Experiences in Using Open Source Software for Teaching Electronic Engineering CAD

Dr Simon Busbridge[1] & Dr Deshinder Singh Gill

School of Computing, Engineering and Mathematics, University of Brighton, Brighton BN2 4GJ

[1]s.c.busbridge@brighton.ac.uk

Abstract

Embedded systems and simulation distinguish modern professional electronic engineering from that learnt at school. First year undergraduates typically have little appreciation of engineering software capabilities and file handling beyond elementary word processing. This year we expedited blended teaching through the experiential based learning process via open source engineering software. Students engaged with the entire electronic engineering product creation process from inception, performance simulation, printed circuit board design, manufacture and assembly, to cabinet design and complete finished product.

Currently students learn software skills using a mixture of electronic and mechanical engineering software packages. Although these have professional capability they are not available off-campus and are sometimes surprisingly poor in simulating real world devices. In this paper we report use of LTspice, FreePCB and OpenSCAD for the learning and teaching of analogue electronics simulation and manufacture. Comparison of the software options, the type of tasks undertaken, examples of student assignments and outputs, and learning achieved are presented. Examples of assignment based learning, integration between the open source packages and difficulties encountered are discussed. Evaluation of student attitudes and responses to this method of learning and teaching are also discussed, and the educational advantages of using this approach compared to the use of commercial packages is highlighted.

Introduction

Most educational establishments use software for simulating or designing engineering. Most commercial packages come with an academic licence which restricts access to on-site computers. Even if off-site access is permitted is usually via a time-limited licence which means that students no longer have access to their work once this expires. The cost of non-academic licences is usually prohibitive for students and indeed many commercial companies are switching to open source alternative due to high licence costs [1, 2]. For these reasons we decided to consider switching to open source software in the teaching of electronic engineering CAD. Although there were inevitably some technical challenges in doing this we found several unexpected educational advantages.

Choice of Software Considered

Although there is a wide variety of open source tools available (mostly running on Linux) we wanted to keep the student computer interface as close as possible to that of their existing experience which meant using tools that would run under Microsoft Windows. We also wanted to keep the existing course objectives of taking students through the electronic product design and production cycle – from circuit design and simulation, PCB design and manufacture, and finally cabinet design, production and product installation – that was previously implemented using commercial packages. Thus we made the following software usage changes:

MultiSim (National Instruments) → LTspice (Linear Technology) *

Ultiboard (National Instruments) → FreePCB (GNU General Public Licence)

Solidworks (Dassault Systèmes) → OpenSCAD (GNU General Public Licence)

* strictly LTspice is freeware rather than open source software [3] however the practical implications from the educational user perspective are the same.

Students were also encouraged to think about using Open Office however the financial costs involved in purchasing Microsoft Office are not nearly so prohibitive as for less generic software. It is important to direct students to correct download site otherwise malware has a tendency to be bundled with the required software.

Software Experiences

1) Analogue Electronics Simulation Experiences with LTspice

LTspice [4] was chosen for a number of reasons (i) it is free for the end user, (ii) it supports schematic capture, (iii) there is copious online support via Yahoo! Groups [5], (iv) device manufacturers provide models that can be downloaded and incorporated into designs and simulations, (v) it is relatively easy to set up new components and devices, (vi) input and output files are simple text files that can easily be edited in any text editor, (vii) there is no limit to the complexity of circuits that can be simulated, subject to available processing power, (viii) it gives reliable simulation results that agree with experiment.

In order to use this type of software correctly students need a good understanding of basic computer file handling, such as directories and file extensions. Our computers are set up so that students can only write to the external USB drive or to their allocated server space. Students needed to understand how to create the necessary path structures. Similar issues occurred when trying to specify the directory of a stored model to be included in a simulation that was not in the main program library. Files can either be located in the same directory as the schematic (in which case no path needs to be specified) or in another perhaps more convenient directory (which must be specified). One issue that occurred with Windows 7 was its habit of concealing filename extensions unless set to display them. Consequently students added an additional false extension and wondered why the files did not behave as expected. Our university uses a virtual learning environment called "Student Central" [6] which is a convenient method for distributing simulation files to students. Unfortunately this has an irritating habit of adding a numerical digit to the filename which does not appear on the Student Central interface. In LTspice the undo and redo operations do not follow the normal Microsoft control-z and control-y keystrokes; instead the function keys are used. Similarly the conventional cut and paste shortcuts do not work. Strangely control-r does work for rotation. It is also necessary to be careful about assigning zero to component values because a singularity can be generated. Once these fundamentals and peculiarities had been understood the program worked extremely well. A rather nice feature is the appearance of voltage and current probes under mouse control once the simulation is running which are then placed on nodes or wires as required.

LTspice is not quite completely accessible via a graphical user interface. Whilst in schematic capture mode the drawing of circuit diagrams and the assignment of component values is undertaken through the GUI. However the inclusion of non-standard models and calling the appropriate analysis, as well as accessing more powerful operations, requires the writing of one or more *spice directives*. These are text commands entered anywhere on the schematic and must follow the prescribed syntax. Students soon realised that these commands, if at first rather tricky to master, gave LTspice a functionality far beyond their expectations.

