Creation of a ladder editor in html5 and JavaScript

by

Fredrik Bengtsson and Ivan Flink
Industrial Electrical Engineering and Automation
Lund University

Introduction

When working with automation processes, reprogramming the process can often be a long and arduous task, often requiring specialist help. The task that the company B&R Automation gave us was the creation of an editor that could be run from web browsers that allowed the user to create and edit programs for their automation processes. This editor, made for ladder programming, is easy to use and contains most of the functionality needed for ladder programming. Moreover it is simple to extend it to add more functions.

Ladder Programming Language

When programming machines and other automation processes there are a huge number of different programming languages available. One of the most common and widely used is the ladder programming language. This is the language our editor was designed to use. It is a graphical programming language that is simple to use and understand. In its simplest form it contains inputs, known as contacts as well as outputs known as coils. The contacts are all tied to boolean values (true or false), can be inputs from the sensors of a machine or they can be variables introduced by the programmers. The coils are also tied to boolean values. Once again these can both be outputs from a machine or variables created by the user.

Ladder programming is done by creating a circuit of these contacts, with coils placed at the end. If an unbroken chain of contacts with the value true reach a coil, that coils variable will be set to true.

So for instance in the case below, the following ladder diagram is presented:

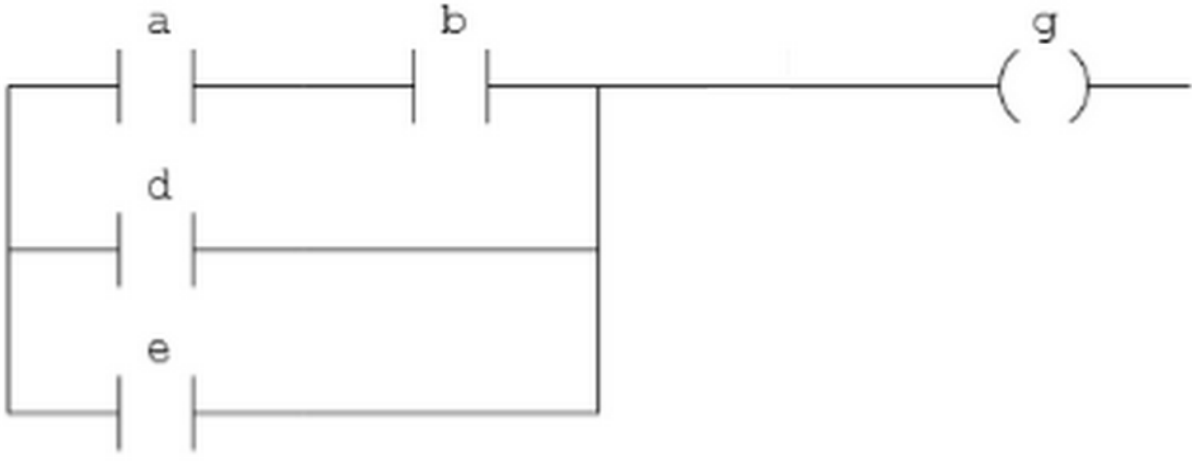

Figure 1: A simple ladder diagram.

In this diagram the output variable *g* will be true if any of the following combinations of variables are true (see figures 2 and 3):

- *a ,b*
- *d*
- *e*

If none of these are true *g* will be false, unless it is set to true by another ladder diagram. When the ladder program is run, it is run in a cyclic fashion, with the program in each cycle checking each of the contact values, and then updating the coils as needed. So in the case above, the program will check the values of *a, b, d* and *e*, and determine if *g* should be true, if so it is set to *true*. When the next cycle is started the process is repeated and once again all the variables are checked and outputs are set appropriately.

