A

EECS 129B

Team Members
Jerry Yang
Rahul Vaidya
Parth Patel

Automated Etch-A-Sketch

Have you ever used an Etch-A-Sketch and wondered why it is so difficult to make an image by hand? Then you would know that it is often difficult to create such images simply because we do not possess the ability to move the dials with absolute precision. We aim to solve this problem by utilizing a computer in combination with an Arduino board and circuit board to control motors to operate as dials of an Etch-A-Sketch, allowing us to generate an image based off of a picture or image.

Description

Our main purpose was to design a way to use an Etch-A-Sketch to create simple and complex drawings without the error of human slight of hand. It is noticeable that we do not have the precision necessary to create an intricate and/or detailed drawing when using an Etch-A-Sketch because we perceive things differently and sometimes may inadvertently disorient drawings. The main advantage to this project is that a computer is able to emulate the instructions given to it and able to maintain proper manipulation of the dials allowing for a rather complex drawing. In the same respect, a computer has the ability to keep track of the organization of how it wishes to draw out what you would like to see. If you are drawing a house, you may not be able to perceive the edges by mere eyesight and could be off by a fraction of a pixel on one side, whereas a computer is able to calculate the exact sensitivity it places on the dial. A computer is also capable of drawing shapes that would require curves, including circles, because it is able to map the trajectory of the dials generate a suitable and proportioned shape. The images we generate may not be clear on the Etch-A-Sketch; we are able to use thousands of paths in a matter of minutes to create an image allowing us to draw complex images.

It is interesting to see how significant an improvement we get out of allowing an Etch-A-Sketch operated through a computer as opposed to by hand. An advantage we see is that a computer can process what we wish it to draw rather quickly since a computer is able to perform complex computations in far less time. It is the complication associated with using an Etch-A-Sketch, to draw a picture by hand, which intrigued us to make it easier. We monitored how a computer goes about generating the images that we once tried so hard to replicate. We are able to utilize the abilities of the computer in order to create complex images on an Etch-A-Sketch.

The main way we implemented the task of creating an automated Etch-A-Sketch involved the use of the Arduino board, a circuit board and motors. In the most basic sense we programmed the Arduino board with the instructions we wish the Etch-A-Sketch to follow in order to generate a picture. We used the circuit board to read in the signals sent by the Arduino board and then from the circuit board we attached the motors. We put these motors in place of the rotating dials allowing us to draw a picture in real-time and producing the result, in this case, the image. We

were able to see the computer's ability to handle the limits of the Etch-A-Sketch and see the improvements we could make to enhance the images.

It is true that the computer cannot capture the human touch in generating an image because we are prone to make errors whereas a computer is not. A significant advantage to carrying out the task of making an automated Etch-A-Sketch lies in the people that would be able to benefit from such a versatile tool. It would help children and adults learn how to go about drawing images using an Etch-A-Sketch at all levels. It can be used as a source of entertainment allowing individuals to show images being created with the sophistication of computer guidance and simulation. It can be used as a teaching tool to show students how with the use of technology we can manipulate a device as simple as this and make it unique and interesting. With the availability of the Etch-A-Sketch and the relative price range would also make it a prime example for interactive learning. Grade school children could each be given an Etch-A-Sketch in which a teacher would then simply tell the program what to output thus generating this output onto every Etch-A-Sketch. This would allow children to see things such as simple math problems and other study material allowing for a more interesting and unique way of learning. Students may also achieve better study habits in trying to understand what is taking place on the Etch-A-Sketch since once it is shaken, the screen is cleared, thus making them more alert and making their brains more active. Hence, with this project we wished to see the improvements it could make in the way that we learn.

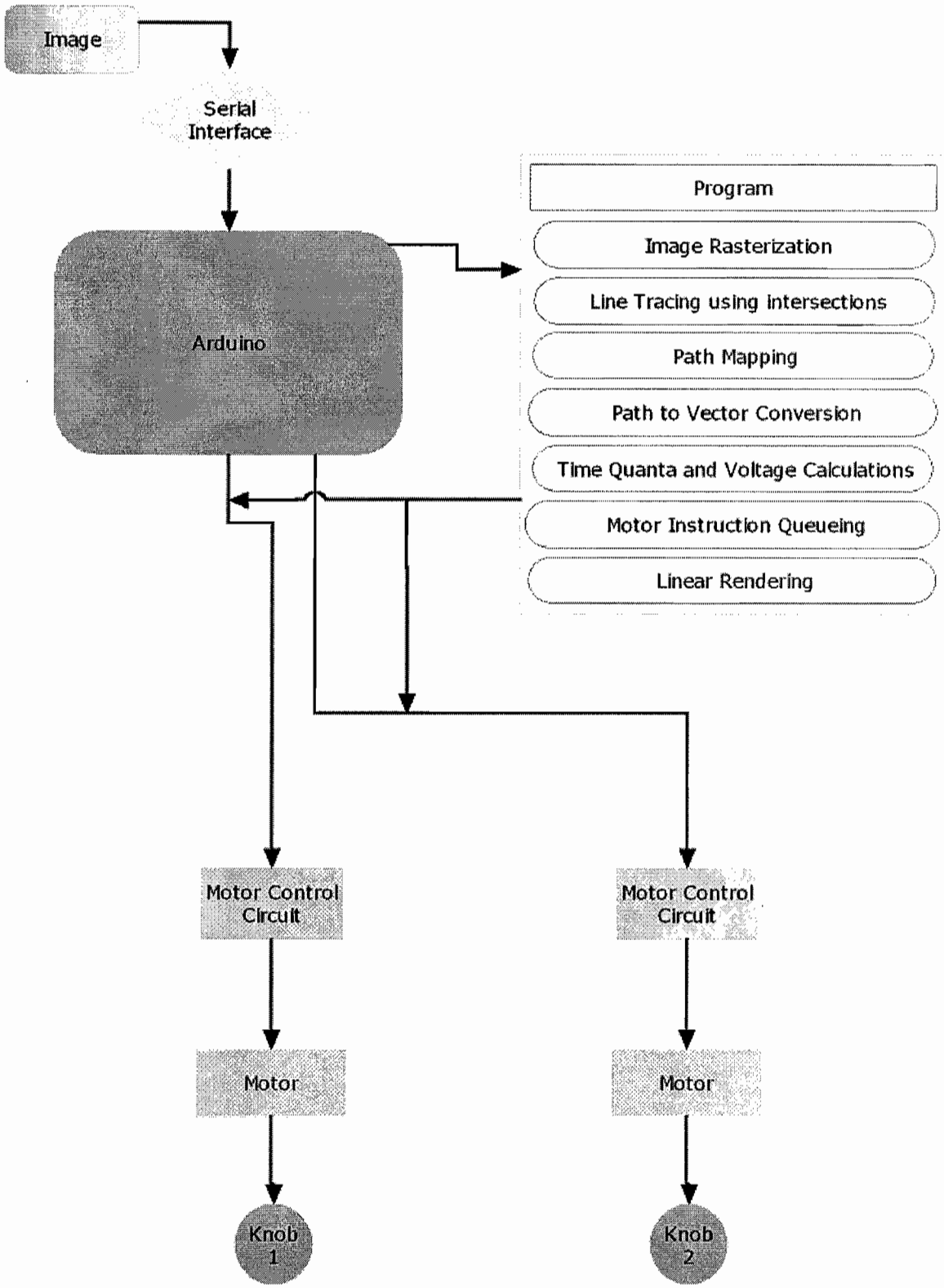
High Level Description

While potentially a simple a project, involving a couple of motors and a relatively simple circuit, the difficulty of this endeavor is complicated by the software required. Since we plan to use real images, we need to first parse the image data into an understandable format. For this application, the best format would be a monochrome raster pixel map, since we can then run algorithms on the pixel data and generate line vectors (since the Etch-a-Sketch draws lines) from this raster data. The Etch-a-Sketch only draws in lines, so we will have to implement methods of doing more advanced graphical work. For instance, if we wish to implement shading, we will have

