



**NYU**

**TANDON SCHOOL  
OF ENGINEERING**

# **AUTONOMOUS 3D PRINTER**

**ARDUINO MINI PROJECT  
ADVANCED MECHATRONICS**

**MITRA VARUN ANAND  
SAMANTH**

## INDEX

<b>Sl No</b>	<b>Content</b>	<b>Page</b>
1	<b>ABSTRACT</b>	3
2	<b>INTRODUCTION</b> 2.1. MightyBoard Rev E 2.2. Features	4 4 5
3	<b>DEVELOPMENT:</b> 3.1. Step 1 Bringing old Replicator back to life 3.2. Bootload 8U2 chip 3.3. ATMEGA 1280 3.4. Step 2: Inkscape Extension 3.5. Python to trace the contours: 3.6. G-Codes 3.7. Additional feature	7 7 8 10 27 34 38 40
4	<b>REFERENCES</b>	43

## **1. ABSTACT**

To to use a Makerbot Replicator 3D printer to print from pre-fed pictures instead of complex 3D solid models. The aim is to achieve this in two different ways:

1. An Inkscape extension to convert any Serial Vector Graphics image to an SCAD file, later converted to an STL format and printed.
2. A python program to accept any input image (Bitmap) and trace its boundary contours and find coordinates that can be interpreted to GCodes and printed in a 2 Dimensional format.

## **2.Introduction:**

### **2.1. MightyBoard Rev E:**

MightyBoard is a DIY Open Source 3D Printer Control Board designed by MakerBot company, it is mainly used in desktop 3D printer DIY creation. It adapts ATMEGA's Atmega1280 as the master control chip. MightyBoard provides stable and excellent performance while reducing the difficulty of development; it is very suitable for those who enjoy DIY. What's more, MightyBoard supports dual extruders, with the control software of MakerBot, it can realize double color printing to make printed items more attractive.



### **2.2. Features:**

1. Supports dual extruders.
2. Adopts professional thermocouple temperature measurement chip, to ensure accurate temperature measurement.
3. The heating module has protection function to make sure the safety of using.
4. Unique RGBLED display.
5. Compatible with the Arduino development environment, easy to develop further functions.
6. Use active cooling in the printing process to make sure the printed items can form better.

MightyBoard adopts the Atmega1280 IC and Atmega16u2 that is completely compatible with Arduino. Stepper motor driver module can choose A4988 or DRV8825 motor driver IC, so as to adapt to the needs of different frame structure. MightyBoard uses the thermocouple temperature measurement, with MAX6675 thermocouple temperature measurement IC, to ensure the accuracy of temperature measurement.

### 3. DEVELOPMENT:

#### 3.1. Step 1: Bringing old Replicator back to life:

##### 3.1.1. AVR ISP PROGRAMMER:

**In-system programming (ISP)**, also called **In-Circuit Serial Programming (ICSP)**, is the ability of some programmable logic devices, microcontrollers, and other embedded devices to be programmed while installed in a complete system, rather than requiring the chip to be programmed prior to installing it into the system.

There are several incompatible in-system programming protocols for programming microcontroller devices such as PIC microcontrollers, AVRs, and the Parallax Propeller. ICSP has been primarily implemented by Microchip Technology for programming PIC and dsPIC devices.



The primary advantage of this feature is that it allows manufacturers of electronic devices to integrate programming and testing into a single production phase, and save money, rather than requiring a separate programming stage prior to assembling the system. This may allow manufacturers to program the chips in their own system's

production line instead of buying preprogrammed chips from a manufacturer or distributor, making it feasible to apply code or design changes in the middle of a production run.

Microcontrollers are typically soldered directly to a printed circuit board and usually do not have the circuitry or space for a large external programming cable to another computer.

Typically, chips supporting ISP have internal circuitry to generate any necessary programming voltage from the system's normal supply voltage, and communicate with the programmer via a serial protocol. Most programmable logic devices use a variant of the JTAG protocol for ISP, in order to facilitate easier integration with automated testing procedures. Other devices usually use proprietary protocols or protocols defined by older standards. In systems complex enough to require moderately large glue logic, designers may implement a JTAG-controlled programming subsystem for non-JTAG devices such as flash memory and microcontrollers, allowing the entire programming and test procedure to be accomplished under the control of a single protocol.

An example of devices using ISP is the AVR line of micro-controllers by Atmel such as the ATmega series.

### **3.1.2. Features:**

- Programs both flash and EEPROM
- Supports fuses
- Compatible with Arduino IDE and AVRDUDE

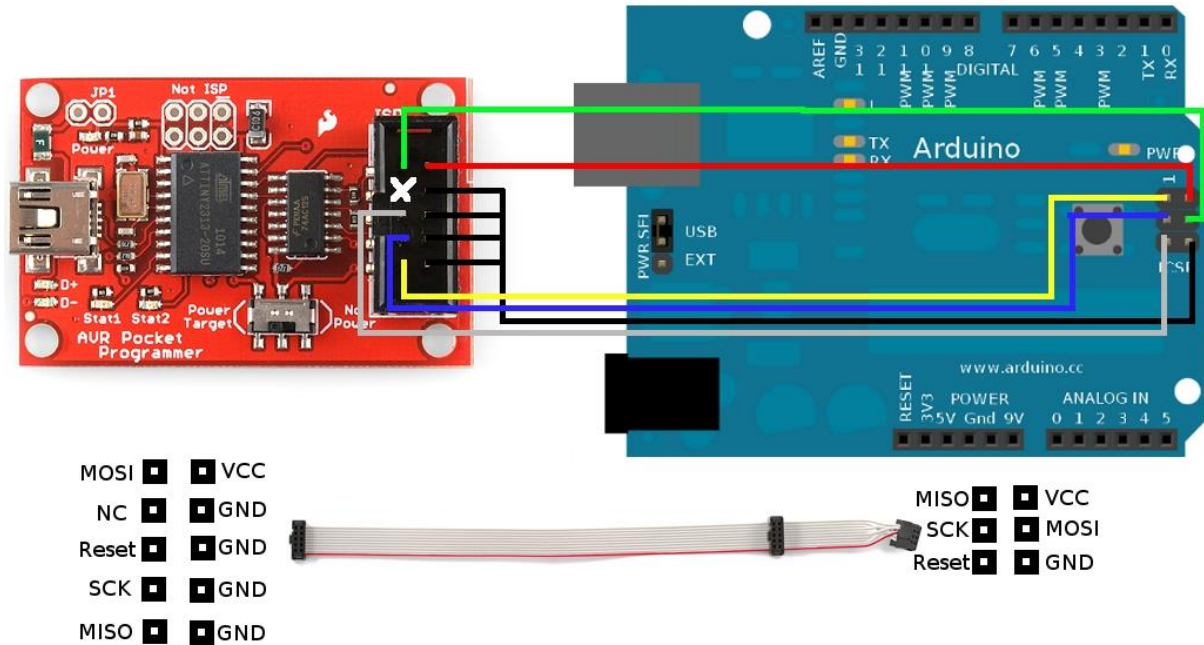
### **3.1.3. FUSES:**

- 3 Bytes of permanent storage
- Determines how the chip will act, whether it has a bootloader, what speed and voltage it likes to run at, etc.
- Are re-settable and don't have anything to do with protection from overpowering (like the fuses in a home).
- Values for a specific M/C is documented in the datasheets or use online fuse calculator

