[1]);
   pulse(servo2[0], servo2[1]);
   pulse(servo3[0], servo3[1]);
   pulse(servo4[0], servo4[1]);
   // save the time of the last pulse
   lastPulse = millis();
 }
}

void pulse(int pin, int puls) {
   digitalWrite(pin, HIGH); // start the pulse
   delayMicroseconds(puls); // pulse width
   digitalWrite(pin, LOW);  // stop the pulse
}
```

Python servo serial file

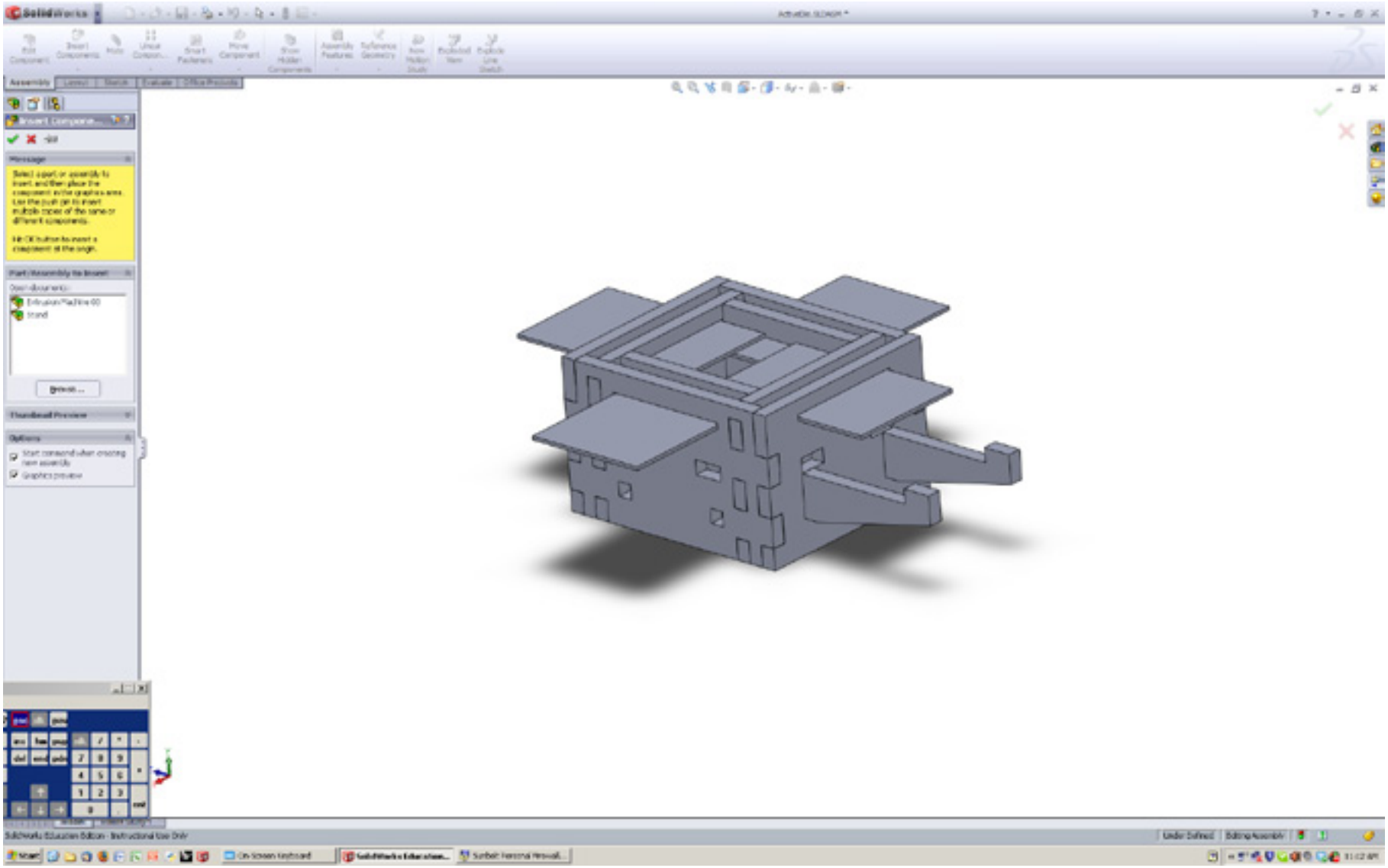
```
#!/usr/bin/env python

############################################
# Module:   servo.py
# Created:  2 April 2008
# Author:   Brian D. Wendt
#   http://principialabs.com/
# Version:  0.2
# License:  GPLv3
#   http://www.fsf.org/licensing/
'''
Provides a serial connection abstraction layer
for use with Arduino "MultipleServos" sketch.
'''
############################################

import serial

usbport = '/dev/ttyUSB0'
ser = serial.Serial(usbport, 9600, timeout=1)
print ser

def move(servo, angle):
  '''Moves the specified servo to the supplied angle.