Students were required to undertake simulation of basic DC circuits containing voltage and current generators, and a network of resistors. An interesting case was inclusion of a potentiometer which does not exist in the standard library; students were required to download the necessary files to include in their simulation. LT spice is quite capable of dealing with pulsed rather than repetitive signals and students were encouraged to explore this. Sinusoidal AC circuit theory was taught by the flip method [8, 9] with students undertaking simulation of reactive capacitor and inductive circuits before the theory was taught in lectures or real experimentation performed. Such is the convenience of freeware that they could return to the simulation whenever required to reflect on new material presented. Students undertook both frequency and transient analyses by writing the appropriate spice directive. This work inevitably led to filters and finally to the inclusion of non-linear device models and sub-circuits for diodes, transistors and operational amplifiers.

Fig. 1 shows a typical LTspice schematic with a corresponding voltage output plotted as a function of resistor value as seen on the screen (window layout, colours etc can be tailored according to personal taste). This was

obtained with a "print screen" request however LTspice has a rather nice "copy bitmap to clipboard" function that is very useful for creating reports. This circuit formed the basis of an assignment fairly early in the course in which students had to enter the correct spice directive to achieve the desired computation. Thus students were, without them realising it, learning about syntax, functions and arguments, loops and limits, i.e. some of the elementary constructs of computer programming. Of course this example is easily amenable to mathematical analysis by application of circuit laws and indeed some of the better students were able to validate their simulation results by doing precisely that.

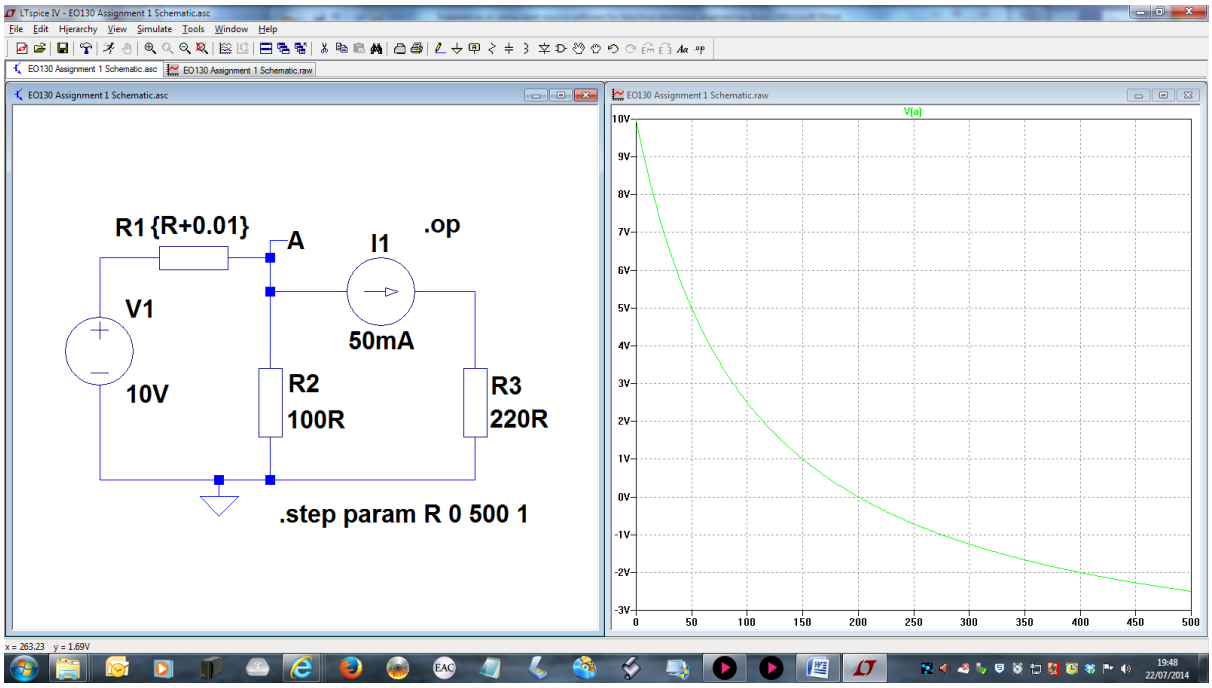

Fig. 1. Typical screen from LTspice showing schematic capture and analysis windows

Although LTspice can simulate digital circuits it requires an arcane "digital signal viewer" to trace outputs. More significantly its built in digital library is rather sparsely populated with devices and external standard spice libraries are not compatible. Future work might explore software such as Logisim [6].

2) PCB Design and Manufacture Experiences with FreePCB

Although open source, FreePCB [10] is as capable and as easy to use as many of its commercial counter parts. It was chosen because (i) it is free for the end user, (ii) it is operated by an intuitive GUI, (iii) it has a comprehensive footprint library including surface mount components, (iv) virtually all board parameters – pad, hole, track sizes etc. – can be adjusted, (v) it has an in-built footprint editor, (vi) online support is available, (vii) input and output files are text files, (viii) multi-layer boards are supported, (ix) gerber files are supported for CAM.

