Electrical Circuit Creation on Android

Deqin David Xu, Michelle Hy, Simar Kalra, David Yan, Nasser Giacaman, Oliver Sinnen
Department of Electrical and Computer Engineering, University of Auckland, New Zealand
davidxu1991@gmail.com, mhy001@aucklanduni.ac.nz, simarkalra1993@gmail.com,
dyan040@aucklanduni.ac.nz, n.giacaman@auckland.ac.nz, o.sinnen@auckland.ac.nz

ABSTRACT

Most modern circuit design applications use a drag-and-drop approach to circuit creation. This can be both slow and unintuitive. This paper presents Voltique Designer, an Android-based circuit creation tool. The app seeks to combine the benefit of electronic circuit creation with the ease of hand drawing. Using RATA.SSR, Voltique Designer can recognize hand-drawn circuit components. The app analyses the circuit on the fly with a localized version of NGSPICE. User testing shows that Voltique Designer is quick to learn and easy to use compared to existing applications. An electronic logbook app called Montique is developed in conjunction with Voltique Designer.

Keywords: Sketch recognition, electrical circuits, android application

1. INTRODUCTION

Computer Aided Design (CAD) tools have become an essential part of the electronic design process.

Yet, the drag-and-drop approach to circuit drawing used in most applications is slow and unintuitive compared to drawing with pen and paper. This is especially true for inexperienced users.

Until now, hardware has been the limiting factor preventing fluent pen and paper style human-computer interaction.,

In recent years though, there has been a dramatic increase in the popularity of touch devices. This opens up the possibility for a circuit drawing application that accepts hand-drawn components.

This project seeks to combine the ease of pen-based input with the power of CAD simulation. The result is Voltique Designer, an Android based analogue electronic circuit design tool.

This paper will first describe Voltique Designer from a user's perspective. It will then explain some of the inner workings of Voltique Designer. Before concluding with some usability tests and future works.

2. RELATED WORK

2.1 Sketch Recognition

Pen/stylus based computing can be traced as far back as the Rand tablets (Davis & Ellis, 1964) from the 1960s. But not a lot of research has been done in this field until more recently, with the rise in popularity of touch based computing. The most popular applications of this kind are for artistic purposes. Examples include Adobe's Sketchbook Pro and Paper from FiftyThree. These applications are not concerned with what the user draws, only that marks appear when the user touches the screen.

Several note-taking applications are also allowing touch/pen-based input. For example, Evernote supports pseudo-shape recognition by smoothing out of drawn strokes. Xournals (Xournal, 2012) takes a geometric/heuristic approach. This means it only recognises individual features (e.g. edges, corners). It figures out what a shape is depending on how many of each features it have.

There are also commercial applications such as Notes+ (Notes, 2013) that supports shape recognition.

2.2 Circuit Design Tools

One of the earliest circuit drawing application is SketchPad (Sutherland, 1964). This system consists of a "light pen" and a set of push buttons and nobs for inputting commands. Example commands include "draw line", "circle centre", "move item" etc. The system was unable to do any circuit calculation.

In the 1970s came one of the first circuit simulation tool called MINIE (Spence & Apperley, 1977). The primary input method for MINIE was with light pens, but components were not drawn but chosen from a component menu by tapping.

Since then, the input method for circuit design tools have not changed. The only exception is that instead of tapping on a screen with a light pen. We now use the mouse as the primary input. LTSpice (Cosgrove, 1997) is perhaps one of the most well known example of modern circuit design tool.

With the rise of touch based computing, many mobile circuit design applications have been made. Example include, EveryCircuit (EveryCircuit, 2013), and iCircuit (iCircuit, 2013). While these applications are made for touch devices, they still use the same drag-and-drop approach like the desktop applications. And since they don't support keyboard input, instead of hotkeys, user need to click through a lot of menus to find what they need. This makes these applications even more difficult to use than the desktop applications.