## **3.2. BOOTLOAD 8U2 CHIP:**

- Important first step- to make the chip communicate through USB again

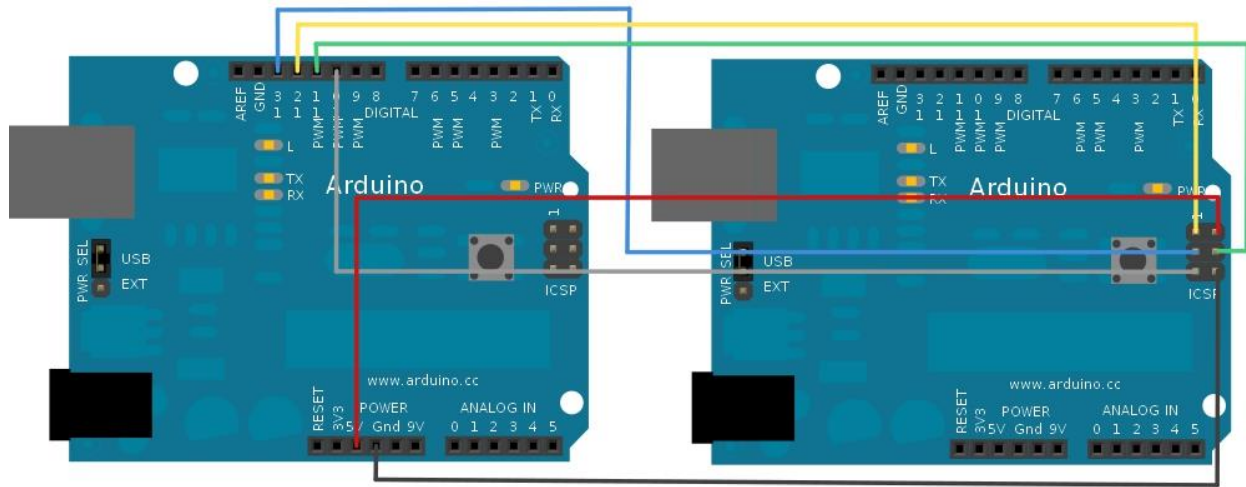
- Hex files can be found in manufacturer website



- After proper connection, command line to be entered in Command Prompt is:  
`avrdude -p at90usb82 -F -P usb -c avrispmkii -U flash:w:Makerbot-usbserial.hex -U lfuse:w:0xFF:m -U hfuse:w:0xD9:m -U efuse:w:0xF4:m -U lock:w:0x0F:m`



### 3.3. ATMEGA 1280:



- EASIER ISP PROGRAMMING USING ARDUINO IDE
- FUSES ARE TAKEN CARE BY IDE ITSELF

#### 3.3.1. CODE:

```
#include "Arduino.h"  
#undef SERIAL
```

```
#define PROG_FLICKER true
```

```
// Configure SPI clock (in Hz).
```

```
// E.g. for an attiny @128 kHz: the datasheet states that both the high
```

```
// and low spi clock pulse must be > 2 cpu cycles, so take 3 cycles i.e.
```

```
// divide target f_cpu by 6:
```

```
// #define SPI_CLOCK      (128000/6)
```

```
//
```

```
// A clock slow enough for an attiny85 @ 1MHz, is a reasonable default:
```

```
#define SPI_CLOCK      (1000000/6)
```

```
// Select hardware or software SPI, depending on SPI clock.
```

```
// Currently only for AVR, for other archs (Due, Zero,...),
// hardware SPI is probably too fast anyway.

#if defined(ARDUINO_ARCH_AVR)

  #if SPI_CLOCK > (F_CPU / 128)
  #define USE_HARDWARE_SPI
  #endif

  #endif

// Configure which pins to use:

// The standard pin configuration.
#ifndef ARDUINO_HOODLOADER2

#define RESET    10 // Use pin 10 to reset the target rather than SS
#define LED_HB   9
#define LED_ERR  8
#define LED_PMODE 7

// Uncomment following line to use the old Uno style wiring
// (using pin 11, 12 and 13 instead of the SPI header) on Leonardo, Due...

// #define USE_OLD_STYLE_WIRING

#ifdef USE_OLD_STYLE_WIRING

#define PIN_MOSI    11
#define PIN_MISO    12
#define PIN_SCK     13

#endif

// HOODLOADER2 means running sketches on the atmega16u2
// serial converter chips on Uno or Mega boards.
// We must use pins that are broken out:
#else

#define RESET      4
```

```

#define LED_HB      7
#define LED_ERR    6
#define LED_PMODE  5

#endif

// By default, use hardware SPI pins:
#ifndef PIN_MOSI
#define PIN_MOSI    MOSI
#endif

#ifndef PIN_MISO
#define PIN_MISO    MISO
#endif

#ifndef PIN_SCK
#define PIN_SCK     SCK
#endif

// Force bitbanged SPI if not using the hardware SPI pins:
#if (PIN_MISO != MISO) || (PIN_MOSI != MOSI) || (PIN_SCK != SCK)
#undef USE_HARDWARE_SPI
#endif

// Configure the serial port to use.
//
// Prefer the USB virtual serial port (aka. native USB port), if the Arduino has
one:
// - it does not autoreset (except for the magic baud rate of 1200).
// - it is more reliable because of USB handshaking.
//
// Leonardo and similar have an USB virtual serial port: 'Serial'.
// Due and Zero have an USB virtual serial port: 'SerialUSB'.
//
// On the Due and Zero, 'Serial' can be used too, provided you disable autoreset.
// To use 'Serial': #define SERIAL Serial

#ifdef SERIAL_PORT_USBVIRTUAL
#define SERIAL SERIAL_PORT_USBVIRTUAL

```

```

#else
#define SERIAL Serial
#endif

// Configure the baud rate:

#define BAUDRATE 19200
// #define BAUDRATE 115200
// #define BAUDRATE 1000000

#define HWVER 2
#define SWMAJ 1
#define SWMIN 18

// STK Definitions
#define STK_OK 0x10
#define STK_FAILED 0x11
#define STK_UNKNOWN 0x12
#define STK_INSYNC 0x14
#define STK_NOSYNC 0x15
#define CRC_EOP 0x20 //ok it is a space...

void pulse(int pin, int times);

#ifdef USE_HARDWARE_SPI
#include "SPI.h"
#else

#define SPI_MODE0 0x00

class SPISettings {
public:
// clock is in Hz
SPISettings(uint32_t clock, uint8_t bitOrder, uint8_t dataMode) :
clock(clock){
(void) bitOrder;
(void) dataMode;
};