Students were required to lay out a PCB design for a small audio amplifier circuit based around the 741 op amp and two output transistors. The board was of a pre-determined size to fit into the cabinet designed in the next phase of the course. It was necessary to design and save several footprints as these were not in the library. Students had to pay attention to pitch, pad and hole size. The placing and routing resolutions should be set the same. The same issues with directories and paths occurred. Students had some difficulties with polarised components however their greatest challenges were understanding nets and avoiding track clashes requiring links. Issues such as setting insufficient grid resolution or changing resolution and not being able to snap to the correct coordinates were also apparent. Note that the Microsoft copy and paste shortcuts are observed for this program. It is necessary to remember net numbers or consult the netlist, which can be confusing. Students

were advised to increase the pad size and track width from the default to suit our routing machine. FreePCB makes extensive use of function keys (fortunately it indicates their function) which is very useful because whilst one had can operate the mouse the other is free to press keys.

Fig. 2 shows a screenshot of FreePCB with a partially completed board for this phase of the course. Note the mounting holes in readiness for the cabinet design.

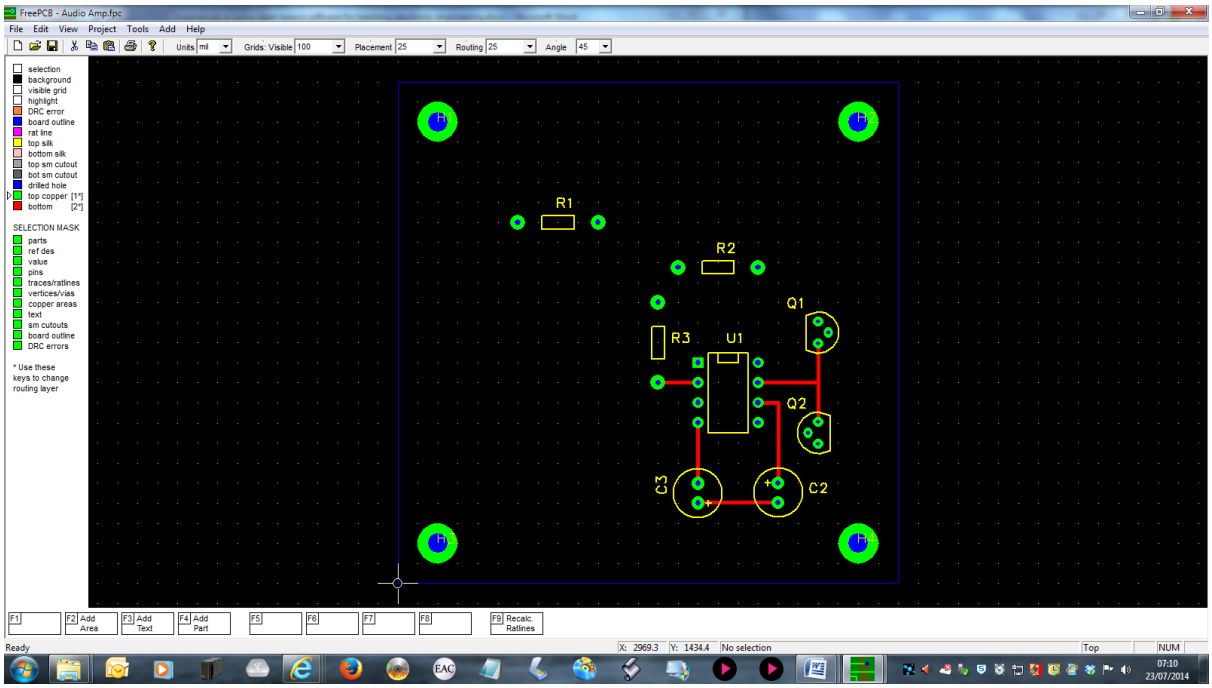

Fig. 2. Typical screen from FreePCB showing placement and routing in progress

Some students discovered that if they had set the number of layers to 1 they could only route tracks on the top side of the board. Some students proceeded to complete the layout without realising this. Fortunately it is a relatively easy matter to export their project as a text file, edit it offline with e.g. Notepad and then reimport it. The layers could then easily be changed. Finally students were asked to manufacture their PCB by outputting gerber CAM files to be submitted to our PCB router (which is a dry milling machine). They could then solder the parts in place and get their circuit to work.

Students were strongly advised to check the gerber CAM files before fabricating to ensure that everything was in order. We used gerbv [11] (released under the gEDA licence) for this task. This program (not to be confused with "gerbview") is free, open source and intuitive to use. It allows each gerber file to be loaded individually for checking. Multiple files can be loaded simultaneously to produce an overall view.

An interesting assignment was asking the students to find out how to export the netlist from LTspice, edit it to contain footprint information, and then import it into FreePCB so that the parts were already connected with nets which could be "rubber banded" by moving components. This is the essence of modern circuit design; there is little point in having a good simulation only to introduce further human generated errors by requiring the layout connections to be done manually. That it is possible to do this with open source tools is quite remarkable.

Students found this quite challenging. Those who researched it found an excellent resource online [12]. The issues included ensuring the correct format of netlist between the two programs (there are several standards for netlist formats) but by far the biggest difficulty encountered was pin assignment. With unpolarised components this is of little importance, but it is essential to have consistency of pin assignment between LTspice and

FreePCB for electrolytic capacitors, transistors, integrated circuits etc. Unfortunately the default positions are inconsistent! The best way to deal with this is by a pin swapping routine written into the device model so that they can be parsed between the two programs. This ensures that LTspice uses the correct physical pin allocations and not arbitrary ones. As this is rather subtle only the better students managed to comprehend this. Most students merely imported the netlist as-is and then physically reassigned nets to pins within the PCB package. Although this showed an understanding that pin assignment was important it rather defeated the object of automating the process as it introduced an extra stage of potential human error.