CircuitBoard (Shane W. Zamora, 2009), is a JAVA applet is one of the closest to what this project is trying to develop. CircuitBoard recognises hand-drawn logic gates by breaking the strokes into lines and curves. Depending on the closeness of gaps, the program determines if a contour is formed. Due to the special property that all gate symbols being closed contours, if a closed contour is found, then it is assumed to be a symbol.

Christine Alvarado also developed a logic circuit diagram sketching tool, based on another application called SketchRead (Alvarado, 2007). Unlike CircuitBoard, Alcarado's application first classifies the strokes individually before symbol recognition.

This method could also be used for electric circuit recognition, but with only a success rate of about 60% (Alvarado, 2007)

Many of these earlier applications have aspects that are similar to the this project. But, some are too simple, like the shape recognition in note taking applications. Some are more

This paper appeared at ITX 2014, incorporating the 5th annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2014), the 27th Annual Conference of the National Advisory Committee on Computing Qualifications and CSANZ2014, Auckland, New Zealand, October 8-10, 2014. Mike Lopez and Michael Verhaart, (Eds).

of a research project for proof of concept with limited application in the real world. While others, in the case of EveryCircuit and iCircuit, uses the same drag-and-drop approach to circuit drawing as the desktop applications.

3. VOLTIQUE DESIGNER

3.1 User Interface

Voltique Designer's user interface can be seen in Figure 1. All the editing tools can be found on the toolbar at the top of the screen. To create a circuit diagram, simply draw on the canvas area.

After transient circuit analysis, a graph will open on the right hand of the screen showing the results. The colour of the graph line corresponds to the colour of the wires on the circuit. Multiple graphs can be opened so the user can compare results between different circuits.

Tapping the "logbook" button on the top left hand corner will open a dialogue where the user can take notes.

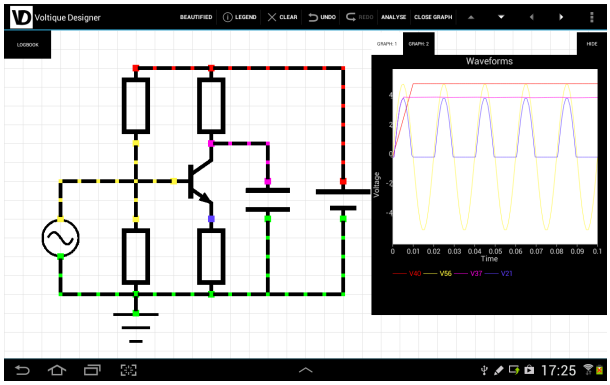

Figure 1: UI of Voltique Designer

3.2 Viewing Modes

Three different viewing modes are provided; Beautified View, Stroke Only View, and Both Beautified and Stroke View (Figure 2).

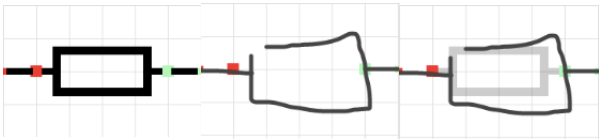

Figure 2: Different Viewing Modes (from left to right: beautified, stroke only, both)

In beautified view, as soon as a stroke is drawn, the hand-drawn stroke would be replaced by an image. This immediately tells the user what the drawn stroke has been recognised as. Some users may find the beautification of drawn stroke visually jarring. This is why Stroke Only View was also implemented; so the user can draw as if they are drawing on paper. In this mode, coloured squares will still appear at intersections on the grid that have been interpreted as ports just like they do in Beautified Viewing Mode. Alternatively, the user can choose to see both the beautified circuit and his/her hand-drawn strokes.

4. SKETCH INTERPRETATION

Voltique Designer uses RATA.SSR recogniser trainer to create custom trained sketch recogniser. The following stroke types are recognised: resistor, inductor, transistor, ground, straight wire, wire, deletion.

Once a stroke has been recognised, RATA returns a label indicating what the stroke is, and an array of coordinates representing the path of the stroke.