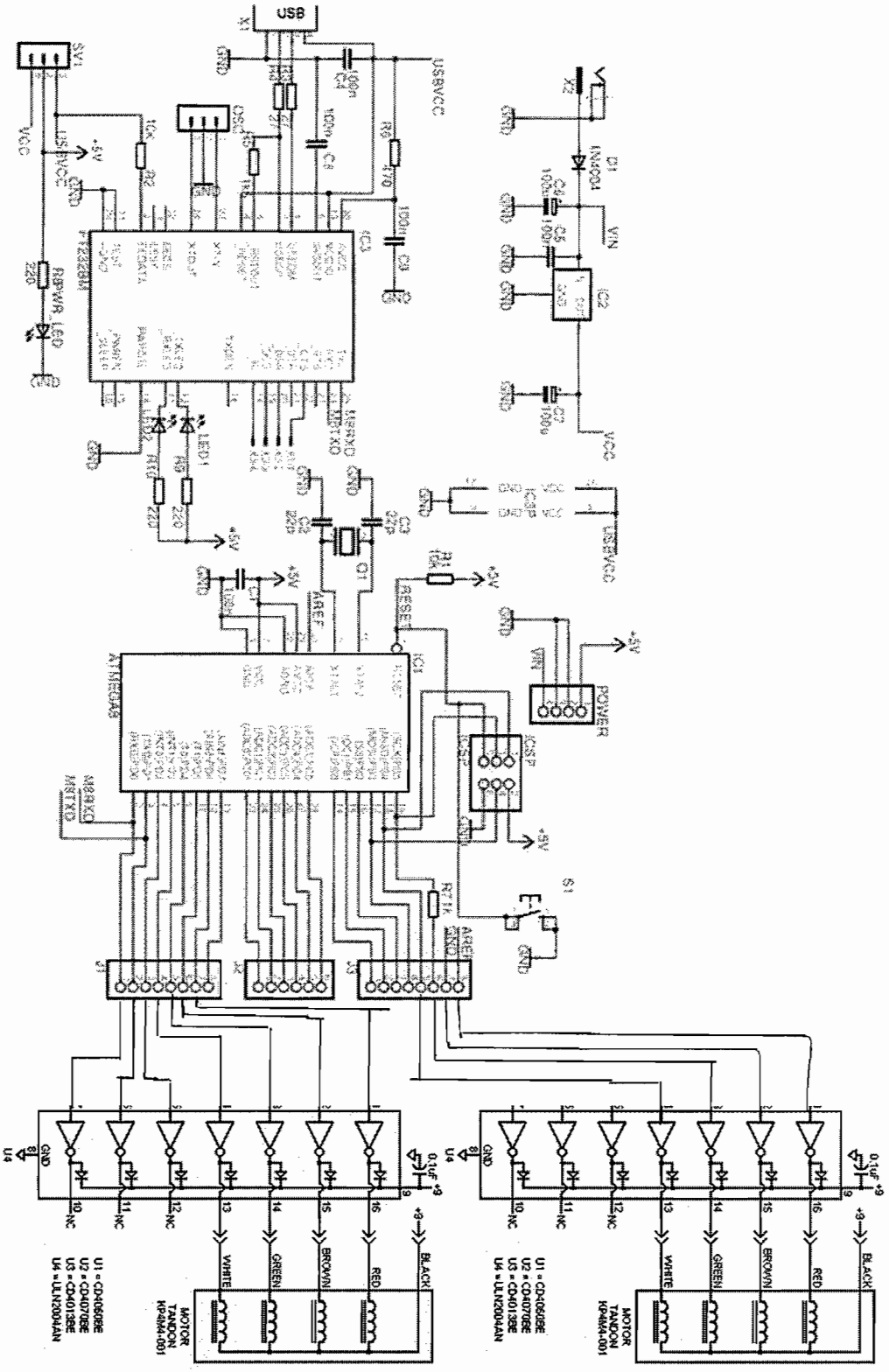
to use crisscross patterns since we have to use lines. After this, we will then need to serialize this data so that we can send it to the Arduino microcontroller board. There are two ways to approach this. One way is to send pure serialized image data to the Arduino, after which all the processing will be done. This will be rather intensive on the board itself. The other way is to do all the image processing, mapping, and instruction generating on the computer in C or another similar language (Java). Then, encoded instructions can be sent to the microcontroller, and all the controller has to do is execute them. This in itself is a non-trivial task, since the microcontroller has to operate the motors correctly. If we are using standard geared motors, we will need to calculate proper and accurate rotation time quanta and speeds. Variable speeds are accomplished through modulating the potentiometer (variable resistance will yield variable current, which in turn will vary the motor's speed). This, however, can lead to inaccuracies in the drawing, since geared motors are not measurable in discrete units. If we use stepper motors instead of geared motors, however, we do gain substantial accuracy. This comes at a cost though: the configuration and control of stepper motors is far more complicated than the control for geared motors, since there are eight wires to connect instead of two.

To reiterate, we are parsing and processing raw image data (most likely PPM pixel maps) as monochrome pixel data, which will then be used to compute a linear path throughout the picture. We will then implement that path through the motors by turning the Etch-a-Sketch knobs, which will render our picture.

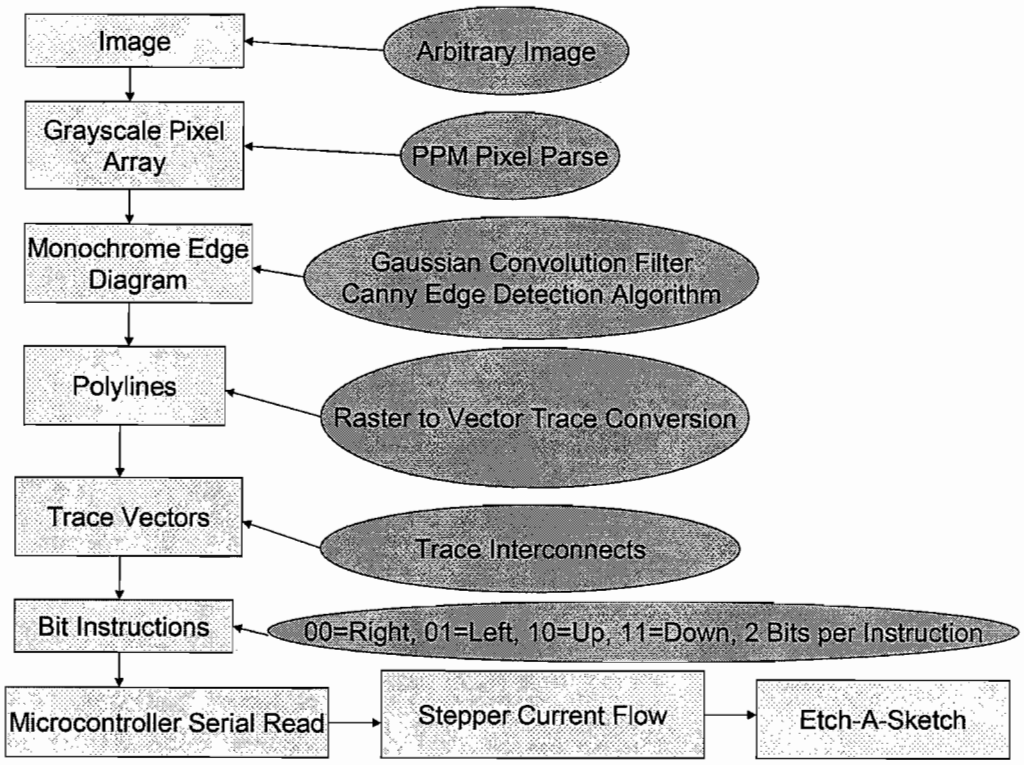
System Level Block Diagram

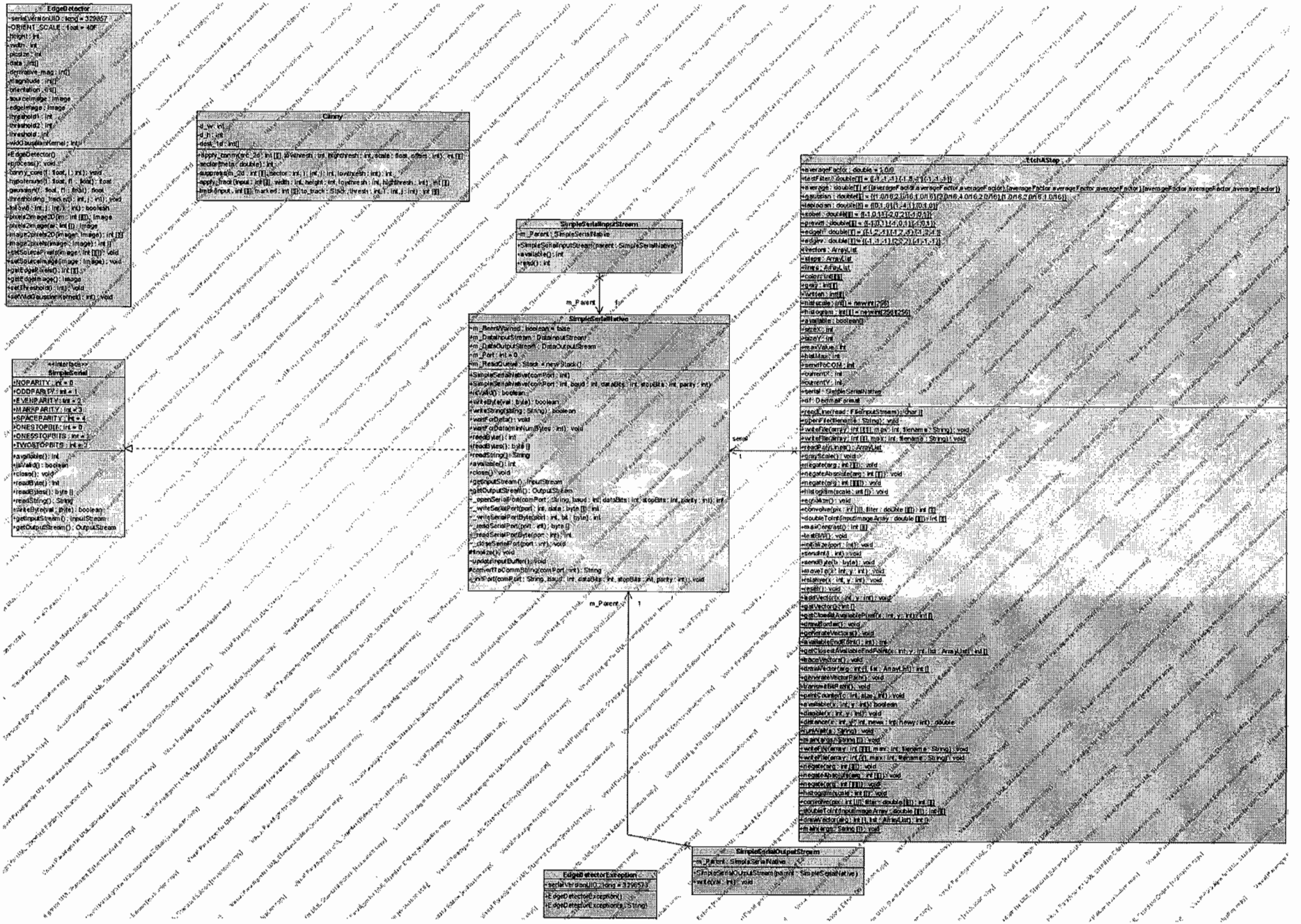


Circuit Level Diagram:



Software Description




EdgeDetector
Canny
SimpleSerialInputStream
SimpleSerialNative
«interface»
SimpleSerial
EtchASketch
SimpleSerialOutputStream
EdgeDetectorException
m_Parent
serial

System Test Plan

- **Wire the control circuit on a bread board and test to see if it is functional.**
    - **We will see if using a normal geared motor is a viable option.**
    - **The stepper motor gives more accuracy but allows faster design development. Additionally, the geared motor would probably supply a higher torque. This is an important concern because the knobs will resist being turned, and a stepper motor which uses magnetic fields to rotate might not supply sufficient torque to turn the knobs.**
- **Our motors require secure casings (they need to be affixed to the knobs for secure rotation, and also to the body of the Etch-A-Sketch so that they don't spin freely. We need to make sure that the casing can withstand the torque.**
- **Program a basic motor control scheme on the Arduino for further motor testing. This will include simple bounded rotations, without any complicated instructions. Basically, we want to get the motor control scheme down before we actually engage in any image work.**
- **Program an image drawing algorithm using a simple hardcoded or hand-drawn image. We may just have simple primitive geometric figures, or simple lines. Nothing too complicated.**
- **Create the parser that will read in the pixel map data from actual images, and attempt to draw them. Currently, we have a tentative image parser written in C for PPM files, which are basically the pixel maps that we want. We will need to convert the color data into proper, usable grayscale (if we wish to implement crosshatch shading), or pure black and white data (representable by one bit) if we do not wish to use any shading.**
- **If we wish to push the brunt of the processing to the computer rather than the Arduino, do so now. For more complicated images, we don't have the processing power or the space on the microcontroller for reliable rendering.**
- **Optimize certain parts of the circuit. In our hardware circuit, we use two muxes and two decoders, which use two transistors each. Perhaps we can cut down on our transistor**

count by creating a more efficient schematic. For now, the current schematic is functional, so it is fine, but in the future we may want to change it.

- **Make sure all components work completely. Again, we do not want to overload the motors with torque. If we absolutely need to, we may replace the motors with higher torque equivalents.**

Cost Analysis

- **Arduino USB Board: $31.95**
- **Transistor Array: $1**
- **Stepper Motors(2): $2.50(each)**
- **Printed Circuit Board: $15.95**
- **Etch-A-Sketch: $20**

Summary

Send instructions via PC to USB Arduino Board. Allow USB Arduino Board to control the circuit created using the transistor array, which in turn operates the two stepper motors in place of the dials on the Etch-A-Sketch. Watch as the resultant image is generated as per instructions sent from the initial program instructions.

Appendix – Code

EdgeDetector.java

```
import java.awt.*;
import java.awt.image.*;

public class EdgeDetector extends Component {
	static final long serialVersionUID = 329857;

	public EdgeDetector() {
		threshold1 = 10;
		threshold2 = 240;
		setThreshold(128);
		setWidGaussianKernel(15);
	}

	public void process() throws EdgeDetectorException {
		if (threshold < 0 || threshold > 255)
			throw new EdgeDetectorException("The value of the threshold is out of its
valid range.");
		if (widGaussianKernel < 3 || widGaussianKernel > 40)
			throw new EdgeDetectorException("The value of the widGaussianKernel is
out of its valid range.");
		width = sourceImage.getWidth(this);
		height = sourceImage.getHeight(this);
		picsize = width * height;
		data = new int[picsize];
		magnitude = new int[picsize];
		orientation = new int[picsize];
		float f = 1.0F;
		canny_core(f, widGaussianKernel);
		thresholding_tracker(threshold1, threshold2);
		for (int i = 0; i < picsize; i++)
```

```
                if (data[i] > threshold)
                        data[i] = 0xff000000;
                else
                        data[i] = -1;

        edgeImage = pixels2image(data);
        data = null;
        magnitude = null;
        orientation = null;
}

private void canny_core(float f, int i) {
        derivative_mag = new int[picsize];
        float af4[] = new float[i];
        float af5[] = new float[i];
        float af6[] = new float[i];
        data = image2pixels(sourceImage);
        int k4 = 0;
        do {
                if (k4 >= i)
                        break;
                float f1 = gaussian(k4, f);
                if (f1 <= 0.005F && k4 >= 2)
                        break;
                float f2 = gaussian((float) k4 - 0.5F, f);
                float f3 = gaussian((float) k4 + 0.5F, f);
                float f4 = gaussian(k4, f * 0.5F);
                af4[k4] = (f1 + f2 + f3) / 3F / (6.283185F * f * f);
                af5[k4] = f3 - f2;
                af6[k4] = 1.6F * f4 - f1;
                k4++;
        } while (true);
        int j = k4;
        float af[] = new float[picsize];
        float af1[] = new float[picsize];
        int j1 = width - (j - 1);
        int l = width * (j - 1);
        int i1 = width * (height - (j - 1));
        for (int l4 = j - 1; l4 < j1; l4++) {
                for (int l5 = l; l5 < i1; l5 += width) {
                        int k1 = l4 + l5;
                        float f8 = (float) data[k1] * af4[0];
                        float f10 = f8;
                        int l6 = 1;
                        int k7 = k1 - width;
                        for (int i8 = k1 + width; l6 < j; i8 += width) {
                                f8 += af4[l6] * (float) (data[k7] + data[i8]);
                                f10 += af4[l6] * (float) (data[k1 - l6] + data[k1 + l6]);
                                l6++;
                                k7 -= width;
                        }

                        af[k1] = f8;
                        af1[k1] = f10;
                }
```

```
}

float af2[] = new float[picsize];
for (int i5 = j - 1; i5 < j1; i5++) {
        for (int i6 = l; i6 < i1; i6 += width) {
                float f9 = 0.0F;
                int l1 = i5 + i6;
                for (int i7 = 1; i7 < j; i7++)
                        f9 += af5[i7] * (af[l1 - i7] - af[l1 + i7]);

                af2[l1] = f9;
        }

}

af = null;
float af3[] = new float[picsize];
for (int j5 = k4; j5 < width - k4; j5++) {
        for (int j6 = l; j6 < i1; j6 += width) {
                float f11 = 0.0F;
                int i2 = j5 + j6;
                int j7 = 1;
                for (int l7 = width; j7 < j; l7 += width) {
                        f11 += af5[j7] * (af1[i2 - l7] - af1[i2 + l7]);
                        j7++;
                }

                af3[i2] = f11;
        }

}

af1 = null;
j1 = width - j;
l = width * j;
i1 = width * (height - j);
for (int k5 = j; k5 < j1; k5++) {
        for (int k6 = l; k6 < i1; k6 += width) {
                int j2 = k5 + k6;
                int k2 = j2 - width;
                int l2 = j2 + width;
                int i3 = j2 - 1;
                int j3 = j2 + 1;
                int k3 = k2 - 1;
                int l3 = k2 + 1;
                int i4 = l2 - 1;
                int j4 = l2 + 1;
                float f6 = af2[j2];
                float f7 = af3[j2];
                float f12 = hypotenuse(f6, f7);
                int k = (int) ((double) f12 * 20D);
                derivative_mag[j2] = k >= 256 ? 255 : k;
                float f13 = hypotenuse(af2[k2], af3[k2]);
                float f14 = hypotenuse(af2[l2], af3[l2]);
                float f15 = hypotenuse(af2[i3], af3[i3]);
                float f16 = hypotenuse(af2[j3], af3[j3]);
```

```
                        float f18 = hypotenuse(af2[l3], af3[l3]);
                        float f20 = hypotenuse(af2[j4], af3[j4]);
                        float f19 = hypotenuse(af2[i4], af3[i4]);
                        float f17 = hypotenuse(af2[k3], af3[k3]);
                        float f5;
                        if (f6 * f7 <= (float) 0
                                ? Math.abs(f6) >= Math.abs(f7)
                                ? (f5 = Math.abs(f6 * f12))
                                        >= Math.abs(f7 * f18 - (f6 + f7) * f16)
                                && f5
                                        > Math.abs(f7 * f19 - (f6 + f7) * f15) : (
                                                f5 = Math.abs(f7 * f12))
                                        >= Math.abs(f6 * f18 - (f7 + f6) * f13)
                                && f5
                                        > Math.abs(f6 * f19 - (f7 + f6) * f14) : Math.abs(f6)
                                        >= Math.abs(f7)
                                        ? (f5 = Math.abs(f6 * f12))
                                                >= Math.abs(f7 * f20 + (f6 - f7) * f16)
                                && f5
                                        > Math.abs(f7 * f17 + (f6 - f7) * f15) : (
                                                f5 = Math.abs(f7 * f12))
                                        >= Math.abs(f6 * f20 + (f7 - f6) * f14)
                                && f5 > Math.abs(f6 * f17 + (f7 - f6) * f13)) {
                                magnitude[j2] = derivative_mag[j2];
                                orientation[j2] = (int) (Math.atan2(f7, f6) * (double) 40F);
                        }
                }

        }

        derivative_mag = null;
        af2 = null;
        af3 = null;
}

private float hypotenuse(float f, float f1) {
        if (f == 0.0F && f1 == 0.0F)
                return 0.0F;
        else
                return (float) Math.sqrt(f * f + f1 * f1);
}

private float gaussian(float f, float f1) {
        return (float) Math.exp((-f * f) / ((float) 2 * f1 * f1));
}

private void thresholding_tracker(int i, int j) {
        for (int k = 0; k < picsize; k++)
                data[k] = 0;

        for (int l = 0; l < width; l++) {
                for (int i1 = 0; i1 < height; i1++)
                        if (magnitude[l + width * i1] >= i)
                                follow(l, i1, j);

        }
```

```
}

private boolean follow(int i, int j, int k) {
        int j1 = i + 1;
        int k1 = i - 1;
        int l1 = j + 1;
        int i2 = j - 1;
        int j2 = i + j * width;
        if (l1 >= height)
                l1 = height - 1;
        if (i2 < 0)
                i2 = 0;
        if (j1 >= width)
                j1 = width - 1;
        if (k1 < 0)
                k1 = 0;
        if (data[j2] == 0) {
                data[j2] = magnitude[j2];
                boolean flag = false;
                int l = k1;
                do {
                        if (l > j1)
                                break;
                        int i1 = i2;
                        do {
                                if (i1 > l1)
                                        break;
                                int k2 = l + i1 * width;
                                if ((i1 != j || l != i)
                                        && magnitude[k2] >= k
                                        && follow(l, i1, k)) {
                                        flag = true;
                                        break;
                                }
                                i1++;
                        } while (true);
                        if (!flag)
                                break;
                        l++;
                }
                while (true);
                return true;
        } else {
                return false;
        }
}
private Image pixels2image2D(int[][] im) {
        int d_w = im.length;
        int d_h = im[0].length;
        width = im.length;
        height = im[0].length;
        int[] tmp_1d = new int[d_w*d_h];
        for(int i = 0; i < d_w; i++){
                for(int j = 0; j < d_h; j++){
                        tmp_1d[i+(j*d_w)] = im[i][j];
```

```
                }
        }
        return pixels2image(tmp_1d);
}
private Image pixels2image(int ai[]) {
        MemoryImageSource memoryimagesource =
                new MemoryImageSource(
                        width,
                        height,
                        ColorModel.getRGBdefault(),
                        ai,
                        0,
                        width);
        return Toolkit.getDefaultToolkit().createImage(memoryimagesource);
}
private int[][] image2pixels2D(Image image) {
        int[] tmp_1d = image2pixels(image);
        int d_w = width;
        int d_h = height;
        int[][] src_2d = new int[width][height];
        for(int i = 0; i < d_w; i++){
                for(int j = 0; j < d_h; j++){
                        src_2d[i][j] = tmp_1d[i+(j*d_w)];
                }
        }
        return src_2d;
}
private int[] image2pixels(Image image) {
        int ai[] = new int[picsize];
        PixelGrabber pixelgrabber =
                new PixelGrabber(image, 0, 0, width, height, ai, 0, width);
        try {
                pixelgrabber.grabPixels();
        } catch (InterruptedException interruptedexception) {
                interruptedexception.printStackTrace();
        }
        boolean flag = false;
        int k1 = 0;
        do {
                if (k1 >= 16)
                        break;
                int i = (ai[k1] & 0xff0000) >> 16;
                int k = (ai[k1] & 0xff00) >> 8;
                int i1 = ai[k1] & 0xff;
                if (i != k || k != i1) {
                        flag = true;
                        break;
                }
                k1++;
        } while (true);
        if (flag) {
                for (int l1 = 0; l1 < picsize; l1++) {
                        int j = (ai[l1] & 0xff0000) >> 16;
                        int l = (ai[l1] & 0xff00) >> 8;
                        int j1 = ai[l1] & 0xff;
                        ai[l1] =
```

```
                                        (int) (0.29799999999999999D * (double) j
                                                + 0.58599999999999997D * (double) l
                                                + 0.113D * (double) j1);
                        }

                } else {
                        for (int i2 = 0; i2 < picsize; i2++)
                                ai[i2] = ai[i2] & 0xff;

                }
                return ai;
        }
        public void setSourcePixels(int[][] image) {
                setSourceImage(pixels2image2D(image));
        }

        public void setSourceImage(Image image) {
                sourceImage = image;
        }
        public int[][] getEdgePixels() {
                return image2pixels2D(getEdgeImage());
        }
        public Image getEdgeImage() {
                return edgeImage;
        }

        public void setThreshold(int i) {
                threshold = i;
        }

        public void setWidGaussianKernel(int i) {
                widGaussianKernel = i;
        }

        final float ORIENT_SCALE = 40F;
        private int height;
        private int width;
        private int picsize;
        private int data[];
        private int derivative_mag[];
        private int magnitude[];
        private int orientation[];
        private Image sourceImage;
        private Image edgeImage;
        private int threshold1;
        private int threshold2;
        private int threshold;
        private int widGaussianKernel;
}
```