```

```

private:
    uint32_t clock;

friend class BitBangedSPI;
};

class BitBangedSPI {
public:
    void begin() {
        digitalWrite(PIN_SCK, LOW);
        digitalWrite(PIN_MOSI, LOW);
        pinMode(PIN_SCK, OUTPUT);
        pinMode(PIN_MOSI, OUTPUT);
        pinMode(PIN_MISO, INPUT);
    }

    void beginTransaction(SPISettings settings) {
        pulseWidth = (500000 + settings.clock - 1) / settings.clock;
        if (pulseWidth == 0)
            pulseWidth = 1;
    }

    void end() {}

    uint8_t transfer (uint8_t b) {
        for (unsigned int i = 0; i < 8; ++i) {
            digitalWrite(PIN_MOSI, (b & 0x80) ? HIGH : LOW);
            digitalWrite(PIN_SCK, HIGH);
            delayMicroseconds(pulseWidth);
            b = (b << 1) | digitalRead(PIN_MISO);
            digitalWrite(PIN_SCK, LOW); // slow pulse
            delayMicroseconds(pulseWidth);
        }
        return b;
    }

private:
    unsigned long pulseWidth; // in microseconds
};

```

```

static BitBangedSPI SPI;

#endif

void setup() {
  SERIAL.begin(BAUDRATE);

  pinMode(LED_PMODE, OUTPUT);
  pulse(LED_PMODE, 2);
  pinMode(LED_ERR, OUTPUT);
  pulse(LED_ERR, 2);
  pinMode(LED_HB, OUTPUT);
  pulse(LED_HB, 2);

}

int error = 0;
int pmode = 0;
// address for reading and writing, set by 'U' command
unsigned int here;
uint8_t buff[256]; // global block storage

#define beget16(addr) (*addr * 256 + *(addr+1) )
typedef struct param {
  uint8_t devicecode;
  uint8_t revision;
  uint8_t progtype;
  uint8_t parmode;
  uint8_t polling;
  uint8_t selftimed;
  uint8_t lockbytes;
  uint8_t fusebytes;
  uint8_t flashpoll;
  uint16_t eeprompoll;
  uint16_t pagesize;
  uint16_t eepromsize;
  uint32_t flashsize;
}
parameter;

```

**parameter param;**

**// this provides a heartbeat on pin 9, so you can tell the software is running.**

**uint8\_t hbval = 128;**

**int8\_t hbdelta = 8;**

**void heartbeat() {**

**static unsigned long last\_time = 0;**

**unsigned long now = millis();**

**if ((now - last\_time) < 40)**

**return;**

**last\_time = now;**

**if (hbval > 192) hbdelta = -hbdelta;**

**if (hbval < 32) hbdelta = -hbdelta;**

**hbval += hbdelta;**

**analogWrite(LED\_HB, hbval);**

**}**

**static bool rst\_active\_high;**

**void reset\_target(bool reset) {**

**digitalWrite(RESET, ((reset && rst\_active\_high) || (!reset && !rst\_active\_high)) ? HIGH : LOW);**

**}**

**void loop(void) {**

**// is pmode active?**

**if (pmode) {**

**digitalWrite(LED\_PMODE, HIGH);**

**} else {**

**digitalWrite(LED\_PMODE, LOW);**

**}**

**// is there an error?**

**if (error) {**

**digitalWrite(LED\_ERR, HIGH);**

**} else {**

**digitalWrite(LED\_ERR, LOW);**

**}**

**// light the heartbeat LED**

```

heartbeat();
if (SERIAL.available()) {
    avrISP();
}
}

uint8_t getch() {
    while (!SERIAL.available());
    return SERIAL.read();
}
void fill(int n) {
    for (int x = 0; x < n; x++) {
        buff[x] = getch();
    }
}

#define PTIME 30
void pulse(int pin, int times) {
    do {
        digitalWrite(pin, HIGH);
        delay(PTIME);
        digitalWrite(pin, LOW);
        delay(PTIME);
    } while (times--);
}

void prog_lamp(int state) {
    if (PROG_FLICKER) {
        digitalWrite(LED_PMODE, state);
    }
}

uint8_t spi_transaction(uint8_t a, uint8_t b, uint8_t c, uint8_t d) {
    SPI.transfer(a);
    SPI.transfer(b);
    SPI.transfer(c);
    return SPI.transfer(d);
}

void empty_reply() {

```



```
if (CRC_EOP == getch()) {  
    SERIAL.print((char)STK_INSYNC);  
    SERIAL.print((char)STK_OK);  
} else {  
    error++;  
    SERIAL.print((char)STK_NOSYNC);  
}  
}
```

```
void breply(uint8_t b) {  
    if (CRC_EOP == getch()) {  
        SERIAL.print((char)STK_INSYNC);  
        SERIAL.print((char)b);  
        SERIAL.print((char)STK_OK);  
    } else {  
        error++;  
        SERIAL.print((char)STK_NOSYNC);  
    }  
}
```

```
void get_version(uint8_t c) {  
    switch (c) {  
        case 0x80:  
            breply(HWVER);  
            break;  
        case 0x81:  
            breply(SWMAJ);  
            break;  
        case 0x82:  
            breply(SWMIN);  
            break;  
        case 0x93:  
            breply('S'); // serial programmer  
            break;  
        default:  
            breply(0);  
    }  
}
```

```
void set_parameters() {
```

```

// call this after reading paramter packet into buff[]
param.devicecode = buff[0];
param.revision   = buff[1];
param.progtype   = buff[2];
param.parmode    = buff[3];
param.polling    = buff[4];
param.selftimed  = buff[5];
param.lockbytes  = buff[6];
param.fusebytes  = buff[7];
param.flashpoll  = buff[8];
// ignore buff[9] (= buff[8])
// following are 16 bits (big endian)
param.eeprompoll = beget16(&buff[10]);
param.pagesize  = beget16(&buff[12]);
param.eepromsize = beget16(&buff[14]);

// 32 bits flashsize (big endian)
param.flashsize = buff[16] * 0x01000000
                + buff[17] * 0x00010000
                + buff[18] * 0x00000100
                + buff[19];

// avr devices have active low reset, at89sx are active high
rst_active_high = (param.devicecode >= 0xe0);
}

void start_pmode() {

// Reset target before driving PIN_SCK or PIN_MOSI

// SPI.begin() will configure SS as output,
// so SPI master mode is selected.
// We have defined RESET as pin 10,
// which for many arduino's is not the SS pin.
// So we have to configure RESET as output here,
// (reset_target() first sets the correct level)
reset_target(true);
pinMode(RESET, OUTPUT);
SPI.begin();