4.1 Single Stroke Component

For the interpretation of beautification of single stroke components, Voltique Designer uses the label and coordinates returned by the recogniser for the drawn stroke to place them on the canvas.

4.1.1 Step 1: Centre point and orientation

Based on the list of coordinates representing the path of the stroke, Voltique Designer can determine the centre of the drawn stroke, and the orientation of the component depending on the length and width of the stroke. For transistor and ground, by convention, no rotation is allowed.

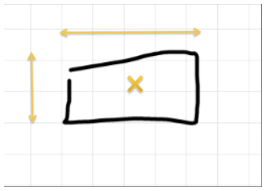

Figure 3

4.1.2 Step 2: Select "effective" centre point

Voltique Designer chooses the intersection closest to the "raw" centre point as the effective centre. This is the anchor point of the component. It makes sure every component is aligned against the background grid.

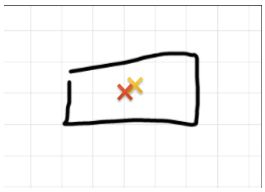

Figure 4

4.1.3 Step 3: Set occupied intersection

Depending on the component and the orientation of the component, occupied intersections are determined and set. This prevents multiple components from stacking

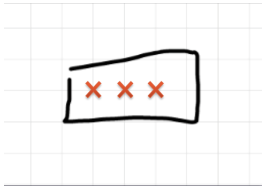

Figure 5

4.1.4 Step 4: Set the ports of the component

If two or more components' ports share an intersection, then Voltique Designer assumes they are connected. Intersections where ports are set are described as 'nodes'. Section 5 will go into more detail about how node setting works and why it's important.

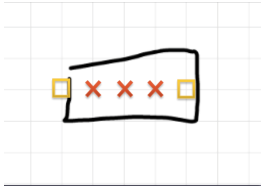

Figure 6

4.1.5 Step 5: Beautification

Once all the properties are set, an image of the component is displayed on the canvas with the effective centre point as the centre point of the image, while all the properties of the component are stored in a component list.

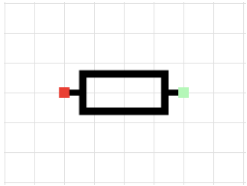

Figure 7

4.2 Multi-Stroke Component

There are two Multi-Stroke components available in Voltique Designer: Capacitor and Power Supply. RATA.Gesture recogniser does not support multi-stroke recognition. To recognise, interpret and beautify capacitors and power supplies, Voltique Designer must first take the following steps

4.2.1 Step 1: Check Previous Stroke

Both capacitor and power supply are drawn with two consecutive, short, straight wires. So when a new stroke has been drawn and RATA identifies it as a straight wire, Voltique Designer will check to see if the previous stroke drawn is also a straight wire.

4.2.2 Step 2: Check Relative Position

If both the previously stroke and the current stroke are straight wires, then check to see if they are approximately parallel, less than 80 pixels apart, and that their centre points are less than 40 pixels apart.

4.2.3 Step 3: check Wire Length

To be a capacitor or a power supply, both strokes must be less than 200 pixels long. Then if the ratio of the shorter wire is more than 75% the length of the longer wire, then the component is recognised as a capacitor. If the shorter wire is less than 75% the length of the longer wire, then the component is recognised as a power supply.

Once all the multi-stroke component has been recognised, its properties are set the same way as single stroke components; by finding the "raw" centre point, effective centre point, orientation, occupied intersection and port, except now, these are determined by the coordinates both of the current stroke and the previous stroke}

4.3 Wire

Unlike components, wires do not have predetermined shapes and forms. As a result, every coordinates of the stroke's path must be taken into consideration when interpreting wires.

To do this, Voltique Designer interpret and beautifies wires by first identifying various points of importance, before plotting from one point of importance to another depending on the plotting direction of the previous segment and the location of the next segment.