EdgeDetectorException.java

```
public class EdgeDetectorException extends Exception
{
        static final long serialVersionUID = 3298573;
  public EdgeDetectorException()
```

```
  {
    //do something?
  }
  public EdgeDetectorException(String s)
  {
    super(s);
  }
}
```

etch.pde

```
/*
  Etch-A-Sketch Control Program
 Written by Rahul Vaidya
 */
// 560 374
int hpins[4] = {
  2,3,4,5};
int vpins[4] = {
  6,7,8,9};
int hstate = 0;
int vstate = 0;
int steps[16] = {
  1,0,1,0,0,1,1,0,0,1,0,1,1,0,0,1};


void setup() {
  for(int i=0;i<4;i++) {
    pinMode(hpins[i],OUTPUT);
    pinMode(vpins[i],OUTPUT);
  }
  Serial.begin(9600);
  for (int i=0; i<3; i++) {
    digitalWrite(13, HIGH);
    delay(200);
    digitalWrite(13, LOW);
    delay(200);
  }
}

void loop() {
  if(Serial.available() >= 3) {
    int h = Serial.read();
    int v = Serial.read();
    int m = Serial.read();
    //relative((int)h*hstep,(int)v*vstep, m);
    relative(h,v,m);
    if(Serial.available() == 0) Serial.print(1);
  }
}
void relative(int x, int y, int m) {
  float slope;
  if(x == 0) {
    if(y == 0) return;
    else slope = 999;
  }
```

```
else slope = (float)y / (float)x;
int cx = 0;
int cy = 0;
int last = -1;
int xs, ys, ax, ay;
ax = 99;
ay = 99;
float currentSlope;
if(m == 0) {
  xs = 1;
  ys = -1;
}
else if(m == 1) {
  xs = -1;
  ys = -1;
}
else if(m == 2) {
  xs = 1;
  ys = 1;
}
else if(m == 3) {
  xs = -1;
  ys = 1;
}
while(ax >= 1 || ay >= 1) {
  ax = x - cx;
  ay = y - cy;
  if(ax == 0) currentSlope = 998;
  else currentSlope = (float)ay / (float)ax;
  if(currentSlope >= slope) {
    cx++;
    hstate += xs;
    if(hstate >= 4) hstate = 0;
    if(hstate <= -1) hstate = 3;
    for(int j=0;j<4;j++) {
      digitalWrite(hpins[j],steps[hstate*4+j]);
    }
    if(last == 0 || last == -1) {
      delay(40);
    }
    else {
      delay(20);
    }
    last = 0;
  }
  else {
    cy++;
    vstate += ys;
    if(vstate >= 4) vstate = 0;
    if(vstate <= -1) vstate = 3;
    for(int j=0;j<4;j++) {
      digitalWrite(vpins[j],steps[vstate*4+j]);
    }
    if(last == 1 || last == -1) {
      delay(40);
    }
```

```
    else {
      delay(20);
    }
    last = 1;
   }
  }
}
```