3) Cabinet Design and Manufacture Experiences with OpenSCAD

The inclusion of a mechanical element in the course is slightly beyond the scope of the learning of most electronic engineering students, although product design students are quite adept at this. It was felt that the former group should experience the complete electronic engineering design process, from schematic, analysis, performance simulation, PCB manufacture and finally enclosure design. It was clear that students had never before thought about how controls, indicators and transducers should be mounted, how to apply power to a completed product and the implications for connectors. They had not thought about which part of the enclosure should be internally accessible, e.g. for battery access, how PCBs should be mounted or indeed what shape and colour should be selected. Students were therefore asked to think about the final customer user and what demographic their product would be aimed at. Most students had never done this before. As well as a software design exercise students were asked to 3D rapid prototype their result and then assemble their electronics inside it. Students who had hobby experience with electronics produced interesting and innovative products whereas those new to it tended to stick with cuboid structures.

Usually physical components are designed with a 3D graphical design package such as SolidWorks or Autocad. This is perfectly compatible with product design or mechanical engineering students but is a little out of place for electronic engineering students. Although there are equivalent open source alternatives we decided to try OpenSCAD [13] because we thought it might help reinforce ideas in mathematics and geometry. OpenSCAD is a script based renderer based on constructive solid geometry. Users must describe the article to be rendered by reference to a coordinate system and by using vector based operations such as translate and rotate. Boolean operations can form more complex structures from simple shapes. The renderer allows 3D rotation and zooming for full a perspective view. The script can be saved, reloaded and even edited offline as a simple text file. OpenSCAD will product standard .stl files recognised by most rapid prototyping 3D printers. A typical screen shot is shown in fig. 3.

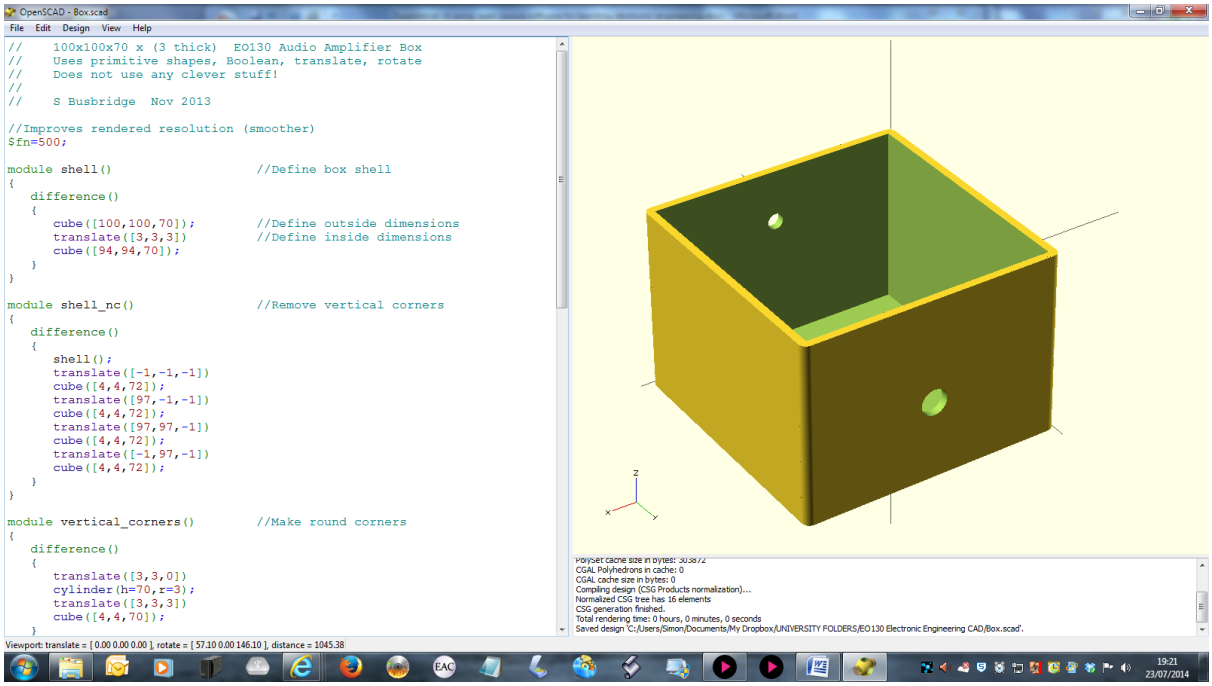

Fig. 3. Typical screen from OpenSCAD showing script and render windows

The script is essentially a mathematical description of the object to be rendered. The approach taken was to manipulate primitive shapes line by line, although the program is capable of much more complex structures such as loops and minkowski operations. As this was the first time that most students had encountered the need to describe objects and operations in code the simpler approach was used. In spite of this many students found 3D constructive vector geometry concepts challenging. The amount of detailed support provided for this phase of the course was deliberately reduced so that the flip method of teaching was more apparent. Clues how to build specific objects by using showing carefully selected operations forced students to use the online manual and support. Most students rose to this challenge very well, although a few disliked it and wanted to be told step by step. There were additional constraints to consider for 3D rapid prototyping, such as minimum wall thickness. Furthermore students had to carefully consider the overall size and mounting options for their electronics, as well as the external appearance.

