**PCB Wizard**

**Designing Circuit Boards**  
To design a circuit board, simply drag and drop components onto your document and connect them together using the intelligent wiring tool. Then select the menu option 'Convert to PCB' and leave PCB Wizard 3 to do the rest for you.   
  
 If you want to simulate your design before turning it into a circuit board, PCB Wizard 3 offers tight integration with Control Studio 2 and Livewire.

**3.3 Proteus**

Proteus was initially created as a multiplatform ([DOS](https://en.wikipedia.org/wiki/DOS" \o "DOS),[Windows](https://en.wikipedia.org/wiki/Windows), [Unix](https://en.wikipedia.org/wiki/Unix)) system utility, to manipulate text and binary files and to create [CGI](https://en.wikipedia.org/wiki/Common_Gateway_Interface) scripts. The language was later focused on Windows, by adding hundreds of specialized functions for: network and serial communication, database interrogation, system service creation, console applications, keyboard emulation, [ISAPI](https://en.wikipedia.org/wiki/ISAPI) scripting (for [IIS](https://en.wikipedia.org/wiki/Internet_Information_Services)). Most of these additional functions are only available in the Windows flavour of the interpreter, even though a [Linux](https://en.wikipedia.org/wiki/Linux) version is still available.

Proteus was designed to be practical (easy to use, efficient, complete), readable and consistent.

Its strongest points are:

* powerful string manipulation;
* comprehensibility of Proteus scripts;
* availability of advanced data structures: [arrays](https://en.wikipedia.org/wiki/Array_data_structure), [queues](https://en.wikipedia.org/wiki/Queue_%28data_structure%29) (single or double), [stacks](https://en.wikipedia.org/wiki/Stack_%28data_structure%29), [bit maps](https://en.wikipedia.org/wiki/Bit_array), [sets](https://en.wikipedia.org/wiki/Set_%28computer_science%29), [AVL trees](https://en.wikipedia.org/wiki/AVL_tree).

The language can be extended by adding user functions written in Proteus or [DLLs](https://en.wikipedia.org/wiki/Dynamic-link_library) created in C/[C++](https://en.wikipedia.org/wiki/C%2B%2B).

**3.4. Embedded C**

Embedded C is a set of language extensions for the C Programming language by the C Standards committee to address commonality issues that exist between C extensions for different embedded systems. Historically, embedded C programming requires nonstandard extensions to the C language in order to support exotic features such as fixed-point arithmetic, multiple distinct memory banks, and basic I/O operations. Embedded C uses most of the syntax and semantics of standard C, e.g., main() function, variable definition, datatype declaration, conditional statements (if, switch, case), loops (while, for), functions, arrays and strings, structures and union, bit operations, macros, etc.

**Features:-**

1. It is small and simpler to learn, understand, program and debug.
2. Compared to assembly language, C code written is more reliable and scalable, more portable between different platforms.
3. C compilers are available for almost all embedded devices in use today, and there is a large pool of experienced C programmers.
4. C has advantage of processor-independence and is not specific to any particular microprocessor/microcontroller or any system.
5. As C combines functionality of assembly language and features of high level languages.
6. It is fairly efficient.
7. It supports access to I/O and provides ease of management of large embedded projects.