EtchAStep.java

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.ArrayList;
public class EtchAStep {
        public static final double averageFactor = 1.0/9;
        public static final double[][] testFilter = {{-1,-1,-1},{-1,8,-1},{-1,-1,-1}};
        public static final double[][] average =
{{averageFactor,averageFactor,averageFactor},{averageFactor,averageFactor,averageFactor},{ave
rageFactor,averageFactor,averageFactor}};
        public static final double[][] gaussian =
{{1.0/16,2.0/16,1.0/16},{2.0/16,4.0/16,2.0/16},{1.0/16,2.0/16,1.0/16}};
        public static final double[][] laplacian = {{0,1,0},{1,-4,1},{0,1,0}};
        public static final double[][] sobel = {{-1,0,1},{-2,0,2},{-1,0,1}};
        public static final double[][] prewitt = {{-1,0,1},{-1,0,1},{-1,0,1}};
        public static final double[][] edgeh = {{-1,2,-1},{-1,2,-1},{-1,2,-1}};
        public static final double[][] edgev = {{-1,-1,-1},{2,2,2},{-1,-1,-1}};
        public static ArrayList vectors, steps, lines;
        public static int[][][] color;
        public static int[][] gray;
        public static int[][] written;
        public static int[] histscale = new int[256];
        public static int[][] histogram = new int[256][256];
        public static boolean[] available;
        public static int sizeX, sizeY, maxValue, histMax, sendToCOM, currentX, currentY,
threshold;
        public static SimpleSerialNative serial;
        public static DecimalFormat df;
        public static char[] readLine(FileInputStream read) throws IOException {
                char temp = 'h';
                char[] agg = new char[100];
                int count = 0;
                while(temp != '\n') {
                        temp = (char) read.read();
                        agg[count] = temp;
                        count++;
                }
                char[] returnVal = new char[count];
                for(int i=0;i<count;i++) {
                        returnVal[i] = agg[i];
                }
```

```
        return returnVal;
}
public static void openFile(String filename) {
        char[] buffer;
        try {
                FileInputStream read = new FileInputStream(filename);
                buffer = readLine(read);
                if((buffer[0] != 'P') || (buffer[1] != '6')) {
                        System.out.println("Incorrect filetype!");
                        System.exit(0);
                }
                do buffer = readLine(read); while(buffer[0] == '#');
                String dims = String.copyValueOf(buffer);
                int sep = dims.indexOf(" ");
                sizeX = Integer.parseInt(dims.substring(0,sep));
                sizeY = Integer.parseInt(dims.substring(sep+1).trim());
                buffer = readLine(read);
                dims = String.copyValueOf(buffer);
                maxValue = Integer.parseInt(dims.trim());
                color = new int[sizeX][sizeY][3];
                int temp = 0;
                for(int j=0;j<sizeY;j++) {
                        for(int i=0;i<sizeX;i++) {
                                for(int k=0;k<3;k++) {
                                        temp = read.read();
                                        color[i][j][k] = temp;

                                }
                        }
                }
        } catch(Exception e) { e.printStackTrace(); }
}
public static void writeFile(int[][][] array, int max, String filename) {
        try {
                FileOutputStream fw = new FileOutputStream(filename);
                fw.write((byte) 'P');
                fw.write((byte) '6');
                fw.write((byte) '\n');
                String t = array.length + "";
                for(int i=0;i<t.length();i++) fw.write((byte) t.charAt(i));
                fw.write((byte) ' ');
                t = array[0].length + "";
                for(int i=0;i<t.length();i++) fw.write((byte) t.charAt(i));
                fw.write((byte) '\n');
                t = max + "";
                for(int i=0;i<t.length();i++) fw.write((byte) t.charAt(i));
                fw.write((byte) '\n');
                for(int i=0;i<array[0].length;i++) {
                        for(int j=0;j<array.length;j++) {
                                for(int k=0;k<3;k++) {
                                        fw.write(array[j][i][k]);
                                }
                        }
                }
                fw.flush();
        } catch(Exception e) { e.printStackTrace(); }
```

```
}
public static void writeFile(int[][] array, int max, String filename) {
	try {
		FileOutputStream fw = new FileOutputStream(filename);
		fw.write((byte) 'P');
		fw.write((byte) '5');
		fw.write((byte) '\n');
		String t = array.length + "";
		for(int i=0;i<t.length();i++) fw.write((byte) t.charAt(i));
		fw.write((byte) ' ');
		t = array[0].length + "";
		for(int i=0;i<t.length();i++) fw.write((byte) t.charAt(i));
		fw.write((byte) '\n');
		t = max + "";
		for(int i=0;i<t.length();i++) fw.write((byte) t.charAt(i));
		fw.write((byte) '\n');
		for(int i=0;i<array[0].length;i++) {
			for(int j=0;j<array.length;j++) {
				fw.write(array[j][i]);
			}
		}
		fw.flush();
	} catch(Exception e) { e.printStackTrace(); }
}
public static ArrayList readPolyLines() {
	ArrayList polylines = new ArrayList();
	ArrayList polyline = new ArrayList();
	int[] vector = new int[2];
	FileReader fr = null;
	BufferedReader br;
	try {
		fr = new FileReader("input.ply");
		br = new BufferedReader(fr);
		String b, t1, t2;
		int index;
		b = br.readLine();
		while(b != null) {
			if(b.trim().equals("POLYLINE")) {
				polyline = new ArrayList();
				polylines.add(polyline);
				b = br.readLine();
				continue;
			}
			if(b.trim().equals("END")) {
				b = br.readLine();
				continue;
			}
			index = b.indexOf(" ");
			t1 = b.substring(0,index+1).trim();
			t2 = b.substring(index).trim();
			vector[0] = Integer.parseInt(t1);
			vector[1] = Integer.parseInt(t2);
			polyline.add(vector);
			vector = new int[2];
			b = br.readLine();
		}
```

```
                br.close();
                fr.close();
                return polylines;
            } catch(Exception e) { e.printStackTrace(); }
            return null;
        }
        public static void grayScale() {
            int max = 0;
            int temp = 0;
            histMax = 0;
            gray = new int[sizeX][sizeY];
            for(int i=0;i<sizeX;i++) {
                for(int j=0;j<sizeY;j++) {
                    temp = (int) Math.round((color[i][j][0] + color[i][j][1] + color[i][j][2] ) /
3.0);
                    if(temp > max) max = temp;
                    gray[i][j] = temp;
                    histscale[temp]++;
                    if(histscale[temp] > histMax) histMax = histscale[temp];
                }
            }
            maxValue = max;
        }
        public static void negate(int[][] arg) {
            for(int i=0;i<sizeX;i++) {
                for(int j=0;j<sizeY;j++) {
                    arg[i][j] = maxValue - arg[i][j];
                }
            }
        }
        public static void negateAbsolute(int[][] arg) {
            for(int i=0;i<sizeX;i++) {
                for(int j=0;j<sizeY;j++) {
                    arg[i][j] = 255 - arg[i][j];
                }
            }
        }
        public static void negate(int[][][] arg) {
            for(int i=0;i<sizeX;i++) {
                for(int j=0;j<sizeY;j++) {
                    for(int k=0;k<3;k++) {
                        color[i][j][k] = maxValue - arg[i][j][k];
                    }
                }
            }
        }
        public static void histogram(int[] scale) {
            for(int i=0;i<256;i++) {
                for(int j=0;j<scale[i] * 255 / histMax;j++) {
                    histogram[i][j] = 255;
                }
            }
        }
        public static void equalize() {
            double p = 0;
            double s = 0;
```

```
                histMax = 0;
                for(int i=0;i<256;i++) {
                        p = ((double) histscale[i])/(sizeX*sizeY);
                        s += p * maxValue;
                        histscale[i] = (int) Math.round(s);
                }
                int[] temph = new int[256];
                for(int i=0;i<sizeX;i++) {
                        for(int j=0;j<sizeY;j++) {
                                gray[i][j] = histscale[gray[i][j]];
                                temph[gray[i][j]]++;
                                if(temph[gray[i][j]] > histMax) histMax = temph[gray[i][j]];
                        }
                }
                histscale = temph;
        }
        public static int[][] convolve(int[][] pix, double[][] filter) {
                double normConst = 0.0;
                for(int i = 0; i < 3; i++)
                        for(int j = 0; j < 3; j++)
                                normConst += filter[i][j];
                if(normConst == 0.0) normConst = 1.0;
                normConst = 1.0;
                double[][] coutput = new double[sizeX][sizeY];
                int ix, iy;
                for(int i=0;i<sizeX;i++) {
                        for(int j=0;j<sizeY;j++) {
                                for(int k=0;k<filter.length;k++) {
                                        for(int l=0;l<filter[0].length;l++) {
                                                ix = i+k-(filter.length/2);
                                                iy = j+l-(filter[0].length/2);
                                                if(ix < 0) ix += sizeX;
                                                if(ix >= sizeX) ix -= sizeX;
                                                if(iy < 0) iy += sizeY;
                                                if(iy >= sizeY) iy -= sizeY;
                                                coutput[i][j] += pix[ix][iy]*filter[k][l]/normConst;
                                                // asserts
                                                //if(((k == 1 && l == 1) && (ix != i || iy != j)) ||
(coutput[i][j] > 255) || (coutput[i][j] <= 0)) System.out.print("*");
                                                //if((ix > i+1 && i != 0 && k != 0) || (ix < i-1 && i !=
sizeX-1 && k != 2) || (iy > j+1 && j != 0 && l != 0) || (iy < j-1 && j != sizeY-1 && l != 2))
System.out.print("+");
                                        }
                                }
                                if(coutput[i][j] > 255) coutput[i][j] = 255;
                                if(coutput[i][j] < 0) coutput[i][j] = 0;
                        }
                }
                return doubleToInt(coutput);
        }
        static int[][] doubleToInt(double[][] inputImageArray) {

                int numImgRows = inputImageArray.length;
                int numImgCols = inputImageArray[0].length;