There are of course many more components in ladder logic such as inverted contacts (these function so that they will help set coils if they are false), function blocks allowing arithmetic functions and many others possible components. These enable the user to create much more complex programs then the one presented here

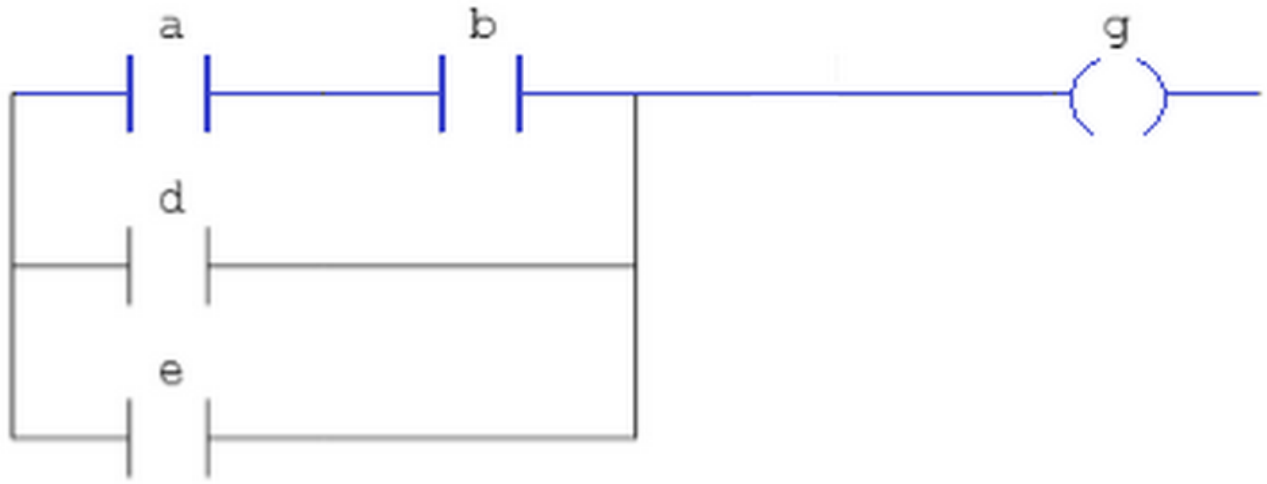

Figure 2: If a and b are true, g is true.

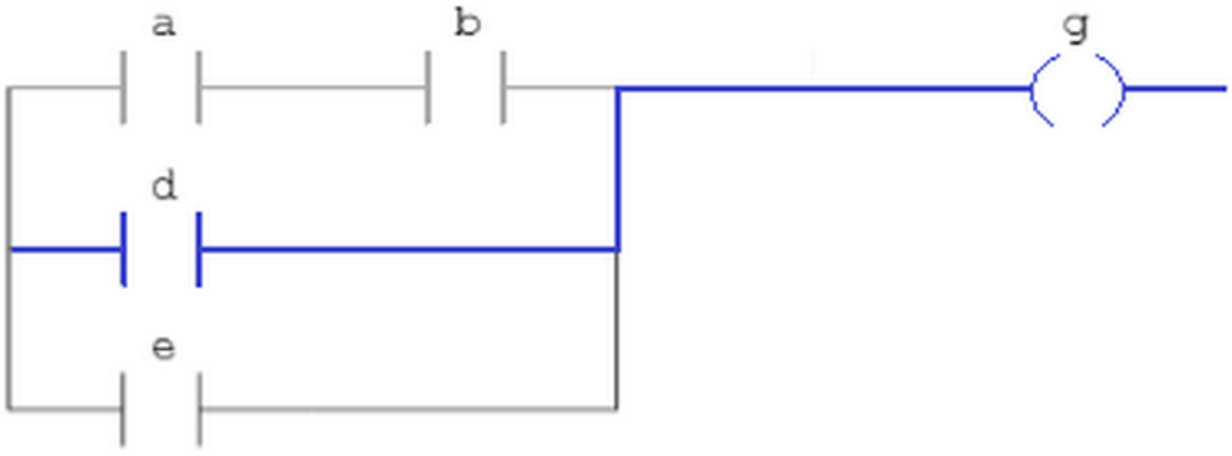

Figure 3: , If d is true, g is true.

The Editor

The task of our thesis project for B&R Automation was to create an editor for ladder programming in html5 and JavaScript. By using these languages, an editor would be created that could be run by most common web browsers.

The editor that was created differs greatly from most other editors. In most other editors the components (such as contacts and coils) are inserted, and then the user draws lines connecting them to the other components. In the version that is presented here, the user simply clicks where he or she wishes to insert the component, and the editor will automatically redraw the lines. Likewise if the user wishes to delete a component, he or she simply clicks it and then presses the delete button. The component will be removed and the lines will be redrawn to reflect this change. This means that the user can easily change an existing ladder program without it becoming messy and unclear, something many other ladder editors struggle with.