Fig. 4 shows examples of students work [14]. The example on the left was the best of the group, showing a non-cuboid container with all electronics correctly mounted and fully operational. The middle example is ambitious but was poorly finished and the electronics mounting poorly executed. However the script written was impressive. The example on the right shows a case where there student had not properly considered the dimensions. The object was supposed to have a circular cross-section but it turned out to be elliptical.

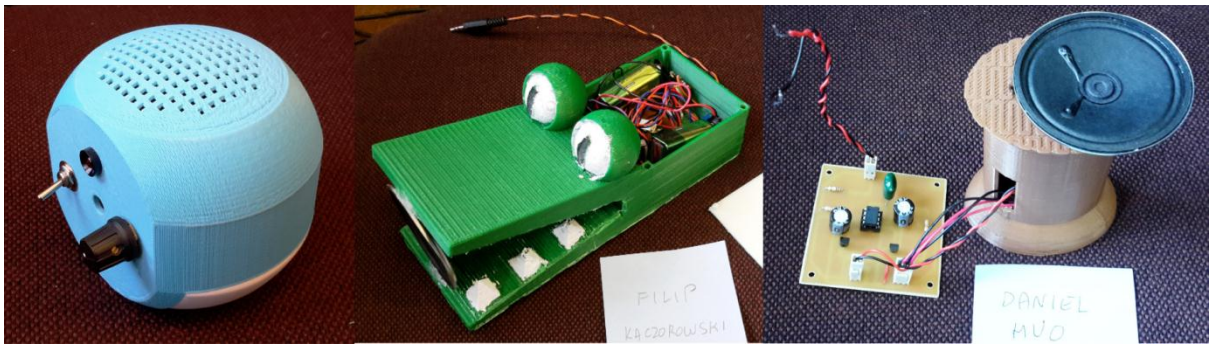

Fig. 4. A selection of student projects created by FreePCB and OpenSCAD open source software

Options for the future include use of a freeware vector-based 2D drawing package such as DraftSight (Dassault Systèmes) to more easily create and then import more complex shapes into OpenSCAD (which can be done as .dfx files)

Student Responses

Student responses were anonymously solicited by means of a questionnaire designed and accessed via SurveyMonkey [15]. Feedback was achieved from 8 students which represents approximately one third of the class cohort. Students were asked to rank their views on a scale from 1 (unfavourable or difficult) to 9 (favourable or easy).

Fig. 5 shows the response to asking if students found it useful to have access to the software offsite on their own computers. 88% of the student who replied stated that they downloaded and install all three packages. The results clearly show a benefit of allowing students access to course resources at their convenience. We did not notice a commensurate reduction in attendance; instruction, guidance, worked examples and tutorial support were provided conventionally in class.

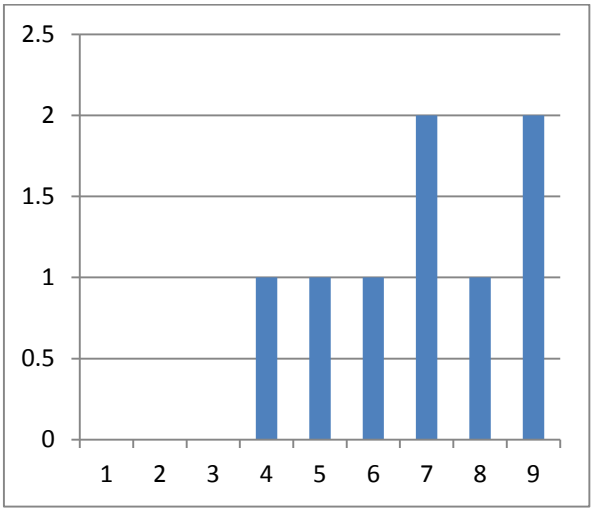

Fig. 5. Distribution of student responses stating they found access to software offsite helpful (9) or not (1)

We asked whether the students found LTspice difficult or easy to simulate analogue electronic circuits. The results are shown in fig. 6. Students adapted to using this software very well and were able to simulate a range of reasonably complex circuits. They told us that whilst the schematic capture worked very well, they found using the spice directive and analysing the results more challenging. The former requires text to be inserted according to the correct syntax which requires more thought than a drag and drop operation, even though its functionality is quite advanced.