                int[][] outputImageArray = new int[numImgRows][numImgCols];
```

```
        for(int row = 0;row < numImgRows;row++){
                for(int col = 0;col < numImgCols;col++){
                        outputImageArray[row][col] =
                                (int)inputImageArray[row][col];
                }
        }
        return outputImageArray;
}
public static int[][] maxContrast() {
        for(int i=0;i<gray.length;i++) {
                for(int j=0;j<gray[0].length;j++) {
                        if(gray[i][j] < 127) gray[i][j] = 0;
                        else gray[i][j] = 255;
                }
        }
        return gray;
}
public static void testBW() {
        System.out.println("");
        for(int i=0;i<gray.length;i++) {
                for(int j=0;j<gray[0].length;j++) {
                        if(gray[i][j] < 255 && gray[i][j] != 0) System.out.print(".");
                        if(gray[i][j] > 0 && gray[i][j] != 255) System.out.print(".");
                }
        }
        System.out.println("");
}
public static void initialize(int port) {
        if(sendToCOM == 1) serial = new SimpleSerialNative(port,9600,8,0,0);
        //serial = new SimpleSerialNative(port);
        currentX = 0;
        currentY = 0;
        written = new int[gray.length][gray[0].length];
        for(int i=0;i<written.length;i++) {
                for(int j=0;j<written[0].length;j++) {
                        written[i][j] = 255;
                }
        }
        df = new DecimalFormat("000.00");
        vectors = new ArrayList();
        steps = new ArrayList();
}
public static void sendInt(int i) {
        //String s = Integer.toBinaryString(i);
        //byte b = Byte.parseByte(s);
        sendByte((byte) i);
}
public static void sendByte(byte b) {
        if(sendToCOM == 1) {
                serial.writeByte(b);
        }
}
public static void moveTo(int x, int y) {
        int relX = x - currentX;
        int relY = y - currentY;
        relative(relX,relY);
```

```
                //System.out.print("x" + relX + "," + relY + " ");
                currentX = x;
                currentY = y;
        }
        public static void relative(int x, int y) {
                float slope;
                int absx = Math.abs(x), absy = Math.abs(y), cx = 0, cy = 0, xs, ys, ax, ay;
                ax = absx - cx;
                ay = absy - cy;
                if(x < 0) xs = 1;
                else xs = 0;
                if(y < 0) ys = 2;
                else ys = 3;
                if(x == 0) {
                        if(y == 0) return;
                        else slope = Float.MAX_VALUE;
                }
                else slope = (float)ay/(float)ax;
                //System.out.println("X: " + x + " Y: " + y + " AX: " + ax + " AY: " + ay + " Slope: " +
slope);
                while(ax >= 1 || ay >= 1) {
                        if(ax == 0) {
                                slope = Float.MAX_VALUE;
                        }
                        else slope = (float)ay/(float)ax;
                        if(slope == 1) {
                                cx++;
                                cy++;
                                steps.add(new Integer(xs));
                                steps.add(new Integer(ys));
                                //System.out.println(4);
                        }
                        else if(slope < 1) {
                                cx++;
                                steps.add(new Integer(xs));
                                //System.out.println(xs);
                        }
                        else {
                                cy++;
                                steps.add(new Integer(ys));
                                //System.out.println(ys);
                        }
                        ax = absx - cx;
                        ay = absy - cy;
                }
                //System.out.println("Done");
                int[] output = new int[steps.size()];
                for(int i=0;i<steps.size();i++) {
                        output[i] = ((Integer) steps.get(i)).intValue();
                }
        }
        public static void reset() {
                moveTo(0,0);
        }
        public static void addVector(int x, int y) {
                int[] temp = new int[2];
```

```
        temp[0] = x;
        if(temp[0] > 255) temp[0] = 255;
        if(temp[0] < 0) temp[0] = 0;
        temp[1] = y;
        if(temp[1] > 255) temp[1] = 255;
        if(temp[1] < 0) temp[1] = 0;
        vectors.add(temp);
}
public static int[] getVector() {
        int[] temp = (int[]) vectors.get(0);
        vectors.remove(0);
        return temp;
}
public static int[] getClosestAvailablePixel(int x, int y) {
        int[] tmp = new int[2];
        double d,td;
        d = Double.MAX_VALUE;
        int ii, jj;
        for(int i=x-2;i<x+2;i++) {
                for(int j=y-2;j<y+2;j++) {
                        ii = i;
                        jj= j;
                        if(i < 0) ii = 0;
                        else if(i > 255) ii = 255;
                        if(j < 0) jj = 0;
                        else if(j > 255) jj = 255;
                        td = distance(x,y,ii,jj);
                        if(td < d && available(ii,jj)) {
                                tmp[0] = ii;
                                tmp[1] = jj;
                                d = td;
                        }
                }
        }
        if(d == Double.MAX_VALUE) {
                for(int i=0;i<gray.length;i++) {
                        for(int j=0;j<gray[0].length;j++) {
                                td = distance(x,y,i,j);
                                if(td < d && available(i,j)) {
                                        tmp[0] = i;
                                        tmp[1] = j;
                                        d = td;
                                }
                        }
                }
        }
        if(d == Double.MAX_VALUE) return null;
        else {
                disable(tmp[0],tmp[1]);
                return tmp;
        }
}
public static void drawBorder() {
        addVector(255,0);
        addVector(255,255);
        addVector(0,255);
```

```
        addVector(0,0);
}
public static void generateVectors() {
        int cx = 0,cy = 0;
        int[] vector = getClosestAvailablePixel(cx,cy);
        while(vector != null) {
                cx = vector[0];
                cy = vector[1];
                addVector(cx,cy);
                vector = getClosestAvailablePixel(cx,cy);
        }
}
public static int availableEndPoint(int i) {
        int tmp = i;
        if(i % 2 == 1) tmp--;
        tmp /= 2;
        if(available[tmp]) return tmp;
        else return -1;
}
public static int[] getClosestAvailableEndPoint(int x, int y, ArrayList list) {
        int[] tmp = new int[3];
        int[] end;
        double d,td;
        d = Double.MAX_VALUE;
        int index = -1;
        for(int i=0;i<list.size();i++) {
                end = (int[]) list.get(i);
                td = distance(x,y,end[0],end[1]);
                int available = availableEndPoint(i);
                if(td < d && available >= 0) {
                        tmp[0] = end[0];
                        tmp[1] = end[1];
                        tmp[2] = available;
                        d = td;
                        index = available;
                }
        }
        if(d == Double.MAX_VALUE) return null;
        else {
                available[index] = false;
                return tmp;
        }
}
public static void traceVectors() {
        lines = readPolyLines();
        ArrayList line;
        ArrayList endpoints = new ArrayList();
        available = new boolean[lines.size()];
        int[] tmp;
        int cx = 0, cy = 0;
        for(int i=0;i<available.length;i++) available[i] = true;
        for(int i=0;i<lines.size();i++) {
                line = (ArrayList) lines.get(i);
                if(line.size() == 0) {
                        System.out.println("Error. Empty Line!");
                        System.exit(0);
```

```
                }
                endpoints.add((int[]) line.get(0));
                endpoints.add((int[]) line.get(line.size()-1));
        }
        for(int i=0;i<lines.size();i++) {
                tmp = getClosestAvailableEndPoint(cx, cy, endpoints);
                if(tmp == null) break;
                cx = tmp[0];
                cy = tmp[1];
                //System.out.println(tmp[0] + " " + tmp[1] + " " + tmp[2]);
                moveTo(cx, cy);
                //currentX = cx;
                //currentY = cy;
                tmp = drawVector(tmp, lines);
                cx = tmp[0];
                cy = tmp[1];
        }
        steps.add(new Integer(0));
}
public static int[] drawVector(int[] arg, ArrayList list) {
        ArrayList line = (ArrayList) list.get(arg[2]);
        int[] beg = (int[]) line.get(0);
        int[] end = (int[]) line.get(line.size()-1);
        if(beg[0] == arg[0] && beg[1] == arg[1]) {
                for(int i=0;i<line.size();i++) {
                        beg = (int[]) line.get(i);
                        //System.out.print(beg[0] + "," + beg[1] + " ");
                        moveTo(beg[0],beg[1]);
                }
                return beg;
        }
        else if(end[0] == arg[0] && end[1] == arg[1]) {
                for(int i=line.size()-1;i>=0;i--) {
                        beg = (int[]) line.get(i);
                        //System.out.print(beg[0] + "," + beg[1] + " ");
                        moveTo(beg[0],beg[1]);
                }
                return beg;
        }
        else {
                System.out.println("Vector Drawing error!");
                System.exit(0);
        }
        return null;
}
public static void generateVectorPath() {
        int[] vector;
        int inc = 0;
        //drawBorder();
        System.out.println("");
        int counter = 1;
        while(vectors.size() > 0) {
                inc++;
                vector = getVector();
                moveTo(vector[0],vector[1]);
                System.out.print("v" + vector[0] + "," + vector[1] + " ");
```

```
                counter++;
        }
}
public static void transmitBitPath() {
        int counter = 0, counter2=0, temp, num;
        BufferedWriter w = null;
        FileWriter f = null;
        int size = steps.size();
        //double step = 1 / lines.size();
        //double stepCounter = 0;
        if(sendToCOM == 0) {
                try {
                        f = new FileWriter("output.txt",false);
                        w = new BufferedWriter(f);
                } catch(Exception e) { e.printStackTrace(); }
        }
        while(steps.size() > 0) {
                num = 0;
                for(int i=0;i<4;i++) {
                        if(steps.size() != 0) {
                                temp = ((Integer)steps.get(0)).intValue();
                                steps.remove(0);
                        }
                        else temp = 0;
                        temp = temp << 6-i*2;
                        num += temp;
                        counter++;
                }
                if(sendToCOM == 1) {
                        sendInt(num);
                        counter2++;
                        if(counter2 % 60 == 0) {
                                System.out.println("C=" + counter + " S=" + size);