4.3.1 Step 1: Identify Points of Importance

There are four types of points of importance. (Figure 8)

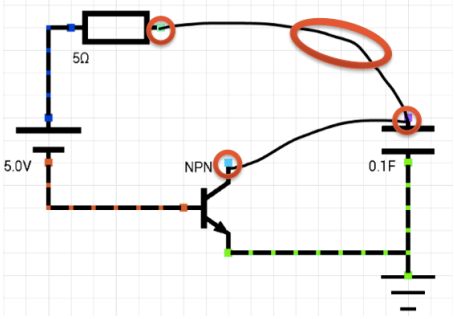

Figure 8: Different points of importance on a wire

1. Starting Point – This is usually the first coordinate of the stroke. However, in cases where the wire begins on an intersection occupied by a component, then the starting point is the first point that begins on either an empty intersection or the port of a component (i.e. node).
2. Turning Points – These are points in the drawn stroke where the stroke changes direction from moving approximately horizontally to approximately vertically or vice versa.
3. Intersection Points – These are points where the drawn stroke passes over a node.
4. Ending Points – This is usually the last coordinate of the stroke. However, like Starting Point, in cases where the wire ends on an occupied intersection, the last point that was not an occupied intersection is set as the ending point.

4.3.2 Step 2: Plotting segments from each point to the next

When plotting from one point to another point with the either the same X coordinate, or the same Y coordinate, a straight line is plotted. Every intersection between the two points has been set as the port of the wire. If one of the intersections is an occupied intersection then the plotting of the segment will fail, and in turn the wire beautification will fail.

On the other hand, when plotting to a point with both a different X coordinate and a different Y coordinate, there are two ways to plot the segment; plot in the X direction first, then the Y direction. Or plot in the Y direction first, then the X direction. See Figure 9.

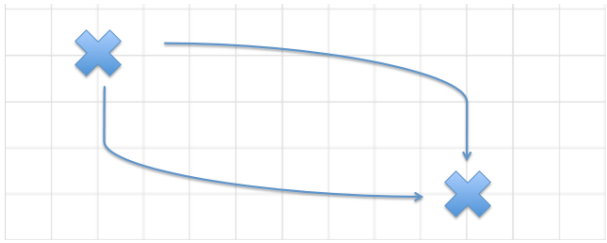

Figure 9

If the segment is the first segment in the wire, then Voltique Designer will first try to plot in the direction that the hand-drawn stroke begins in (for example horizontal direction). If it succeeds, then the segment will be stored. If there is an occupied intersection on the way, then the application will try to plot in the other direction first (for example vertical direction). If the plotting succeeds, then the segment will be stored, other wise the entire wire beautification will fail.

If the segment is NOT the first segment in the wire, then Voltique Designer will take into account the location of the previous point of importance, the location of the current point of importance, the plotting direction of the last segment, and the location of the point of importance that the app is trying to plot to.

For example, if the previous plotting direction was downward, and the vertical position of the next point is further up than the current point, then Voltique Designer will try to start by plotting going in the horizontal direction first. If this fails, then it will try to plot in the vertical direction.

On the other hand, if the vertical position of the next point is further down than the current point, then Voltique Designer will try to start by plotting downwards first.

4.3.3 Step 3: Beautification

Once all the segments are successfully plotted, straight line is drawn on the canvas over each segment.

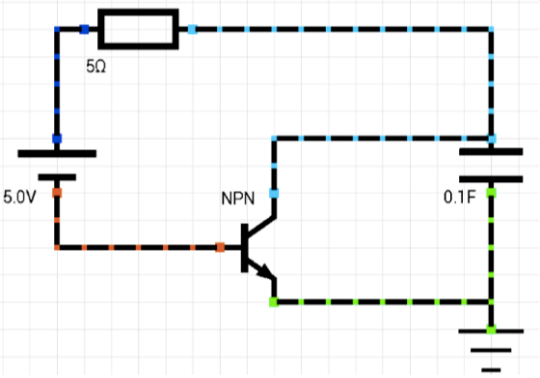

Figure 10: Wire after beautification

5. CIRCUIT ANALAYSIS

For circuit analysis, it is far more important to know which components are connected to each other than to know where these components are in a circuit. Instead of storing a list of components that each port of a component is connected to, Voltique Designer stores a single integer known as the Node ID for each port, and the app assumes that every port of any component with the same Node ID is connected together. All these ports are said to be the same node. A global Node ID counter is incremented every time a new node has been set.