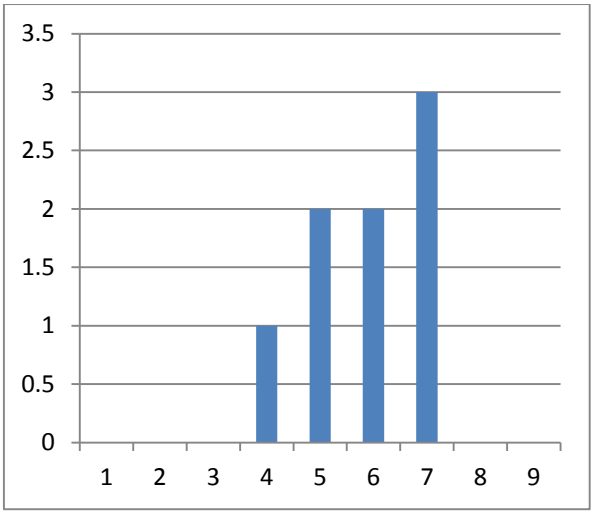

Fig. 6. Distribution of students who found simulating analogue circuits with LTspice difficult (1) or easy (9)

Students were asked if using LTspice improved their understanding of analogue electronics. The results are shown in fig. 7. Apart from a couple of mediocre replies the result is overwhelmingly positive. Of course this could have been achieved with *any* analogue electronics simulation package, but the point is that it was also achieved with an open source (freeware) package means that going open source does not detract from the educational objectives of the course.

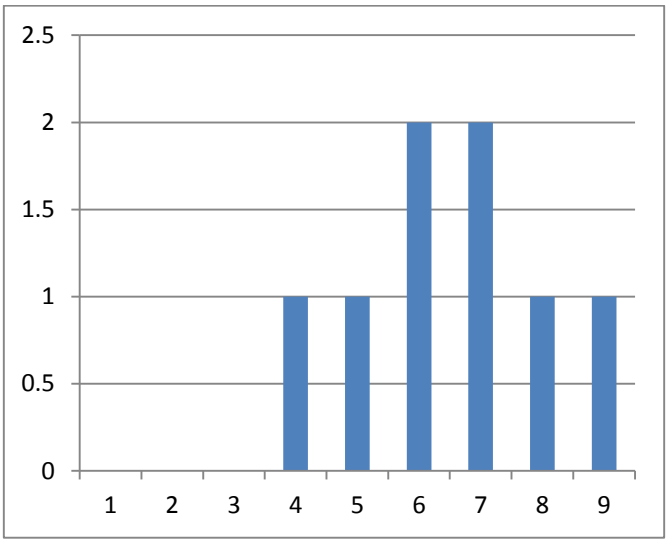

Fig. 7. Distribution of students who said using LT spice improved their understanding of electronics (9 is more)

Moving on to FreePCB, we asked the students how difficult or easy they found designing a board for their circuit and getting it manufactured. The results are shown in fig. 8. It is clear that students coped using this software very well (it is known that amateur electronics enthusiasts use it for hobby applications). This is unlike the results given in fig. 9 which show that students found the integration between LTspice and FreePCB much more difficult. Students particularly found the assignment of pin numbers to polarised components difficult to get right as this, and the assignment of footprints, required manually editing the netlist.

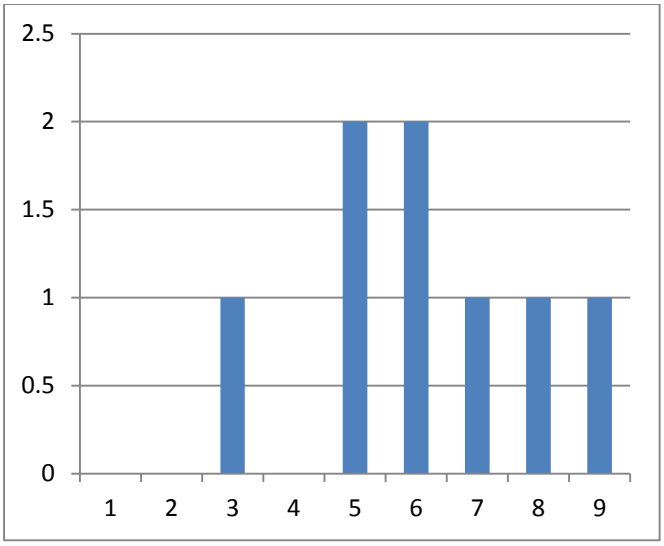

Fig. 8. Distribution of students who found designing PCBs with FreePCB difficult (1) or easy (9)

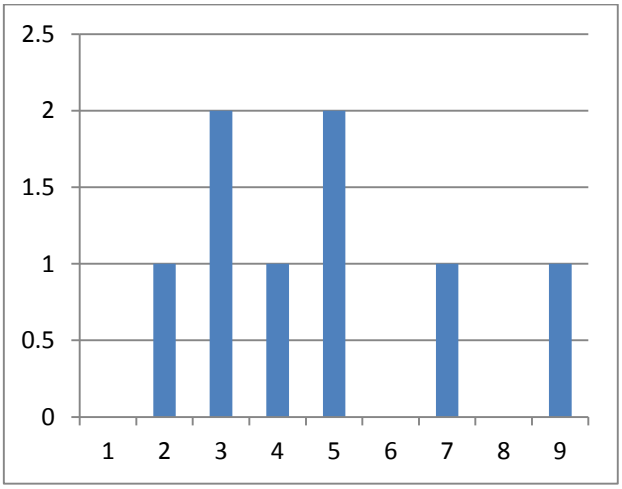

Fig. 9. Distribution of students who found the integration of LTspice with FreePCB difficult (1) or easy (9)

Whilst there are easy to use graphical interfaces to both LTspice and FreePCB, the same is not true of OpenSCAD. This software produces a rotatable 3D image rendered from a script which must be entered as text in the correct syntax in the accompanying box. This is rather more difficult, as it requires an extra level of understanding to convert from the desired physical object to its constructive solid geometric mathematical description. This is quite different to Solidworks, or indeed Google Sketchup, and the extra level of complexity is reflected in the feedback results.