  Arguments:
    servo
      the servo number to command, an integer from 1-4
    angle
      the desired servo angle, an integer from 0 to 180

  (e.g.) >>> servo.move(2, 90)
         ... # "move servo #2 to 90 degrees"''

  if (0 <= angle <= 180):
    ser.write(chr(255))
    ser.write(chr(servo))
    ser.write(chr(angle))
  else:
    print "Servo angle must be an integer between 0 and 180.\n"
```

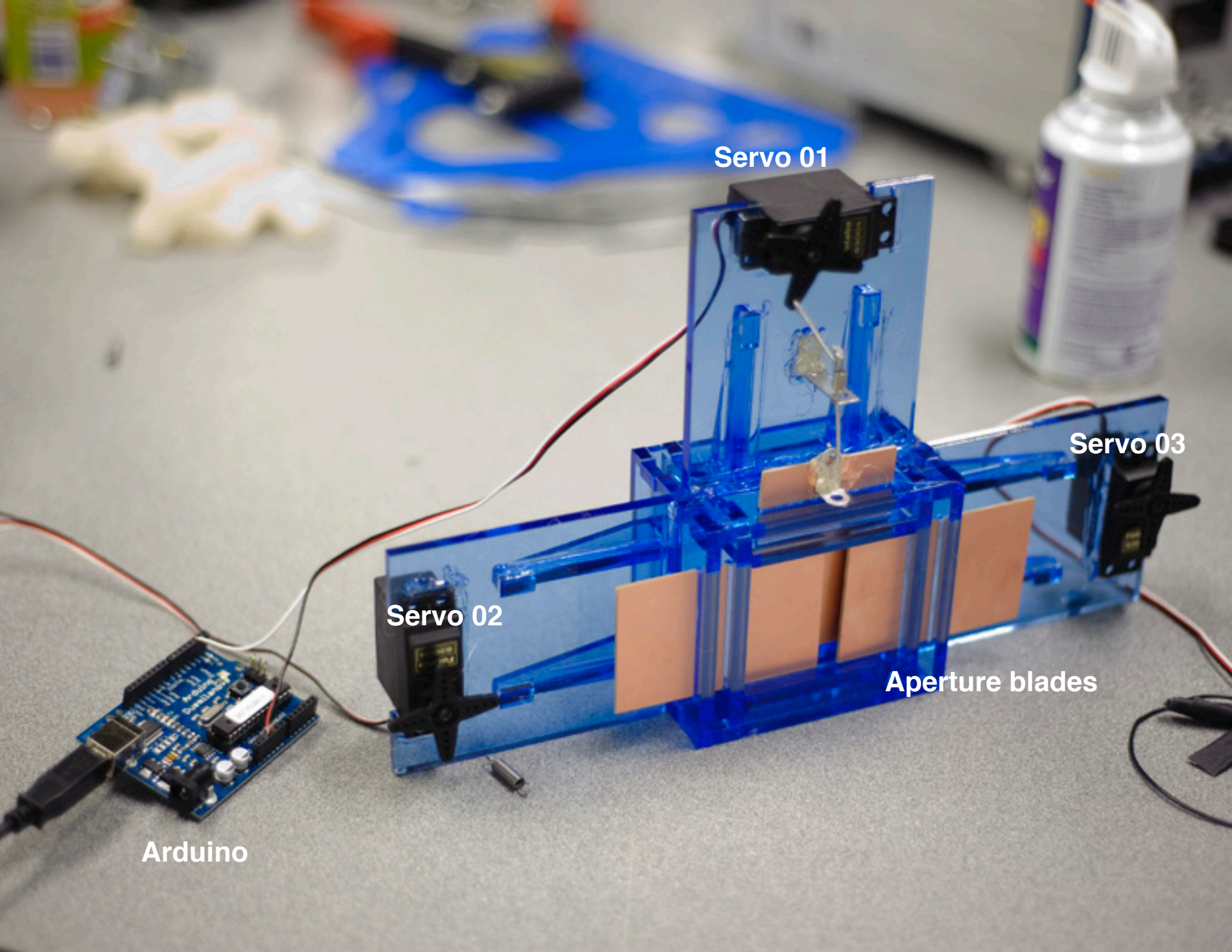
die.py - Interface for active extrusion

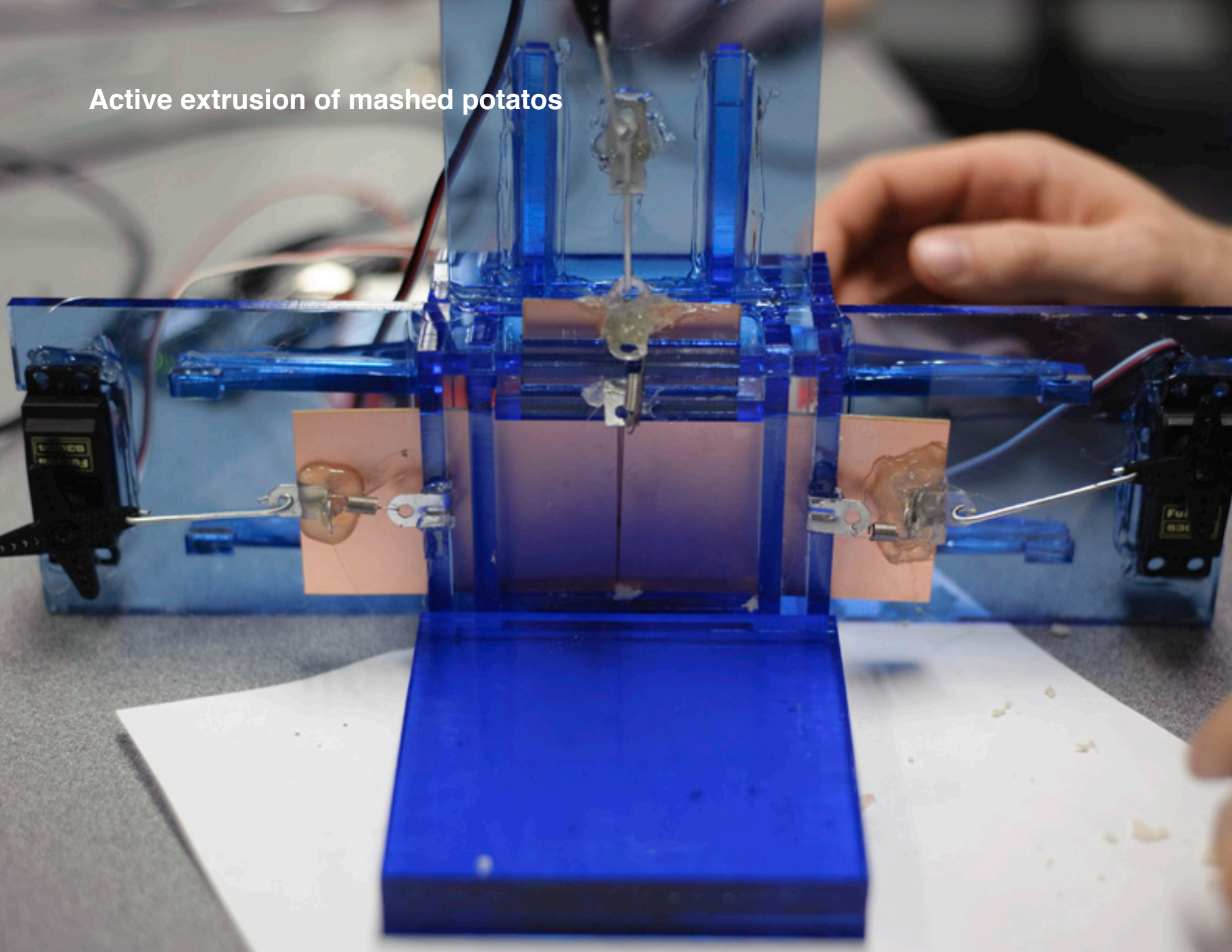
```
#!/usr/bin/env python