Another way this ladder editor differs from the others is how the ladder program is represented. The entire ladder program is represented as a single large JSON object. This is done because JSON objects have one large advantage, which is that they can easily be converted into a string, and this string can easily be converted back into a JSON object. This means that the entire ladder program the user has been working on can be converted into a line of text. The line of text can then be sent from the ladder editor. This allows the ladder program to move between the ladder editor and B&R's program Automation Studios. As Automation Studios has functionality that allows saving texts to file, this can be used to save and load the program. Moreover as Automation Studios can be connected to a machine, it allows the ladder program from the editor to be effectively run on this machine, provided that Automation Studios contains a working interpreter that can interpret the program.

The editor also contains functionality to copy, cut and paste components. This along with undo and redo buttons makes the editor easy to use and forgiving for mistakes by the user.

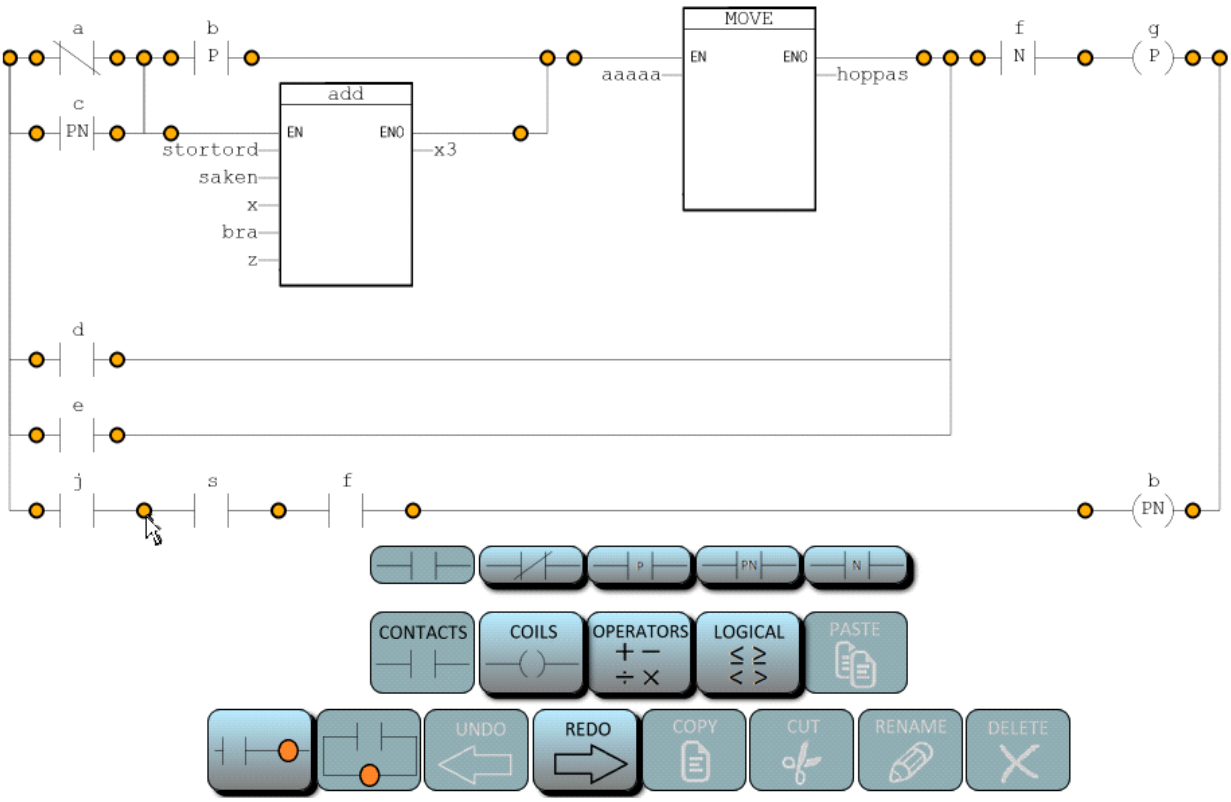

Figure 4: The user inserts components by clicking where he or she wishes to add them.

Conclusion

In conclusion a complete and easy to use editor for the ladder programming language was created. The editor implements all common ladder components, has functions such as copy, paste and undo and can easily be adapted to add more components. Communication with B&R's program Automation Studios is also implemented, which can be used to save and load the program.