```

```
SPI.beginTransaction(SPISettings(SPI_CLOCK, MSBFIRST,  
SPI_MODE0));
```

```
// See avr datasheets, chapter "SERIAL_PRG Programming Algorithm":
```

```
// Pulse RESET after PIN_SCK is low:  
digitalWrite(PIN_SCK, LOW);  
delay(20); // discharge PIN_SCK, value arbitrarily chosen  
reset_target(false);  
// Pulse must be minimum 2 target CPU clock cycles  
// so 100 usec is ok for CPU speeds above 20KHz  
delayMicroseconds(100);  
reset_target(true);
```

```
// Send the enable programming command:  
delay(50); // datasheet: must be > 20 msec  
spi_transaction(0xAC, 0x53, 0x00, 0x00);  
pmode = 1;  
}
```

```
void end_pmode() {  
SPI.end();  
// We're about to take the target out of reset  
// so configure SPI pins as input  
pinMode(PIN_MOSI, INPUT);  
pinMode(PIN_SCK, INPUT);  
reset_target(false);  
pinMode(RESET, INPUT);  
pmode = 0;  
}
```

```
void universal() {  
uint8_t ch;  
  
fill(4);  
ch = spi_transaction(buff[0], buff[1], buff[2], buff[3]);  
breply(ch);  
}
```

```
void flash(uint8_t hilo, unsigned int addr, uint8_t data) {
```

```

spi_transaction(0x40 + 8 * hilo,
                addr >> 8 & 0xFF,
                addr & 0xFF,
                data);
}
void commit(unsigned int addr) {
if (PROG_FLICKER) {
    prog_lamp(LOW);
}
spi_transaction(0x4C, (addr >> 8) & 0xFF, addr & 0xFF, 0);
if (PROG_FLICKER) {
    delay(PTIME);
    prog_lamp(HIGH);
}
}

```

```

unsigned int current_page() {
if (param.pagesize == 32) {
    return here & 0xFFFFFFFF0;
}
if (param.pagesize == 64) {
    return here & 0xFFFFF0;
}
if (param.pagesize == 128) {
    return here & 0xFFFFFC0;
}
if (param.pagesize == 256) {
    return here & 0xFFFF80;
}
return here;
}

```

```

void write_flash(int length) {
fill(length);
if (CRC_EOP == getch()) {
    SERIAL.print((char) STK_INSYNC);
    SERIAL.print((char) write_flash_pages(length));
} else {
    error++;
}
}

```

```

    SERIAL.print((char) STK_NOSYNC);
}
}

uint8_t write_flash_pages(int length) {
    int x = 0;
    unsigned int page = current_page();
    while (x < length) {
        if (page != current_page()) {
            commit(page);
            page = current_page();
        }
        flash(LOW, here, buff[x++]);
        flash(HIGH, here, buff[x++]);
        here++;
    }

    commit(page);

    return STK_OK;
}

#define EECHUNK (32)
uint8_t write_eeprom(unsigned int length) {
    // here is a word address, get the byte address
    unsigned int start = here * 2;
    unsigned int remaining = length;
    if (length > param.eepromsize) {
        error++;
        return STK_FAILED;
    }
    while (remaining > EECHUNK) {
        write_eeprom_chunk(start, EECHUNK);
        start += EECHUNK;
        remaining -= EECHUNK;
    }
    write_eeprom_chunk(start, remaining);
    return STK_OK;
}
// write (length) bytes, (start) is a byte address

```

```

uint8_t write_eeprom_chunk(unsigned int start, unsigned int length) {
  // this writes byte-by-byte,
  // page writing may be faster (4 bytes at a time)
  fill(length);
  prog_lamp(LOW);
  for (unsigned int x = 0; x < length; x++) {
    unsigned int addr = start + x;
    spi_transaction(0xC0, (addr >> 8) & 0xFF, addr & 0xFF, buff[x]);
    delay(45);
  }
  prog_lamp(HIGH);
  return STK_OK;
}

```

```

void program_page() {
  char result = (char) STK_FAILED;
  unsigned int length = 256 * getch();
  length += getch();
  char memtype = getch();
  // flash memory @here, (length) bytes
  if (memtype == 'F') {
    write_flash(length);
    return;
  }
  if (memtype == 'E') {
    result = (char)write_eeprom(length);
    if (CRC_EOP == getch()) {
      SERIAL.print((char) STK_INSYNC);
      SERIAL.print(result);
    } else {
      error++;
      SERIAL.print((char) STK_NOSYNC);
    }
  }
  return;
}
SERIAL.print((char)STK_FAILED);
return;
}

```

```

uint8_t flash_read(uint8_t hilo, unsigned int addr) {

```

```
return spi_transaction(0x20 + hilo * 8,  
                      (addr >> 8) & 0xFF,  
                      addr & 0xFF,  
                      0);  
}
```

```
char flash_read_page(int length) {  
  for (int x = 0; x < length; x += 2) {  
    uint8_t low = flash_read(LOW, here);  
    SERIAL.print((char) low);  
    uint8_t high = flash_read(HIGH, here);  
    SERIAL.print((char) high);  
    here++;  
  }  
  return STK_OK;  
}
```

```
char eeprom_read_page(int length) {  
  // here again we have a word address  
  int start = here * 2;  
  for (int x = 0; x < length; x++) {  
    int addr = start + x;  
    uint8_t ee = spi_transaction(0xA0, (addr >> 8) & 0xFF, addr & 0xFF, 0xFF);  
    SERIAL.print((char) ee);  
  }  
  return STK_OK;  
}
```

```
void read_page() {  
  char result = (char)STK_FAILED;  
  int length = 256 * getch();  
  length += getch();  
  char memtype = getch();  
  if (CRC_EOP != getch()) {  
    error++;  
    SERIAL.print((char) STK_NOSYNC);  
    return;  
  }  
  SERIAL.print((char) STK_INSYNC);  
  if (memtype == 'F') result = flash_read_page(length);  
}
```

```
if (memtype == 'E') result = eeprom_read_page(length);  
SERIAL.print(result);  
}
```

```
void read_signature() {  
  if (CRC_EOP != getch()) {  
    error++;  
    SERIAL.print((char) STK_NOSYNC);  
    return;  
  }  
  SERIAL.print((char) STK_INSYNC);  
  uint8_t high = spi_transaction(0x30, 0x00, 0x00, 0x00);  
  SERIAL.print((char) high);  
  uint8_t middle = spi_transaction(0x30, 0x00, 0x01, 0x00);  
  SERIAL.print((char) middle);  
  uint8_t low = spi_transaction(0x30, 0x00, 0x02, 0x00);  
  SERIAL.print((char) low);  
  SERIAL.print((char) STK_OK);  
}
```

```
////////////////////////////////////  
////////////////////////////////////
```

```
////////////////////////////////////  
////////////////////////////////////
```

```
void avrisp() {  
  uint8_t ch = getch();  
  switch (ch) {  
    case '0': // signon  
      error = 0;  
      empty_reply();  
      break;  
    case '1':  
      if (getch() == CRC_EOP) {  
        SERIAL.print((char) STK_INSYNC);  
        SERIAL.print("AVR ISP");  
        SERIAL.print((char) STK_OK);  
      }  
      else {  
        error++;  
      }  
    }  
}
```



```
    SERIAL.print((char) STK_NOSYNC);
}
break;
case 'A':
    get_version(getch());
    break;
case 'B':
    fill(20);
    set_parameters();
    empty_reply();
    break;
case 'E': // extended parameters - ignore for now
    fill(5);
    empty_reply();
    break;
case 'P':
    if (!pmode)
        start_pmode();
    empty_reply();
    break;
case 'U': // set address (word)
    here = getch();
    here += 256 * getch();
    empty_reply();
    break;

case 0x60: //STK_PROG_FLASH
    getch(); // low addr
    getch(); // high addr
    empty_reply();
    break;
case 0x61: //STK_PROG_DATA
    getch(); // data
    empty_reply();
    break;

case 0x64: //STK_PROG_PAGE
    program_page();
    break;
```

```
case 0x74: //STK_READ_PAGE 't'
    read_page();
    break;

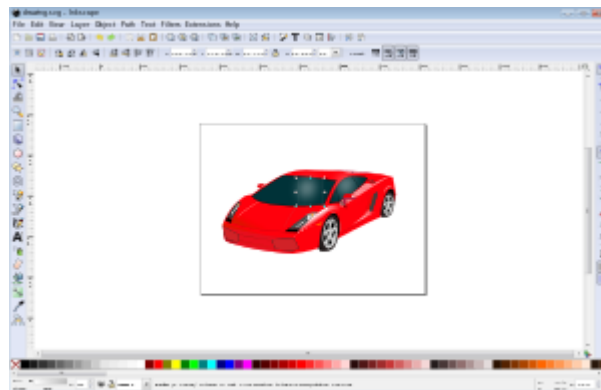
case 'V': //0x56
    universal();
    break;
case 'Q': //0x51
    error = 0;
    end_pmode();
    empty_reply();
    break;

case 0x75: //STK_READ_SIGN 'u'
    read_signature();
    break;
// expecting a command, not CRC_EOP
// this is how we can get back in sync
case CRC_EOP:
    error++;
    SERIAL.print((char) STK_NOSYNC);
    break;

// anything else we will return STK_UNKNOWN
default:
    error++;
    if (CRC_EOP == getch())
        SERIAL.print((char)STK_UNKNOWN);
    else
        SERIAL.print((char)STK_NOSYNC);
}
}
```