5.1 Component Node Setting

When a component is drawn, its ports first checks what the Node ID of the intersection is. If the Node ID if zero, then it means that the intersection is empty. The component will set the intersection to the current count of the Global Node ID counter before incrementing the count. The component will then store that Node ID for that particular port.

If the Node ID is not zero, then the component will store the existing Node ID as the Node ID of that port.

One exception to this is Ground. The Node ID for ground is defined as "-1". If a ground is drawn, it will set the intersection to "-1" regardless if it's empty or not.

5.2 Wire Node Setting

When a wire is drawn, if it begins on an intersection with a Node ID of zero, then the wire will take the current Global Node ID Count as it's own Wire ID before incrementing the Global Node ID Count. Alternatively, if the wire begins on an existing node, then it will take the existing node's Node ID as the Wire ID.

If the wire comes across another node, then it will take on the new node's Node ID as it's Wire ID, unless the current Wire ID is -1, indicating ground, which cannot be overwritten.

Once Voltique Designer is sure that the wire drawing can be successful, and then all the nodes connected to it will be set to the Wire ID.

5.3 SPICE netlist generation and Analysis

For SPICE generation, Voltique Designer simply looks through the component list and reformats the stored properties. The generated netlist is then sent to a localized NGSPICE package for analysis. The simulation results are then returned to Voltique Designer in the form of a txt file. The results contain a list of voltages for each node over time.

For Steady State Analysis, Voltique Designer takes the value of each node at the end of the text file (i.e. After t becomes very big) and calculates the voltage across each component. Voltique Designer will then display the results next to the components with an arrow indicating the node with higher voltage potential.

For Transient Analysis, Voltique Designer takes the data points from the text file and plots them on a graph.

6. USABILITY

An informal user study was conducted with 23 undergraduate Electrical and Computer Engineering. The students were fluent in drawing electric circuits on paper. Most of them had some experience using desktop circuit drawing tools before. Each participant was asked to experiment with the application for approximately five minutes before they were asked to draw the three following circuits.

While these circuits are small, they contain all of the most used components. The assumption is that increase in time taken to draw bigger circuits will be linearly proportional to their increase in size.

6.1 Quantitative Evaluation

The three circuits participants were asked to try and draw are shown in Figure 11-13 As the participants drew, they were timed and the application also recorded the number of deletion, redo and undo done by the participant.