We asked students if including an element of mechanical, or geometrical, design and its production contributed to their understanding of the entire electronic product design process. The results are shown in fig. 10.

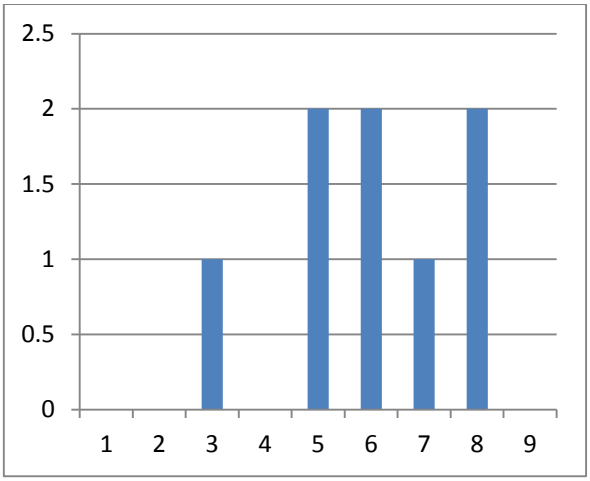

Fig. 10. Distribution of electronics students who found mechanical enclosure design worthwhile (9 is more)

It is clear there are mixed views on this, which is perhaps not surprising as most electronics students do not get to design cabinets (although interestingly hobby enthusiasts do). Despite these views all but one student managed to finish the assignment and produce a 3D printed result with only one disaster designed too small for the application.

We asked if the use of OpenSCAD improved their understanding of 3D vector geometry. The results are shown in fig. 11. The results are generally positive, although it is clear that some of the weaker students struggled with some of these concepts.

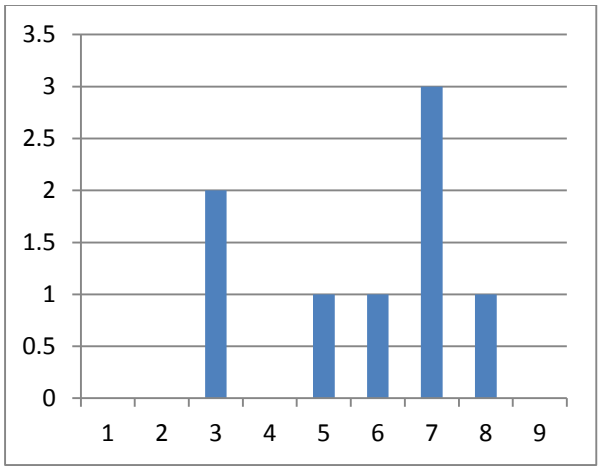

Fig. 11. Distribution of students who said using OpenSCAD improved their understanding of 3D vector geometry (9 is more)

Lastly we asked, on a scale of 1 (no) to 5 (yes) whether students enjoyed using these packages. The results are shown in fig. 12. It is clear that students really liked designing and making the PCB. They also liked doing analogue electronics simulations with LTspice, but they liked using OpenSCAD the least. Anecdotal feedback suggests that this is the script nature of the software selected and the associated challenge of using it, rather than the mechanical/product design nature of that part of the course.

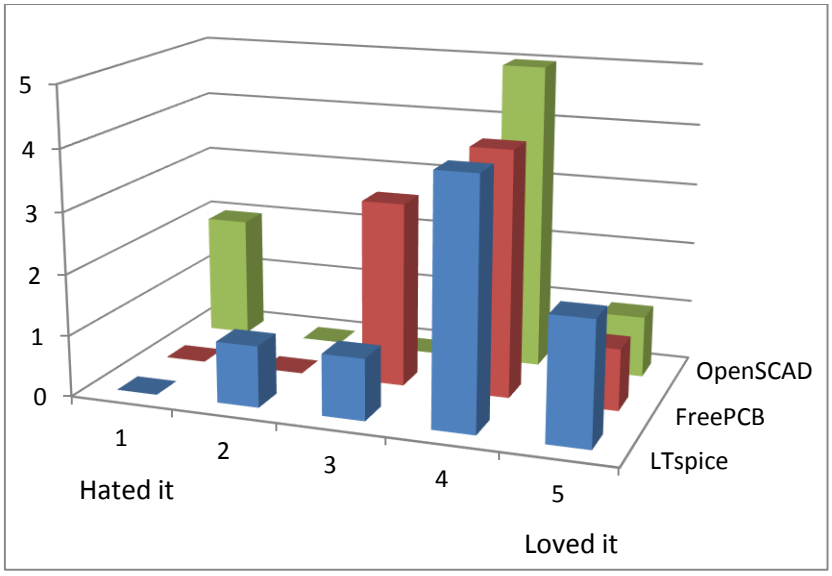

Fig. 12. We asked the student whether or not they enjoyed using each of the three packages

Table shows the average responses for each of the questions.