### 3.4. Step 2: Inkscape Extension:

**Inkscape** is a free and open-source vector graphics editor; it can be used to create or edit vector graphics such as illustrations, diagrams, line arts, charts, logos and complex paintings. Inkscape's primary vector graphics format is Scalable Vector Graphics (SVG) version 1.1. While Inkscape can import and export several formats, all editing workflow inevitably occur within the constraints of the SVG format.



Inkscape can render the primitive vector shapes (e.g. rectangles, ellipses, polygons, arcs, spirals, stars and isometric boxes), text and regions containing raster graphics. It supports image tracing, enabling the editor to create vector graphics from photos and other raster sources. Created shapes can be subjected to further transformations, such as moving, rotating, scaling and skewing. These objects may be filled with solid colors or color gradients, their borders stroked or their transparency changed. As of 2015, Inkscape does not support SVG animation or full Cascading Style Sheet (CSS) specifications.

#### 3.4.1. Features:

- Inkscape is an open-source vector graphics editor, supports custom extensions from extension template
- Built extension to convert a 2D Drawing(SVG) to an SCAD file, then an STL file using Bitmap tracing (python)
- This method is currently limited to closed polygons and less complex shapes

### 3.4.2. CODE FOR PYTHON INKSCAPE EXTENSION:

```
import os.path
import inkex
import simplepath
import simpletransform
import cubicsuperpath
import cspsubdiv
import bezmisc
import re
import string

DEFAULT_WIDTH = 100
DEFAULT_HEIGHT = 100
RE_AUTO_HEIGHT_ID = re.compile(r'.*?_(\d+(?:_\d+)?)_mm$')
RE_AUTO_HEIGHT_DESC = re.compile(
    r'^(?:ht|height):s*(\d+(?:_\d+)?) mm$',
    re.MULTILINE)
DESC_TAGS = ['desc', inkex.addNS('desc', 'svg')]

def parseLengthWithUnits(str):
    """
    Parse an SVG value which may or may not have units attached
    This version is greatly simplified in that it only allows: no units,
    units of px, and units of %. Everything else, it returns None for.
    There is a more general routine to consider in scour.py if more
    generality is ever needed.
    """

    u = 'px'
    s = str.strip()
    if s[-2:] == 'px':
        s = s[:-2]
    elif s[-1:] == '%':
        u = '%'
        s = s[:-1]

    try:
```

```
    v = float(s)
except:
    return None, None
```

```
return v, u
```

```
def pointInBBox(pt, bbox):
```

```
    '''
```

```
    Determine if the point pt=[x, y] lies on or within the bounding
    box bbox=[xmin, xmax, ymin, ymax].
```

```
    '''
```

```
    # if ( x < xmin ) or ( x > xmax ) or ( y < ymin ) or ( y > ymax )
```

```
    if (pt[0] < bbox[0]) or (pt[0] > bbox[1]) or \
        (pt[1] < bbox[2]) or (pt[1] > bbox[3]):
```

```
        return False
```

```
    else:
```

```
        return True
```

```
def bboxInBBox(bbox1, bbox2):
```

```
    '''
```

```
    Determine if the bounding box bbox1 lies on or within the
    bounding box bbox2. NOTE: we do not test for strict enclosure.
```

```
    Structure of the bounding boxes is
```

```
    bbox1 = [ xmin1, xmax1, ymin1, ymax1 ]
```

```
    bbox2 = [ xmin2, xmax2, ymin2, ymax2 ]
```

```
    '''
```

```
    # if ( xmin1 < xmin2 ) or ( xmax1 > xmax2 ) or
```

```
    # ( ymin1 < ymin2 ) or ( ymax1 > ymax2 )
```

```
    if (bbox1[0] < bbox2[0]) or (bbox1[1] > bbox2[1]) or \
        (bbox1[2] < bbox2[2]) or (bbox1[3] > bbox2[3]):
```

```
        return False
```

```
    else:
```

```
        return True
```

```
def pointInPoly(p, poly, bbox=None):
```

```
'''
```

```
Use a ray casting algorithm to see if the point p = [x, y] lies within  
the polygon poly = [[x1,y1],[x2,y2],...]. Returns True if the point  
is within poly, lies on an edge of poly, or is a vertex of poly.
```

```
'''
```

```
if (p is None) or (poly is None):  
    return False
```

```
# Check to see if the point lies outside the polygon's bounding box  
if bbox is not None:  
    if not pointInBBox(p, bbox):  
        return False
```

```
# Check to see if the point is a vertex  
if p in poly:  
    return True
```

```
# Handle a boundary case associated with the point  
# lying on a horizontal edge of the polygon  
x = p[0]  
y = p[1]  
p1 = poly[0]  
p2 = poly[1]  
for i in range(len(poly)):  
    if i != 0:  
        p1 = poly[i - 1]  
        p2 = poly[i]  
        if (y == p1[1]) and (p1[1] == p2[1]) and \  
            (x > min(p1[0], p2[0])) and (x < max(p1[0], p2[0])):  
            return True
```

```
n = len(poly)  
inside = False
```

```
p1_x, p1_y = poly[0]  
for i in range(n + 1):  
    p2_x, p2_y = poly[i % n]  
    if y > min(p1_y, p2_y):  
        if y <= max(p1_y, p2_y):
```

```

    if x <= max(p1_x, p2_x):
        if p1_y != p2_y:
            intersect = p1_x + (y - p1_y) * (p2_x - p1_x) /\
                (p2_y - p1_y)
            if x <= intersect:
                inside = not inside
            else:
                inside = not inside
        p1_x, p1_y = p2_x, p2_y

```

```

return inside

```

```

def polyInPoly(poly1, bbox1, poly2, bbox2):

```

```

'''

```

```

    Determine if polygon poly2 = [[x1,y1],[x2,y2],...]
    contains polygon poly1.