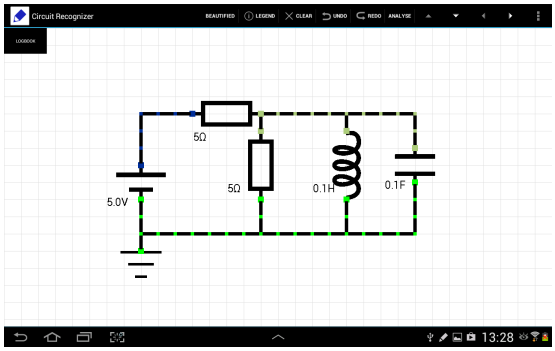

Figure 11: Test Circuit 1

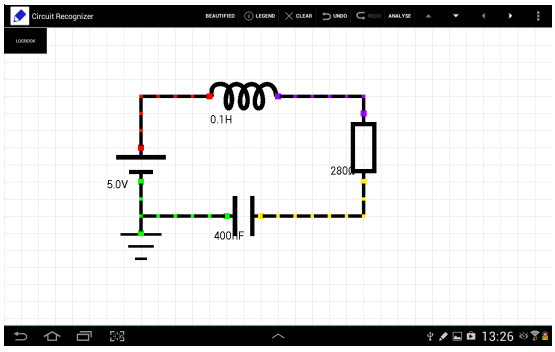

Figure 12: Test Circuit 2

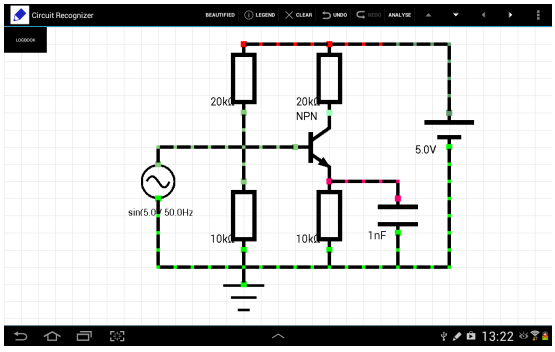

Figure 13: Test Circuit 3

6.1.1 Undo and Edition Rate

On average, the Circuit 1 took the participants 66.2 seconds to draw, Circuit 2 took 58.6 seconds to draw, and Circuit 3 took 194 seconds to draw. Figure 14 shows that though the median and mean of the first and second circuit are very similar, interdecile range for Circuit 2 has reduced by almost 50%. Since Circuit 1 and Circuit 2's are similar in complexity, the difference in range suggests that after having drawn one circuit, most people are able to navigate around the application with ease.

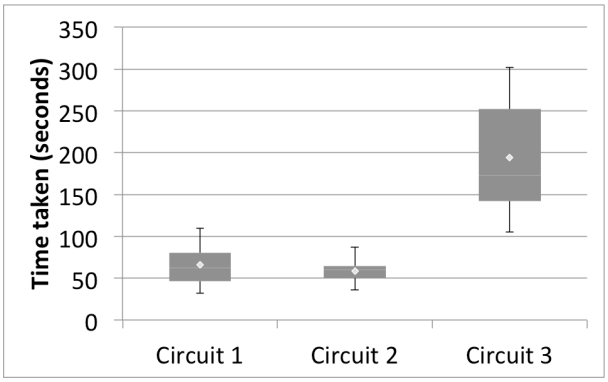

Figure 14: Graph of time taken for the participants to draw the three test circuits

The intuitive design of the UI is further demonstrated by the significant monotonic decrease in both average deletion count and average undo count as the participants moved from Circuit 1 to Circuit 2 (Figure 15). Interestingly, it can be seen that average undo count is not entirely dependent on the average deletion count.

Due to the increasing complexity of Circuit 3, there is a significant upsurge of average deletions. Despite this, the increase of average undo count is rather insignificant. Because circling gesture does deletion, while undo require tapping on a button. This again demonstrates how intuitive gesture based user interface are.

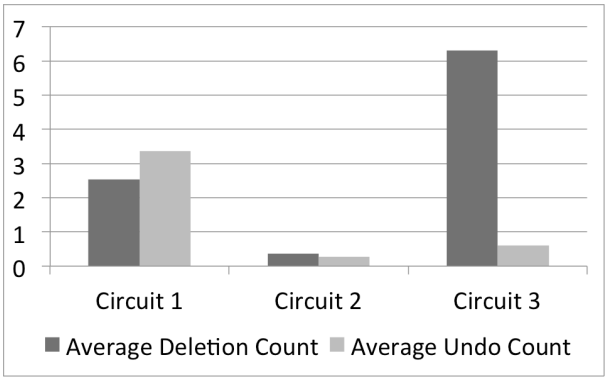

Figure 15: Average number of deletion and undo used for each circuit

Furthermore, we noticed that for Circuit 3, most participants were not making mistakes in terms of operating the software, but deleting and re-drawing the circuit due to running out of space as a result of poor planning.