Was it helpful to have access to the software offsite?	6.9
Did you find LTspice easy or difficult to simulate analogue electronic circuits?	5.9
Did you find using LTspice improved your understanding of analogue electronics?	6.5
Did you find using FreePCB easy or difficult to design and manufacture a PCB?	6.1
Did you find the integration between LTspice and FreePCB easy or difficult?	4.8
Was the inclusion of a mechanical element helpful to understand the complete product design cycle?	6.0
Did you find using OpenSCAD improved your understanding of 3D vector geometry?	5.8
How enjoyable did you find using LTspice? (1 – 5)	3.9
How enjoyable did you find using FreePCB? (1 – 5)	3.8
How enjoyable did you find using OpenSCAD? (1 – 5)	3.4

Table 1. Average question responses (1 – 9 except the last three questions, higher represents more agreement)

Analysis with Bloom's Taxonomy

Bloom's taxonomy [16,17] provides a method of gaining insight into the mechanism and structure of the learning process. During the early stages of teaching students to use open source packages a certain amount of *knowledge* had to be conveyed. This included basic operational information, how to locate, install and run software, facilities required (e.g. use of a USB memory stick for local storage) and basic software operational procedures. It was clear that even at this stage students required a *comprehension* of file name extensions, how these are handled in Microsoft Windows, hidden file extensions, paths and directory structures (e.g. to specify the location of a file to be saved). In many cases students had not encountered the role of the operating system, even though they were adept in using computer and information technology as consumers. These concepts were valid for all the open source packages used however by the time students progressed from LTspice to FreePCB and then to OpenSCAD it was clear that they were *applying* the file handling knowledge learnt during the earlier weeks as this did not have to be re-taught.

Both the integration of netlists between LTspice and FreePCB, and the use of three dimensional vector geometry in OpenSCAD required students to *analyse* objectives in terms of the syntax required. For example, specifying shapes, their translation and rotation required development from the comprehension and application of a task to its analysis in terms of the underlying geometric operations. Similarly students had to think beyond merely generating netlist information learnt in class to an analysis of that information and its structure (e.g. pin assignments, component footprints, etc.). Once students had undertaken this analysis they were required to

synthesise a solution to ensure that their chosen schematic netlist and associated information would import directly into FreePCB and that the geometric shapes manipulated in OpenSCAD would 3D print to a viable solution.

Evaluation of achievement was undertaken by academic members of staff during the assignment marking process, although it was clear that some students had already thought about this phase of the learning process as they were asked to critically review their work. In some cases the evaluation was trivial (e.g. too small for the intended task) but others required more insight (for example how to change batteries in a portable electrical product, intended target market etc.). It was felt that student evaluation (as opposed to staff evaluation) could be better enhanced by the use of blogging techniques.

Conclusions

We have shown that it is possible to perform effective teaching and learning of electronic engineering CAD using open source software. Apart from being considerably cheaper for the host institution, students appreciated unrestricted access to their work at a time and place of their choosing. This provides ideal opportunities to use the flip technique. Without doubt the use of open source tools is not as user-friendly as their commercial counterparts. Students must have, or acquire, an elementary understanding of computer file handling.

This work has given an indication of what could be done in electronic engineering. There are many more opportunities, both within electronics and in general, some of which could be explored in future work.

References

1. "Open Source Software Gives Competitive Advantage Gartner Survey 729638", *eWeek*, QuinStreet Enterprise, New York, February 8, 2011
2. M. Wheatley, "The Myths of Open Source; It isn't all about cheap: Companies keep finding good reasons to take advantage of open-source software", *CIO*, March 2004, vol. 17, issue 10, pp. 1-
3. G. F. Murray, "Categorization of Open Source Licenses: More Than Just Semantics", *Computer and Internet Lawyer*, Jan 2009, vol. 26, issue 1, pp. 1-11
4. http://www.linear.com/designtools/software/ (last accessed 28 July 2014)
5. https://groups.yahoo.com/neo/groups/LTspice/info (last accessed 28 July 2014)
6. NewBay Media LLC, "Blackboard Learn", *Technology & Learning*, Jan. 2014, vol. 34, issue 6, pp. 42-.
7. http://logisim.en.uptodown.com/ (last accessed 28 July 2014)
8. A. J. Moore, M. R. Gillett and M. D. Steele, "Fostering Student Engagement with the Flip", *The Mathematics Teacher*, February 2014, vol. 107, no. 6, pp. 420-425.
9. J. Bergmann, "To Flip or not To Flip?, *Learning and Leading with Technology*, June 2012, vol. 39, issue 8, pp. 6-.
10. http://www.freepcb.com/ (last accessed 28 July 2014)
11. http://gerbv.geda-project.org/ (last accessed 18 September 2014)
12. http://andyc.diy-audio-engineering.org/ltspice_freepcb_1.html (last accessed 28 July 2014)
13. http://www.openscad.org/ (last accessed 28 July 2014)
14. Thanks to George Beattie, Filip Kaczorowski and Daniel Huo for access to photographs of their work.
15. http://www.surveymonkey.com (last accessed 28 July 2014)
16. Bloom, B., Englehart, M. Furst, E., Hill, W., & Krathwohl, D. (1956) *Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain*. New York, Toronto: Longmans, Green
17. Anderson, L. W., Krathwohl, D. R., et al. (2001) *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Allyn & Bacon. Boston, MA