```

```

    The bounding box information, bbox=[xmin, xmax, ymin, ymax]
    is optional. When supplied it can be used to perform rejections.
    Note that one bounding box containing another is not sufficient
    to imply that one polygon contains another. It's necessary, but
    not sufficient.

```

```

'''

```

```

# See if poly1's bounding box is NOT contained by poly2's bounding box
# if it isn't, then poly1 cannot be contained by poly2.

```

```

if (bbox1 is not None) and (bbox2 is not None):

```

```

    if not bboxInBBBox(bbox1, bbox2):

```

```

        return False

```

```

# To see if poly1 is contained by poly2, we need to ensure that each
# vertex of poly1 lies on or within poly2

```

```

for p in poly1:

```

```

    if not pointInPoly(p, poly2, bbox2):

```

```

        return False

```

```

# Looks like poly1 is contained on or in Poly2

```

```
return True
```

```
def subdivideCubicPath(sp, flat, i=1):
```

```
    '''
```

```
    [ Lifted from eggbot.py with impunity ]
```

```
    Break up a bezier curve into smaller curves, each of which  
    is approximately a straight line within a given tolerance  
    (the "smoothness" defined by [flat]).
```

```
    This is a modified version of cspsubdiv.cspsubdiv(): rewritten  
    because recursion-depth errors on complicated line segments  
    could occur with cspsubdiv.cspsubdiv().
```

```
    '''
```

```
while True:
```

```
    while True:
```

```
        if i >= len(sp):
```

```
            return
```

```
        p0 = sp[i - 1][1]
```

```
        p1 = sp[i - 1][2]
```

```
        p2 = sp[i][0]
```

```
        p3 = sp[i][1]
```

```
        b = (p0, p1, p2, p3)
```

```
        if cspsubdiv.maxdist(b) > flat:
```

```
            break
```

```
        i += 1
```

```
    one, two = bezmisc.beziersplitatt(b, 0.5)
```

```
    sp[i - 1][2] = one[1]
```

```
    sp[i][0] = two[2]
```

```
    p = [one[2], one[3], two[1]]
```

```
    sp[i:1] = [p]
```

**3.4.3. Method:**



The program converts any SVG Image to an SCAD 3D solid format, the extruding height of which can be selected by the user. This CAD file and can be fed to a Makerbot software and printed directly.

### 3.5. Python to trace the contours:

The code for finding the boundaries of any png black and white image is as follows:

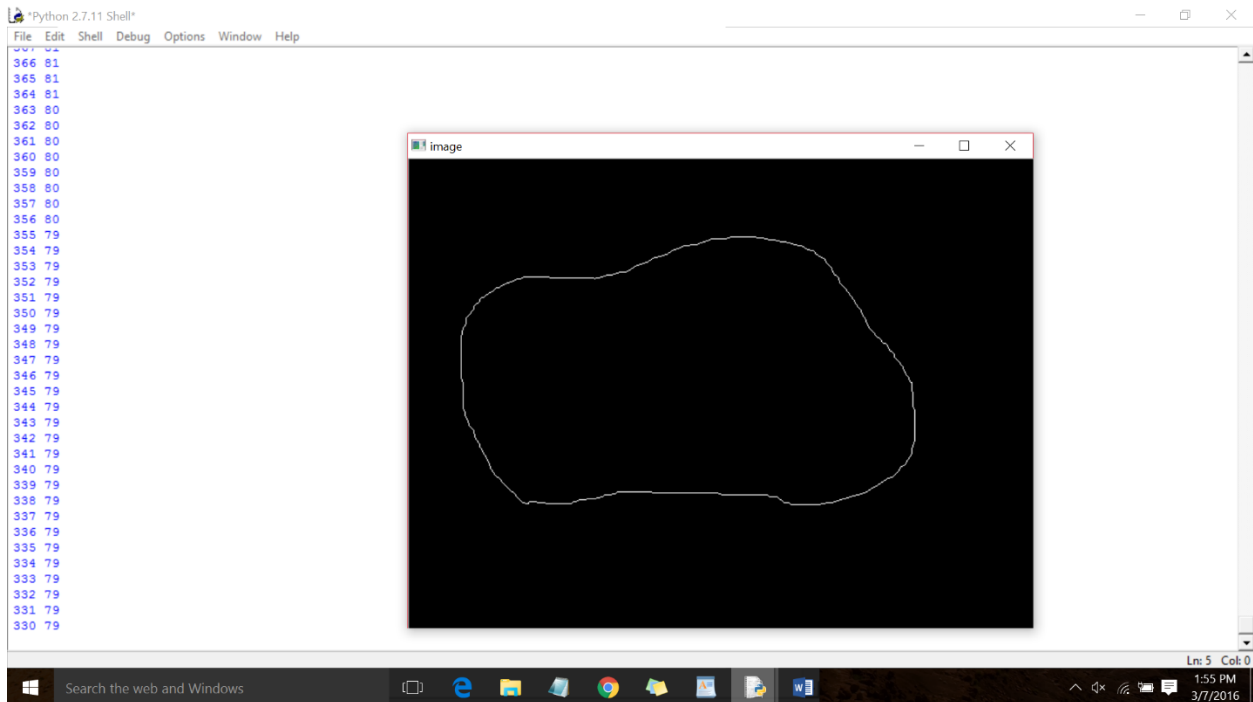
---

```
import numpy as np
import cv2
import time

##im = cv2.imread('test.jpg')
im = cv2.imread('C:/Users/Samanth/Desktop/d2.bmp',0)
img = cv2.imread('C:/Users/Samanth/Desktop/d1.jpg',3)
##imggray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
##ret,thresh = cv2.threshold(imggray,127,255,0)

image, contours, hierarchy = cv2.findContours(im,cv2.RETR_EXTERNAL ,cv2.CHAIN_APPROX_NONE)
cnt = contours[0]
print len(cnt)
for i in range (0,len(cnt)):
    ##print cnt[i]
    x= cnt[i][0][0]
    y= cnt[i][0][1]
    print x,y
    img[y,x] = [255,255,255]
    cv2.imshow('image',img)
    cv2.waitKey(1)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imwrite('messigray.jpg',im)
|
```

### 3.5.1. OUTPUT:



### 3.5.2. Method:

The png image is fed to the python program that traces the contours and finds the coordinates as shown above. These coordinates can be either edited with a simple Text Editor file, or can be fed to a GCODE interpreter that can be directly fed to the printer and printed as a 2 dimensional drawing, just like how the image is.

### 3.6. GCode:

**G-code** (also *RS-274*), which has many variants, is the common name for the most widely used numerical control (NC) programming language. It is used mainly in computer-aided manufacturing to control automated machine tools. G-code is sometimes called **G programming language**, not to be confused with LabVIEW's G programming language.