#
# pie.py
#
# Forrest Green and Steffen Reichert
#
# (c) Massachusetts Institute of Technology 2009
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#


import servo
from Tkinter import *
import Tkinter
import time

class servoControl:

		def cut(self):
				print „----- Cut -----“
				servo.move(self.cutServoIndex, 120)
				time.sleep(self.cutServoIndex)
				servo.move(self.cutServoIndex, self.cutServoSlider.get())

		def moveServo(self, index, angle):


				print „SERVO %d: Move to %d“ % (index, angle)
				servo.move(index, angle)

		def addServo(self, servoName, servoIndex):

				servoLabel = Tkinter.Label(self.root, text=servoName).grid(row=self.nextRow, column=0, sticky=W)
				servoSlider = Tkinter.Scale(self.root, from_=0, to=120,   orient=Tkinter.HORIZONTAL, \
				    command=lambda x: self.moveServo(servoIndex, int(x)))
				servoSlider.grid(row=self.nextRow, column=1, sticky=W+E)
				self.nextRow += 1
				if servoIndex == self.cutServoIndex:
						print „Setting cut servo slider“
						self.cutServoSlider=servoSlider
		def __init__(self):

				self.nextRow = 0
				#build root object
				self.root = Tkinter.Tk()
				self.root.grid_columnconfigure(1, weight=1)
				self.cutServoIndex = 1

				self.addServo(„SERVO 01“, 1)
				self.addServo(„SERVO 02“, 2)
				self.addServo(„SERVO 03“, 3)

				Tkinter.Button(self.root, text=“Cut“, command=self.cut).grid(row=self.nextRow, column=1, sticky=E)
				self.nextRow += 1