                                serial.waitForData();
                                while(serial.available() > 0) serial.readByte();
                                //stepCounter += step;
                                //printCounter(stepCounter);
                        }
                }
                else {
                        try {
                                w.write(num + " ");
                                //printCounter(counter, size);
                        } catch (IOException e) {
                                e.printStackTrace();
                        }
                }
        }
        if(sendToCOM == 0) {
                try {
                        w.flush();
                        f.close();
                        w.close();
                } catch (IOException e) {
                        e.printStackTrace();
```

```
                }
            }
        }
        public static void printCounter(double f) {
            System.out.print("\b\b\b\b\b\b\b" + df.format(f) + "%");
            //System.out.print("\b\b\b\b\b\b\b" + df.format(percent) + "%");
        }
        public static boolean available(int x, int y) { return ((written[x][y] & gray[x][y]) != 0); }
        public static void disable(int x, int y) { written[x][y] = 0; }
        public static double distance(int x, int y, int newx, int newy) { return
Math.sqrt(Math.pow(newx-x, 2)+Math.pow(newy-y, 2)); }
        public static void runWait(String s) {
            Process p = null;
            try {
                p = Runtime.getRuntime().exec(s);
                p.waitFor();
            } catch(Exception e) { e.printStackTrace(); }


        }
        public static void main(String[] args) {
            if(args.length != 5) {
                System.out.println("Syntax: java EtchAStep <input ppm> <send to COM>
<convolve> <COM port H> <threshold>");
                System.exit(0);
            }
            System.out.println("Conversion: Bit Depth=8, Colors=3, Format=PPM
Resize=560X374 Resample=TRUE");
            runWait("i_view32.exe " + args[0] + " /resize=(560,374) /resample /bpp=8
/convert=input.ppm");
            openFile("input.ppm");
            int comPort = Integer.parseInt(args[3]);
            int convolve = Integer.parseInt(args[2]);
            sendToCOM = Integer.parseInt(args[1]);
            threshold = Integer.parseInt(args[4]);
            System.out.println("Processing: GRAYSCALE");
            grayScale();
            initialize(comPort);
            if(convolve == 1) {
                //gray = convolve(gray,gaussian);
                //gray = convolve(gray,laplacian);
                //Canny canny = new Canny();
                EdgeDetector edge = new EdgeDetector();
                edge.setSourcePixels(gray);
                edge.setThreshold(threshold);
                System.out.println("Processing: Gaussian Smoothing/Canny Edge Filter");
                try {
                    edge.process();
                } catch(EdgeDetectorException e) { e.printStackTrace(); }
                gray = edge.getEdgePixels();
                negateAbsolute(gray);
                //maxContrast();
                //testBW();
            }
            writeFile(gray, maxValue, "inputout.pgm");
```

```
                System.out.println("Conversion: Bit Depth=1, Colors=1, Format=BMP
Resize=560X374 Resample=TRUE");
                runWait("i_view32.exe inputout.pgm /resize=(560,374) /resample /bpp=1
/convert=input.bmp");
                System.out.println("Tracing: Vectorizing Bitmap Trace");
                runWait("rv2launcher.exe");
                System.out.println("Tracing: Generating 2-bit path");
                traceVectors();
                //vectors = new ArrayList();
                //sendInt(0);
                //sendInt(1);
                //sendInt(0);
                //addVector(-255,-255);
                System.out.println("Tracing: Complete, Polylines=" + lines.size() + " Nodes=" +
steps.size());
                //generateBitPath();
                System.out.println("Transmission: Transmit bit path");
                //printCounter(0);
                transmitBitPath();
                //sendInt(256);
                //sendByte((byte)(byte)240);
                //serial.waitForData();
                //while(serial.available() > 0) System.out.print(serial.readByte() + " ");
                System.out.println("\nSketch complete.");
                if(sendToCOM == 1) serial.close();
        }
}
```

input.ply

```
POLYLINE
403     16
406     20
404     18
END
POLYLINE
403     16
402     16
401     15
400     15
END
POLYLINE
403     16
408     5
END
POLYLINE
447     27
433     33
452     4
END
POLYLINE
447     27
425     52
408     62
387     65
375     59
```

```
372     46
367     36
368     36
379     28
392     12
398     4
END
```

SimpleSerial.java

```
import java.io.*;

/*
Interface class for SimpleSerial.

If you don't know what an interface is, don't worry.  An interface defines the functions
that can be called by implementors of this class.  Currently there are two implementors,
SimpleSerialNative, and SimpleSerialJava.  There might be more in the future.

SimpleSerialNative requires the file "SimpleSerialNative.dll" to be in the same folder as your Java
project file.  It's that simple.

SimpleSerialJava requires the correct installation of Sun's Javacomm communication package.
It's much
more powerful, but it can be tricky for the newcomer to configure, and use.

If you have problems with one, why don't you try the other.

A VERY SIMPLE IMPLEMENTATION:

public static void main(String args[]) {
   SimpleSerial       ss;              // delcare the SimpleSerial object

   ss = new SimpleSerialNative(2);       // The argument is the commport number.  There is no
Comm0.

   ss.writeByte((byte)'a');              // write a byte to the serial port

   // Give the PIC chip time to digest the byte to make sure it's ready for the next byte.
   try { Thread.sleep(1); } catch (InterruptedException e) {}

   ss.writeByte((byte)'!');              // write another byte to the serial port

   String     inputString = ss.readString();     // read any string from the serial port
   System.out.println("I read the string:  " + inputString);
}

A few important things to note:

1. When you write data to the serial port, it just writes the data, whether or not the PIC chip is
   ready to receive data.  There's no handshaking or signaling.  If you send data and the PIC chip
   isn't ready for it, it will be ignored.  Some PIC chips have a hardware UART which will buffer a
   few bytes of data for later retrieval.  But this isn't a cure-all.  If the buffer fills up, it
   creates an error condition which prevents further data from being read.  You need to include
   custom PIC code to reset the error flag.
```

2 In contrast to the PIC chip, the computer has a rather large hardware buffer. If the PIC chip sends serial data to your computer when you're not around to read it, it gets stored.

3. If you make a call to readByte(), and there's no data to be read, readByte() waits until there is data. If you want to know how much data (if any) is available, make a call to available().

4. Conversely, if you make a call to readBytes() or readString() and there's no data to be read, readBytes() returns a byte array of zero length, and readString() returns and empty string.

5. If you want to use Sun's JavaComm package instead of this code, subsitute your calls to
   new SimpleSerialNative(2);
   with
   new SImpleSerialJava(2);

*/

```
public interface SimpleSerial {

  // These are copied right out of WINBASE.H
  // Most applications should be fine with the defaults.

  public static final int NOPARITY            = 0;
  public static final int ODDPARITY           = 1;
  public static final int EVENPARITY          = 2;
  public static final int MARKPARITY          = 3;
  public static final int SPACEPARITY         = 4;

  public static final int ONESTOPBIT          = 0;
  public static final int ONE5STOPBITS        = 1;
  public static final int TWOSTOPBITS         = 2;

  /*
  Returns the number of bytes available at the time of this call.
  It's possible there are more bytes by the time readByte() or
  readBytes() is executed.  But there will never be fewer.
  */
  public int available();

  /*
  returns TRUE if port is fully operationsal
  returns FALSE if there is a problem with the port
  */
  public boolean isValid();

  /*
  Be sure to close your serial port when done.  Note that port is
  automatically closed on exit if you don't close it yourself
  */
  public void close();

  /*
  Reads a single byte from the input stream.
  Return value will be from -128 to 127 (inclusive)
  If no data at serial port, waits in routine for data to arrive.
  Use available() to check how much data is available.
```

```
    If error, returns 256.
    */
    public int readByte();

    /*
    Reads all the bytes in the serial port.
    If no bytes availble, returns array with zero elements.
    Never waits for data to arrive
    */
    public byte[] readBytes();

    /*
    Reads bytes from serial port and converts to a text string.
    DO NOT use this routine to read data.  Char->Byte converstion
    does strange things when the values are negative.  For non-
    text values, use readBytes() above
    */
    public String readString();

    /*
    Writes a single byte to the serial port.
    This writes the data, whether the PIC is ready to receive or not.
    Be careful not to overwhelm the PIC chip with data.
    On pics without a hardware UART, the data will be ignored.
    On pics with a hardware UART, overflowing will loose data AND require
    the UART on the PIC to be reset.  You can reset the UART in PIC code,
    or manually turn the PIC off and then on.