This need to redraw for some participants but not others can accounts for the almost 200 seconds range for drawing time of Circuit 3.

6.1.2 Speed Comparison to Existing Applications

To see how Voltique designer compare to existing applications, a separate study, another group of 20 undergraduate Engineering Students were asked to recreate two circuits on both LTSpice, and Voltique Designer. LTSpice is an existing and popular tool for circuit design taught to all Electrical and Computer Engineering students at the University of Auckland. Participants were fluent in drawing circuit by hand. All participants had some experience using LTSpice and no participants have used Voltique Designer before. The simpler of the two circuits were similar in complexity to circuit 2 from the last study and the complex circuits were similar in complexity to circuit 3 from the last study. The participants were timed as they drew and analyzed the circuits. The results can be seen in Figure 16

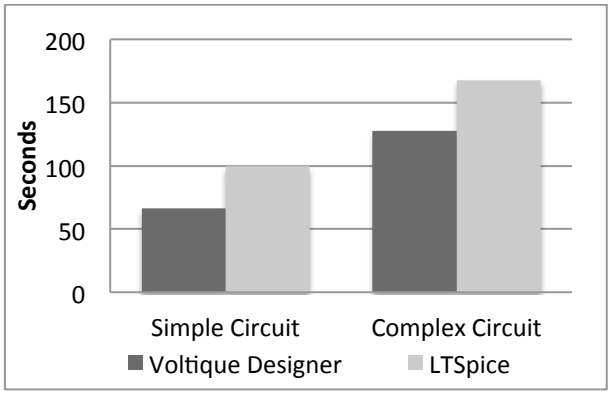

Figure 16: Average time for the participants to draw the two circuits using different platform

The timing result shows that on average, Voltique Designer was much more intuitive for drawing electric circuits than typical desktop based, drag and drop approach.

Additionally, we also noticed that with Voltique Designer, the participants spent a significant amount of time on inputting the value of the components, rather than drawing the actual circuit. The reason is, currently, to input a component value, the user need to press down on a component for a few seconds before a menu pops up. We learnt that this was not obvious to first time users.

At the same time, the fact that user spent significantly longer time trying to figure out how to input component value and still took much shorter time than they would on LTSpice is further evidence that the drawing process on Voltique Designer is indeed fast and easy to learn.

6.1.3 System Usability Scale (SUS)

After drawing the three circuits, the user was asked to fill out a System Usability Questionnaire (Brooke, 1996). It is a ten-item attitude Likert scale that measures the subjective perception of usability. Since it's conception, SUS has become an industry standard for usability evaluation. The results for Voltique Designer can be seen in Table 1.

Computing the results gives a SUS Score of 86.25, which according to Bangor (Bangor, Kortum, & Miller, 2009) is within range of "Excellent" usability, approaching "Best Imaginable".

Table 1: Built-in styles

Questions	Average Response
I think that I would like to use this system frequently	4.33
I found the system unnecessarily complex	1.25
I thought the system was easy to use.	4.58
I think that I would need the support of a technical person to be able to use this system	1.58
I found the various functions in this system	4.5

were well integrated	
I thought there was too much inconsistency in this system	1.67
I would imagine that most people would learn to use this system very quickly.	4.33
I found the system very cumbersome to use.	1.5
I felt very confident using the system	4.25
I needed to learn a lot of things before I could get going with this system	1.5

6.2 Qualitative Evaluation

Overall response to Voltique Designer was very positive. Participants especially praised the accuracy and responsiveness of sketch the recognition. Participants who have used desktop circuit drawing applications were especially impressed by how much easier Voltique Designer was to learn and use compared to other applications.

However, an issue raised by the participants was the application's inability to allow the user to draw components such as resistors and transistors in multiple strokes. Although most testers got used to drawing these components in one stroke after a little practice, it was noted as an inconvenience, especially if more complex circuit symbols are to be incorporated into the application in the future.