				#self.root.repack()
		def run(self):
```

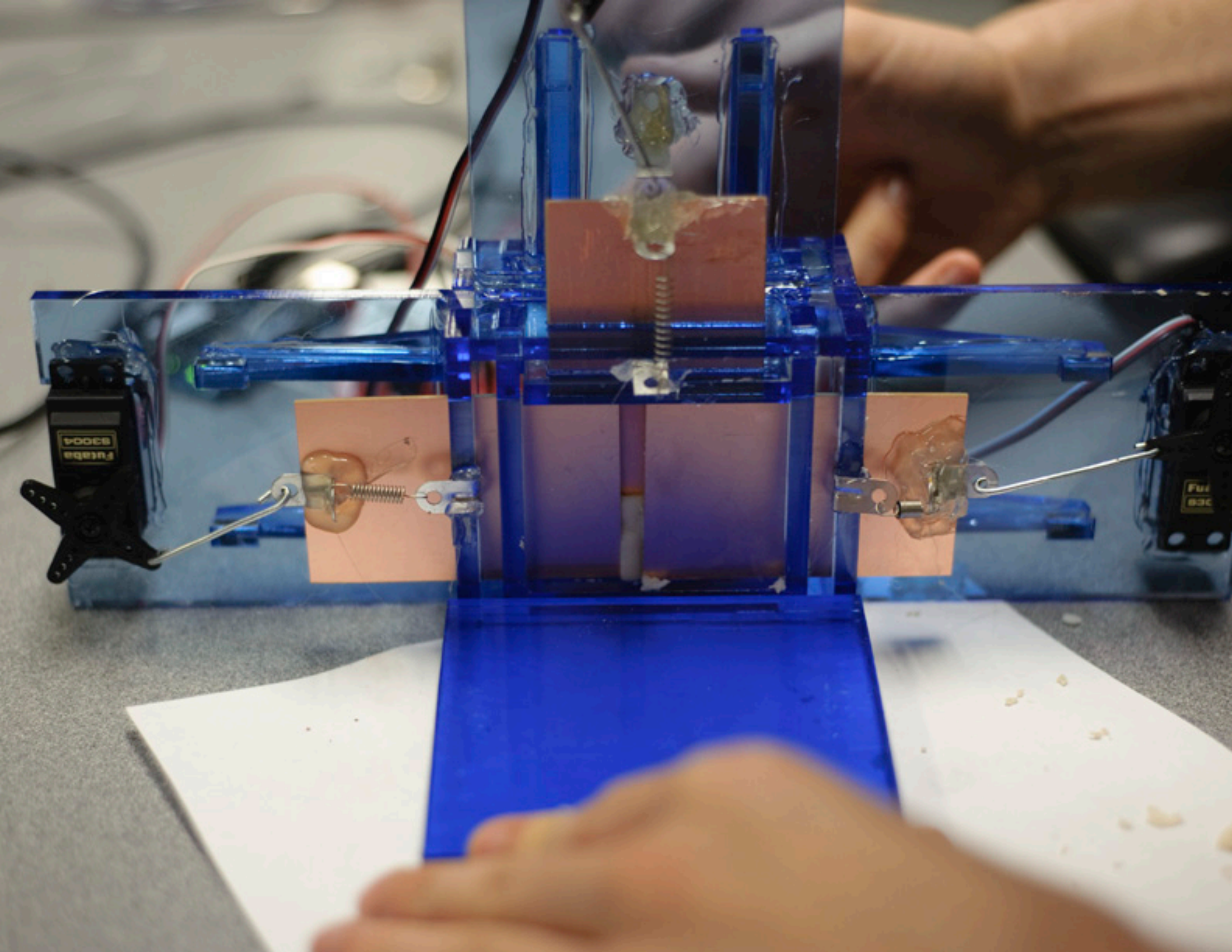


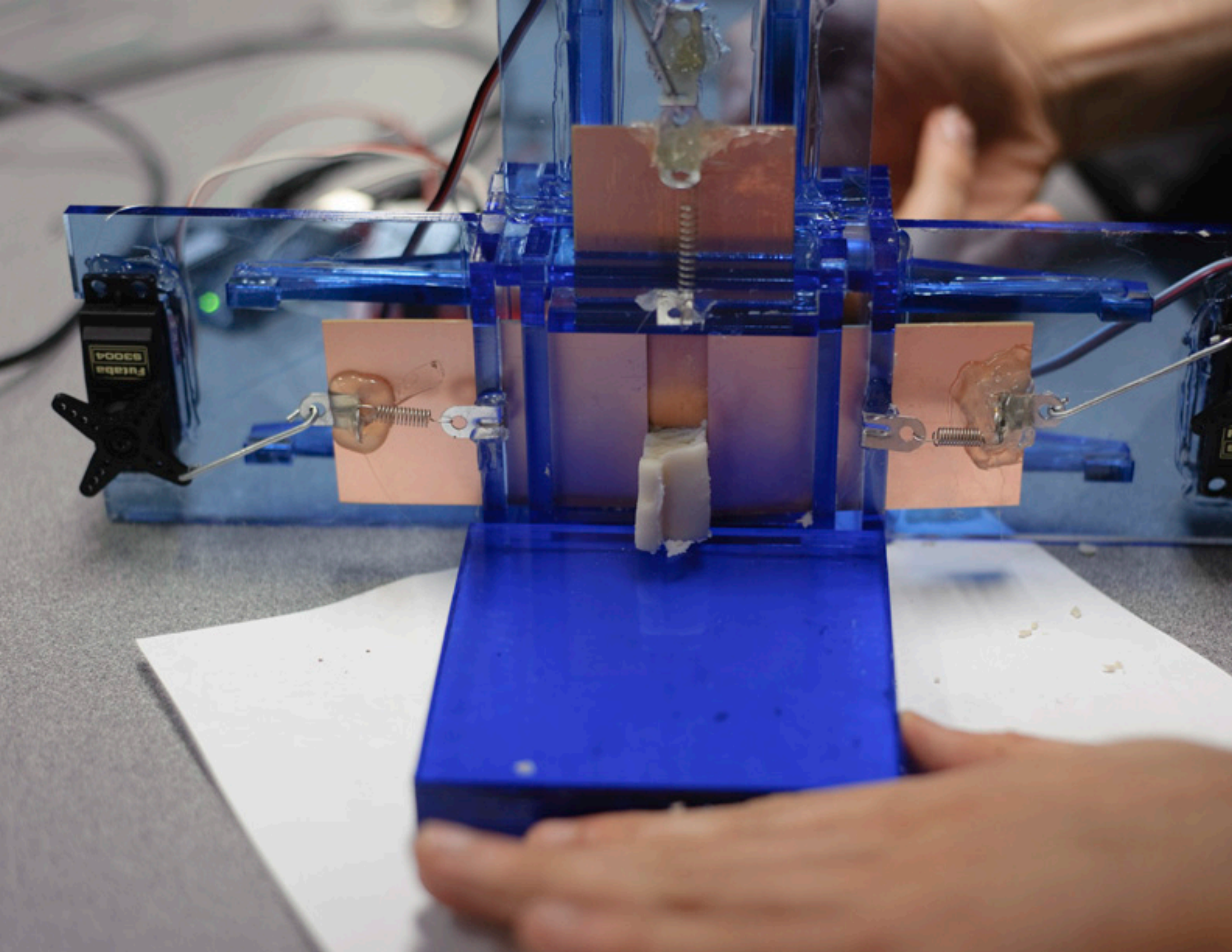
CAD file for laser cut prototype




Servo 01
Servo 03
Servo 02
Aperture blades
Arduino


Active extrusion of mashed potatos




Futaba
S3004