    NOTE:  A byte has a value in the range of -128 to 127
    NOTE:  If you want to write a character, you need to cast it to a byte,
           for example:  simpleSerial.writeByte((char)'b');
    */
    public boolean writeByte(byte val);

    /*
    For more advanced use.  Gets the input stream associated with serial port
    */
    public InputStream getInputStream();

    /*
    For more advanced use.  Gets the output stream associated with serial port
    */
    public OutputStream getOutputStream();

}
```

SimpleSerialNative.java

```
import java.io.*;
import java.util.*;

/*

Created on January 2, 2000
Modified on Januray 5, 2001 to fix compatibility with negative numbers.
```

Modified on January 11, 2001 with better error reporting
Ben Resner
benres@media.mit.edu

Very simple class for implementing serial port communcation on Windows platforms. It's intended for use
with PIC chips, but it should be useful for a variety of serial communications.

To use, you MUST have the file "SimpleSerialNative.dll" located somewhere in your dll search path. This means
it's in your "Windows\System32" directory, or in the same directory as this java file. If this isn't done, you'll get an unsatisfied link error or some other cryptic error.

Read the comment in SimpleSerial.java for some sample code.

NOTE: The term "Native" means the code has been written in something other than Java (such as C++) and compiled
into platform-specific code. Therefore, by definition, native Windows code won't work on Macintosh or UNIX
(without the use of some type of emulator).

*/

```
public class SimpleSerialNative implements SimpleSerial
{
   boolean     m_BeenWarned = false;

   // #### PUBLIC CONSTRUCTORS
   // ####

   // New a serial port.  Pass in comm port number
   SimpleSerialNative(int comPort) {
      _initPort( convertToCommString(comPort), 9600, 8, ONESTOPBIT, NOPARITY);
   }

   // New a serial port.  Similar to above, but allows greater user configuration
   SimpleSerialNative(int comPort, int baud, int dataBits, int stopBits, int parity) {
      _initPort( convertToCommString(comPort), baud, dataBits, stopBits, parity);
   }

   // #### PUBLIC MEMBER FUNCTIONS
   // ####

   // Checks to make sure serial port was initialized properly.
   public boolean isValid() {
      return (m_Port != 0);
   }

   /*
   Writes a byte to the serial port
   For example, to write the character 'b' to the serial port:  serialPort.writeByte('b');
   */
   public boolean writeByte(byte val) {
      try {
         m_DataOutputStream.writeByte(val);
      }
```

```
    catch (IOException e) {
      return false;
    }
    return true;
}

// Writes an entire string to the serial port
// For example, to write the string "hello world":  serialPort.writeString("Hello World");
public boolean writeString(String string) {
    try {
      m_DataOutputStream.writeBytes(string);
    }
    catch (IOException e) {
      return false;
    }
    return true;
}

// Waits here for data to arrive.
public void waitForData() {
    waitForData(1);
}

// Waits here for 'minNumBytes' of data to arrive.
public void waitForData(int minNumBytes) {
    try {
      while (available() < minNumBytes) {
        Thread.sleep(50);
      }
    }
    catch (InterruptedException e) {
      System.out.println("#### Thread interrupted -- could be big trouble");
    }
}

// read a byte of data.  If no data available, wait for data
// If an error occurs, returns -1.  It's not a bad idea to check for this.
public int readByte() {
    waitForData();
    try {
      return m_DataInputStream.readByte();
    }
    catch (IOException e) {
      return 256;
    }
}

// Read an entire string of data.  If no data available, returns an empty string.
// This routine never waits for data.
public byte[] readBytes() {
    try {
      int    len = available();
      if (len > 0) {
        byte   bytes[] = new byte[len];
        m_DataInputStream.read(bytes);
```

```
            return bytes;
         }
         else {
            return new byte[0];
         }
      }
      catch (IOException e) {
         return new byte[0];
      }
   }

   public String readString() {
      int    ii;

      byte data[] = readBytes();

      if (!m_BeenWarned) {
         for (ii = 0; ii < data.length; ii++) {
            if (!m_BeenWarned && data[ii] < 0) {
               m_BeenWarned = true;
               System.out.println("--> #### WARNING:  You are reading string data with values less
than zero.");
               System.out.println("--> #### This can be dangerous as Char->Byte remapping can
change negative values!");
               System.out.println("--> #### It's MUCH safer to use readBytes[] instead");
               System.out.println("--> #### You will only receive this warning ONCE");
               System.out.println("--> ####");
            }
         }
      }

      return new String(data);
   }

   // Returns how many bytes are available to be read.
   // Note that by the time you actually do the reading, more could be available.
   public int available() {
      try {
         return m_DataInputStream.available();
      }
      catch (IOException e) {
         return -1;
      }
   }

   // Close the serial port.  The port is automatically closed on exit, and you shouldn't normally
   // have to do this yourself.  It's only necessary if you want to close one port and re-open
   // another.
   public void close() {
      if (m_Port != 0) {
         _closeSerialPort(m_Port);
         m_Port = 0;
      }
   }

   /*
```

```
   The following two calls are for more advanced implementations.  For instance, if you need to
   peek at data without removing it from the queue, you'll want something like:

   serialPort = new SimpleSerialNative(2);
   DataInputStream dis = new DataInputStream(new
BufferedInputStream(serialPort.getInputStream()));

   // now you can do stuff like

   dis.mark(512);                  // mark this position
   String string = dis.readString();  // read a string (or anything)
   try {
      dis.reset();                 // set back to location where mark() was called
   }
   catch (IOException e) {}

   */

   // Gets the input stream for this port.  Allows greater control over IO
   // Most applications won't need these
   public InputStream getInputStream()
   {
      if (m_Port != 0) {
         return new SimpleSerialInputStream(this);
      }
      else {
         System.out.println("###ERROR:  You can't get input stream because serial port wasn't
opened");
      }
      return null;
   }

   // Gets the output stream for this port.  Allows greater control over IO.
   // Most applications won't need these
   public OutputStream getOutputStream()
   {
      if (m_Port != 0) {
         return new SimpleSerialOutputStream(this);
      }
      else {
         System.out.println("###ERROR:  You can't get input stream because serial port wasn't
opened");
      }
      return null;
   }

   // #### PUBLIC DATA MEMBERS
   // ####

   // You can use these directly if you seek more controll over IO.  Most applications
   // won't use these.

   public DataInputStream     m_DataInputStream;
   public DataOutputStream    m_DataOutputStream;

   /* ********************************** */
```

```
   // non-public members

   static
   {
      System.loadLibrary("simpleserialnative");
   }

   // Opens the serial port and returns the port handle.  The port handle is stored for use in the
_read/_write calls.
   native int _openSerialPort(String comPort, int baud, int dataBits, int stopBits, int parity);

   // Write the bytes in 'data'.  Returns the number of bytes written
   native int _writeSerialPort(int port, byte data[]);

   native int _writeSerialPortByte(int port, byte bit);

   // Reads all available bytes in the serial port buffer.  If nothing available, returns a zero-length
array.
   // Note this function never blocks -- it always returns immediately.
   native byte[] _readSerialPort(int port);

   native int _readSerialPortByte(int port);

   native void _closeSerialPort(int port);

   int                  m_Port = 0;
   Stack                  m_ReadQueue = new Stack();      // we're actually using this as a queue, not a
stack

   protected void finalize() throws Throwable {
      super.finalize();
      close();
   }

   void updateInputBuffer()
   {
      int       ii;
      byte      inputString[] = _readSerialPort(m_Port);

      for (ii = 0; ii < inputString.length; ii++) {

         int insertVal = inputString[ii];

         if (inputString[ii] < 0) {
            insertVal += 256;
         }
         m_ReadQueue.insertElementAt(new Integer(insertVal), 0);
      }
   }

   protected String convertToCommString (int comPort) {
      return new String("\\\\.\\com") + (new Integer(comPort)).toString();
   }

   private void _initPort(String comPort, int baud, int dataBits, int stopBits, int parity)
   {
```

```
        System.out.println("Initing NATIVE port.  Com = " + comPort + ", baud = " + baud);
        // We need to be sure serial port is closed on program exit.
        //System.runFinalizersOnExit(true);

        m_Port = _openSerialPort(comPort, baud, dataBits, stopBits, parity);

        m_DataInputStream = new DataInputStream(getInputStream());
        m_DataOutputStream = new DataOutputStream(getOutputStream());

        if (m_Port == 0) {
           System.out.println("###ERROR:  Couldn't open requested port ");
        }
     }
}

/* ********************************************************************** */
/* ********************************************************************** */
/* ********************************************************************** */
/* ********************************************************************** */


class SimpleSerialInputStream extends InputStream {
   SimpleSerialNative       m_Parent;

   SimpleSerialInputStream(SimpleSerialNative parent) {
      m_Parent = parent;
   }

   public int available() throws IOException {
      m_Parent.updateInputBuffer();
      return m_Parent.m_ReadQueue.size();
   }


   public int read() throws IOException {

      if (m_Parent == null || !m_Parent.isValid()) {
         return -1;
      }

      m_Parent.waitForData();
//       m_Parent.updateInputBuffer();       // this is done in waitForData()

      int  val = ((Integer)(m_Parent.m_ReadQueue.pop())).intValue();

      return val;
   }
}

class SimpleSerialOutputStream extends OutputStream {
   SimpleSerialNative       m_Parent;

   SimpleSerialOutputStream(SimpleSerialNative parent) {
      m_Parent = parent;
   }
```

```
    public void write(int val) throws IOException {
        int retVal = m_Parent._writeSerialPortByte(m_Parent.m_Port, (byte)val);

        if (retVal != 1) {
            System.out.println("### ERROR:  Unkown error writing to serial port.  If persists, probably
time to restart Windows!");
        }
    }
}
```