G-code is a language in which people tell computerized machine tools how to make something. The "how" is defined by instructions on where to move, how fast to move, and what path to move. The most common situation is that, within a machine tool, a cutting tool is moved according to these instructions through a toolpath and cuts away material to leave only the finished workpiece. The same concept also extends to noncutting tools such as forming or burnishing tools, photoplotting, additive methods such as 3D printing, and measuring instruments.

#### 3.6.1. Most frequently appearing G-Code calls:

- G1 – Coordinated Motion
- G21 – Millimeters as units
- G90 – Absolute Positioning
- G91 – Incremental Positioning
- G92 – Set current as home

#### 3.6.2. Useful M-Codes :

M0 Automatic halt

M1 Optional halt M2 End of program

M6 Typically used as “wait for Extruder to heat up” though this is something of a hack.

M101 Extruder on, forward

M102 Extruder on, reverse

M103 Extruder off

M105 get extruder temperature

### 3.6.3. Sample:

```
}; X-49.47 Y-2.59 Z0.14 F2160.0 E1.07
}; X-49.44 Y-2.65 Z0.14 F2160.0 E1.074
}; X-49.38 Y-2.69 Z0.14 F2160.0 E1.077
}; X-49.26 Y-2.7 Z0.14 F2160.0 E1.083
}; X-48.36 Y-2.48 Z0.14 F2160.0 E1.128
}; X-48.09 Y-2.5 Z0.14 F2160.0 E1.142
}; X-47.81 Y-2.67 Z0.14 F2160.0 E1.157
}; X-47.73 Y-2.96 Z0.14 F2160.0 E1.172
}; X-47.82 Y-3.36 Z0.14 F2160.0 E1.192
}; X-47.74 Y-3.43 Z0.14 F2160.0 E1.197
}; X-47.35 Y-3.34 Z0.14 F2160.0 E1.217
}; X-47.04 Y-3.44 Z0.14 F2160.0 E1.233
}; X-46.89 Y-3.7 Z0.14 F2160.0 E1.247
}; X-46.87 Y-3.97 Z0.14 F2160.0 E1.261
}; X-47.08 Y-4.87 Z0.14 F2160.0 E1.306
}; X-47.08 Y-4.99 Z0.14 F2160.0 E1.311
}; X-47.05 Y-5.04 Z0.14 F2160.0 E1.315
}; X-46.99 Y-5.06 Z0.14 F2160.0 E1.317
}; X-46.46 Y-4.94 Z0.14 F2160.0 E1.344
}; X-46.11 Y-5.37 Z0.14 F2160.0 E1.371
}; X-46.22 Y-5.83 Z0.14 F2160.0 E1.394
}; X-46.2 Y-5.88 Z0.14 F2160.0 E1.397
}; X-46.15 Y-5.91 Z0.14 F2160.0 E1.4
}; X-46.03 Y-5.92 Z0.14 F2160.0 E1.406
}; X-45.14 Y-5.71 Z0.14 F2160.0 E1.451
}; X-44.86 Y-5.73 Z0.14 F2160.0 E1.464
}; X-44.59 Y-5.9 Z0.14 F2160.0 E1.48
}; X-44.5 Y-6.19 Z0.14 F2160.0 E1.495
}; X-44.59 Y-6.58 Z0.14 F2160.0 E1.514
}; X-44.52 Y-6.66 Z0.14 F2160.0 E1.519
}; X-44.12 Y-6.56 Z0.14 F2160.0 E1.539
}; X-43.79 Y-6.68 Z0.14 F2160.0 E1.556
}; X-43.69 Y-7.0 Z0.14 F2160.0 E1.573
}; X-43.78 Y-7.44 Z0.14 F2160.0 E1.595
}; X-43.76 Y-7.55 Z0.14 F2160.0 E1.6
}; X-43.64 Y-7.71 Z0.14 F2160.0 E1.61
}; X-43.18 Y-8.07 Z0.14 F2160.0 E1.639
}; X-42.96 Y-8.36 Z0.14 F2160.0 E1.656
;
```

---

G1 X-43.78 Y-7.44 Z0.14 F2160.0 E1.595  
G1 X-43.76 Y-7.55 Z0.14 F2160.0 E1.6  
G1 X-43.64 Y-7.71 Z0.14 F2160.0 E1.61  
G1 X-43.18 Y-8.07 Z0.14 F2160.0 E1.639  
G1 X-42.96 Y-8.36 Z0.14 F2160.0 E1.656  
G1 X-42.9 Y-8.85 Z0.14 F2160.0 E1.68  
G1 X-43.07 Y-9.71 Z0.14 F2160.0 E1.723  
G1 X-43.05 Y-9.82 Z0.14 F2160.0 E1.729  
G1 X-43.02 Y-9.88 Z0.14 F2160.0 E1.732  
G1 X-42.96 Y-9.89 Z0.14 F2160.0 E1.735  
G1 X-42.43 Y-9.78 Z0.14 F2160.0 E1.762  
G1 X-42.08 Y-10.21 Z0.14 F2160.0 E1.789  
G1 X-42.17 Y-10.61 Z0.14 F2160.0 E1.809  
G1 X-42.1 Y-10.69 Z0.14 F2160.0 E1.814  
G1 X-41.7 Y-10.59 Z0.14 F2160.0 E1.834  
G1 X-41.4 Y-10.7 Z0.14 F2160.0 E1.85  
G1 X-41.24 Y-10.94 Z0.14 F2160.0 E1.864  
G1 X-41.2 Y-11.2 Z0.14 F2160.0 E1.877  
G1 X-41.38 Y-12.07 Z0.14 F2160.0 E1.921  
G1 X-41.34 Y-12.38 Z0.14 F2160.0 E1.936  
G1 X-41.22 Y-12.54 Z0.14 F2160.0 E1.946  
G1 X-40.73 Y-12.9 Z0.14 F2160.0 E1.975  
G1 X-40.54 Y-13.2 Z0.14 F2160.0 E1.992  
G1 X-40.46 Y-13.65 Z0.14 F2160.0 E2.015  
G1 X-40.52 Y-14.8 Z0.14 F2160.0 E2.071  
G1 X-40.15 Y-15.12 Z0.14 F2160.0 E2.095  
G1 X-39.73 Y-15.61 Z0.14 F2160.0 E2.126  
G1 X-39.63 Y-16.25 Z0.14 F2160.0 E2.158  
G1 X-39.81 Y-17.55 Z0.14 F2160.0 E2.222  
G1 X-39.73 Y-18.03 Z0.14 F2160.0 E2.246  
G1 X-39.61 Y-18.19 Z0.14 F2160.0 E2.256  
G1 X-39.15 Y-18.55 Z0.14 F2160.0 E2.285  
G1 X-38.92 Y-18.84 Z0.14 F2160.0 E2.302  
G1 X-38.87 Y-19.33 Z0.14 F2160.0 E2.326  
G1 X-39.04 Y-20.19 Z0.14 F2160.0 E2.369  
G1 X-39.02 Y-20.3 Z0.14 F2160.0 E2.375  
G1 X-38.99 Y-20.36 Z0.14 F2160.0 E2.378  
G1 X-38.93 Y-20.38 Z0.14 F2160.0 E2.381

### **3.7. Additional feature:**

Email Alert after printing job is finished:

- Made use of Internet of Things to send an automated email once the job is complete.
- Uses Arduino Uno and a Mechanical Button to detect once the printing is finished.