Comments were also made regarding the user interface. While the black and white colour scheme was generally regarded as aesthetically pleasing, the flatness of the design and the use of small fonts meant that it wasn't immediately clear that there were buttons on the top menu bar. Similarly, inside Legend, almost all participants had to be told that they could press on the component list to see a video tutorial for how to draw the component.

Some suggested that if the tutorial shown upon launch were more interactive would help the user to learn the controls much quicker.

Further more, the current method of moving the canvas around by tapping on the arrow button on the menu bar was seen as especially unintuitive. Participants suggested that for tablet based computing, it would be more natural if the canvas could be moved around by using two fingers to drag, or pinch to zoom.

7. FUTURE WORK

7.1 Electronic Logbook

Voltique designer already allow projects to be saved as XML files. The ability to share the saved files could prove to be invaluable to designers seeking collaboration.

An electronic logbook application called Montique is currently being developed. With it, the user will not only be able to save their projects, but they'll also be able to take notes, backup to the cloud with version control, and share their design with others instantly.

7.2 Multi-stroke sketching

Multi-stroke sketch recognition means user will really be able to draw on the tablet like how they draw on paper; with as many strokes, in whichever order as the user prefers.

This would also mean being able to produce more complicated components that are very similar simply by drawing, for example, NPN transistor and PNP transistor.

8. CONCLUSION

The drag-and-drop approach to circuit drawing used in traditional circuit drawing applications are generally perceived as slow and unintuitive. Circuit sketching by hand is much easier most of the time, but lacks the ability to quickly perform simulations and calculations. This project has successfully developed Voltique Designer, an Android application that combines the ease of circuit sketching by hand with the power of computer simulation.

By creating a custom single stroke sketch recogniser using RATA.Gesture, single stroke gestures were able to be interested as different components by the application. Storing a history of previous strokes allows the user to draw simple, multi-stroke components such as power supply and capacitor.

Using a grid-and-nodes method, components were interpreted and beautified and stored in a component list. Based on the components inside the component list, SPICE netlist can be generated to perform circuit simulation.

An informal study was conducted to evaluate the usability of the application. From the study, the response has been strongly positive both quantitatively and qualitatively.

9. ACKNOWLEDGEMENTS

Many thanks to Associate Professor Beryl Plimmer from Human Computer Interaction Lab for providing us with RATA recogniser.

10. REFERENCES

Adobe. (2013). Autodesk Sketchbook Pro Home Page.

Alvarado, C. (2007). *Sketch Recognition for Digital Circuit Design in the Classroom.* Paper presented at the 2007 Invited Workshop on Pen-Centric Computing Research.

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of usability studies, 4*(3), 114-123.

Brooke, J. (1996). SUS-A quick and dirty usability scale. Usability evaluation in industry, 189, 194.

Cosgrove, K. (1997). Analyzing Circuits with SPICE on Linux. *Linux Journal, 1997*(39es), 2-2.

Davis, M. R., & Ellis, T. O. (1964). The RAND tablet: a man-machine graphical communication device (pp. 325-331): ACM.

Evernote. (2013). Evernote Home Page.

EveryCircuit. (2013). EveryCircuit App Home Page.

Fiftythree. (2013). Homepage of Paper from Fiftythree.

iCircuit. (2013). iCircuit App Home Page.

Ngspice. (2013). NgSpice Online Home Page.

Notes. (2013). Notes Plus App Home Page.

Zamora, S. W., & Eyjólfsdóttir, E. A. (2009, February). Circuitboard: Sketch-based circuit design and analysis. In IUI Workshop on Sketch Recognition.

Spence, R., & Apperley, M. (1977). The interactive-graphic man-computer dialogue in computer-aided circuit design. *Circuits and Systems, IEEE Transactions on, 24*(2), 49-61.

Sutherland, I. E. (1964). Sketch pad a man-machine graphical communication system (pp. 6.329-326.346): ACM.

Xournal. Xournal Sourceforge Home Page (Vol. 2013).