#### **3.7.1. Internet of Things:**

The **Internet of Things (IoT)** is the network of physical objects—devices, vehicles, buildings and other items—embedded with electronics, software, sensors, and network connectivity that enables these objects to collect and exchange data. The IoT allows objects to be sensed and controlled remotely across existing network infrastructure,<sup>[2]</sup> creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit; when IoT is augmented with sensors and actuators, the technology becomes an instance of the more general class of cyber-physical systems, which also encompasses technologies such as smart grids, smart homes, intelligent transportation and smart cities. Each thing is uniquely identifiable through its embedded computing system but is able to interoperate within the existing Internet infrastructure. Experts estimate that the IoT will consist of almost 50 billion objects by 2020.

British entrepreneur Kevin Ashton first coined the term in 1999 while working at Auto-ID Labs (originally called Auto-ID centers, referring to a global network of objects connected to radio-frequency identification, or RFID). Typically, IoT is expected to offer advanced connectivity of devices, systems, and services that goes beyond machine-to-machine (M2M) communications and covers a variety of protocols, domains, and applications. The interconnection of these embedded devices (including smart objects), is expected to usher in automation in nearly all fields, while also enabling advanced applications like a smart grid, and expanding to the areas such as smart cities.

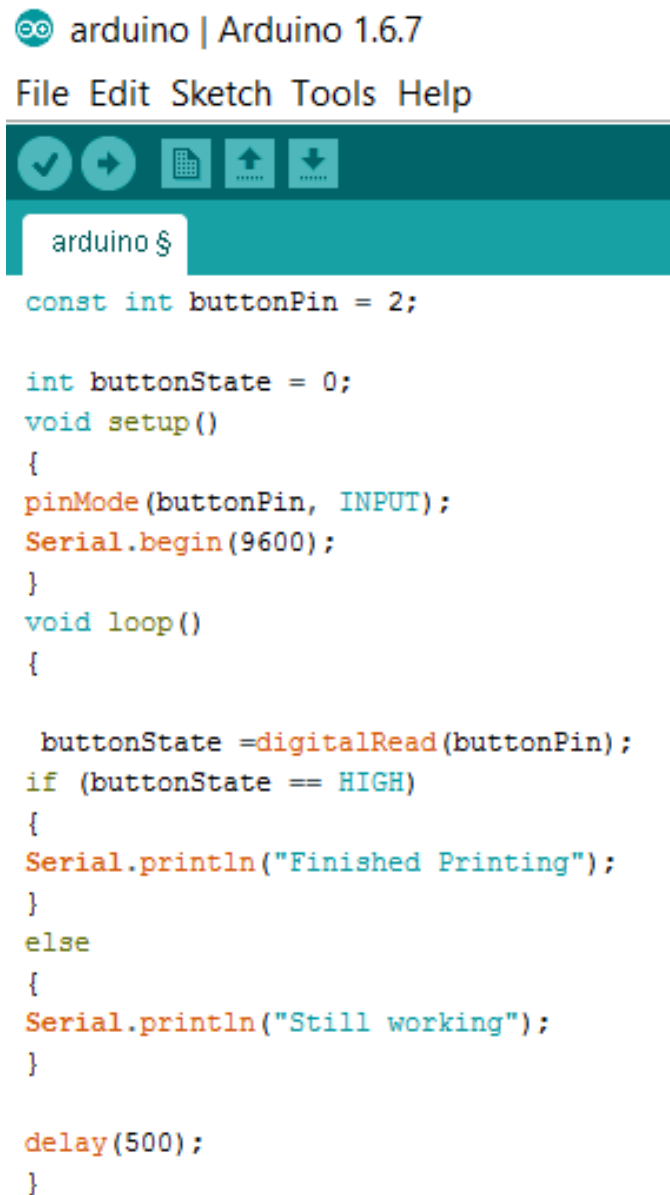


"Things," in the IoT sense, can refer to a wide variety of devices such as heart monitoring implants, biochip transponders on farm animals, electric clams in coastal waters, automobiles with built-in sensors, DNA analysis devices for environmental/food/pathogen monitoring or field operation devices that assist firefighters in search and rescue operations. Legal scholars suggest to look at "Things" as an "inextricable mixture of hardware, software, data and service". These devices collect useful data with the help of various existing technologies and then autonomously flow the data between other devices. Current market examples include smart thermostat systems and washer/dryers that use Wi-Fi for remote monitoring.

As well as the expansion of Internet-connected automation into a plethora of new application areas, IoT is also expected to generate large amounts of data from diverse locations, with the consequent necessity for quick aggregation of the data, and an increase in the need to index, store, and process such data more effectively. IoT is one of the platforms of today's Smart City, and Smart Energy Management Systems.



### 3.7.2. CODE:



The screenshot shows the Arduino IDE interface for version 1.6.7. The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu bar is a toolbar with icons for a checkmark, a right arrow, a grid, an up arrow, and a down arrow. The main editor area shows the following code:

```
arduino $  
  
const int buttonPin = 2;  
  
int buttonState = 0;  
void setup()  
{  
  pinMode(buttonPin, INPUT);  
  Serial.begin(9600);  
}  
void loop()  
{  
  
  buttonState =digitalRead(buttonPin);  
  if (buttonState == HIGH)  
  {  
    Serial.println("Finished Printing");  
  }  
  else  
  {  
    Serial.println("Still working");  
  }  
  
  delay(500);  
}
```

\*python.py - C:\Users\MitraVarun\Downloads\python.py (2.7.11)\*

File Edit Format Run Options Window Help

```
import time
import serial
import smtplib
TO = 'mechatronics.nyu@gmail.com'
GMAIL_USER = 'mechatronics.nyu@gmail.com'
GMAIL_PASS = 'nyupass123'
SUBJECT = '3D Printer!!!'
TEXT = 'Print completed'
ser = serial.Serial('COM4', 9600)
def send_email():
    print("Sending Email")
    smtpserver = smtplib.SMTP("smtp.gmail.com",587)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(GMAIL_USER, GMAIL_PASS)
    header = 'To:' + TO + '\n' + 'From: ' + GMAIL_USER
    header = header + '\n' + 'Subject:' + SUBJECT + '\n'
    print header
    msg = header + '\n' + TEXT + ' \n\n'
    smtpserver.sendmail(GMAIL_USER, TO, msg)
    smtpserver.close()
while True:
    message = ser.readline()
    print(message)
    if message[0] == 'F' :
        send_email()
    time.sleep(0.5)
```

#### **4.REFERENCES**

- [WWW.ARDUINO.CC](http://WWW.ARDUINO.CC)
- User: wdcritth at [www.github.com](http://www.github.com) for Inkscape Extension Template
- [www.makerbot.com](http://www.makerbot.com)