A Simulation Software for Sequential Control

Pablo San Segundo, Diego Rodríguez-Losada
GSITAE research group
Universidad Politécnica de Madrid (UPM)
Madrid, Spain
pablo.sansegundo@upm.es

Finite state machines are a fundamental issue in a big number of fields in industry, such as sequential control, digital electronics, automated systems, automated reasoning etc. In our experience at the Universidad Politécnica de Madrid (UPM) we have found that the traditional approach of teaching the fundamentals of finite state machines through formal models is neither motivating nor allows to transmit the basic underlying principles to the student in a simple way.

Bearing this in mind we have developed a simulation software prototype in C++ programming language which uses the powerful 3D graphical libraries OpenGL and the portable Window Manager (GLUT) to develop simple 3D systems for sequential control which can be visually attractive to the student.

Keywords: Finite state machines, simulation, sequential control, PLC, physical systems.

I. INTRODUCTION

Finite state machines are a fundamental issue in a big number of fields in industry, such as sequential control, digital electronics, automated systems, and automated reasoning to name but a few.

The subject *Automatización Industrial* (Industrial Automatization [1]) is part of the current degree program of Industrial Engineering with specialization in Industrial Electronics, Automation and Sofware Engeneering imparted at the Escuela Universitaria de Ingeniería Técnica Industrial (EUITI) which belongs to the Universidad Politécnica de Madrid (UPM). This subject is taught during the fifth semester (third year, first semester) of the degree by the Department of Electrónica, Automática e Informática Industrial [2], to which the authors belong. The subject is worth 9 credits and has to be taken by all students of the programme, approximately 110 students each year.

The aim of the course is to be able to model and implement the control of industrial production systems using a PLC.
It is divided into theoretical lessons (5 hours per week) and practical ones (1 hour per week with a PLC). The subject is eminently practical and much of the time is spent in describing different technological components, communication networks, human machine interfaces, specific programming languages for PLCs etc. However our experience tells us that an extra effort should be made about the underlying model behind sequential control which is the finite state machine.

One approach to teach the basic underlying principles of finite state machines to the students at the beginning of the course is to employ simulation using as front-end SCADA (Supervisory Control and Data Acquisition) applications (as in [3]) together with a PLC simulation tool (as in [4]). This approach poses a number of problems. In the first place it becomes necessary to overcome the cost of software licenses which can be of several thousand of euros. In the second place, the level of abstraction is in many cases too close to the concrete system for an introductory course on finite state machines. Finally there is the drawback of using proprietary software as there are many practical examples in industry where finite sate machines need to be implemented in a general purpose framework.

Motivated by the above facts we have developed a simulation software prototype in C++ language which uses the powerful yet simple 3D graphical libraries OpenGL and the portable Window Manager (GLUT) to develop simple 3D systems which can be visually attractive for the student. Students are given a concrete control task and the access to the system control files. The tool allows testing the sequential control on the selected system to find out if the required specifications are met. Kinematics has also been included in the models so that erroneous control (i.e. a cylinder which exceeds a sensor limit) is also shown in the screen.

At the moment we dispose of four basic physical environments (garage door, elevator, cylinders and production chain). Each system requires a sequential control which can be implemented by a finite state machine. In each case the student needs to program the control in a plain text file written in C which associates sensor information with the corresponding control actions. After compiling the text file, the student can then watch the effect on the system. Additionally appropriate software engineering decouples the logical and graphical parts of the application so that it is relatively easy to add new elements to the graphical library which can then be employed in some or all systems.

We have used our prototype for the first time this year in the early part of the course Industrial Automatization both in theory (to illustrate the basic principles of sequential control) and in the laboratory (for physical system simulation). In the latter case we think that the prototype gives the students a better understanding of the system control requirements before they start programming the PLC. Additionally, we plan to use the tool in some of the SW courses (e.g. Industrial Informatics or Software fundamentals) to illustrate how to implement a finite state machine in a general purpose language.

The initial informal feedback of the students has convinced us of the effectiveness of our approach as the motivation of the students has become clearly visible. It is as yet too early in the year to correlate effort with understanding of finite state machines but there has clearly been an increase of hours of dedication as opposed to previous years. Additionally, personal opinions of some students have been very gratifying and encourage the authors to develop new 3D systems for future courses.

This paper is structured form here in five sections. Section 2 describes the current logical and graphical components in the application's repository and Section 3 the actual assignments which have been given to the students this year. Section 4 deals with implementation issues and includes an example of the language employed by the control function. Finally Section 5 presents conclusions and future work.

II. SYSTEM REPOSITORY

At the moment we have implemented 4 systems in the 3D repository: a lift, a manufacturing line, a garage door and a pneumatic cylinder working table. Kinematic and logical simulation is provided. For example, if a motor is not stopped when it reaches its limit, it will break and not work any more. User interaction is provided by buttons and edit boxes in the interface. We will describe each system in turn.

A. Lift

Figure 1 shows the interface of our 3D lift in the repository. The system has the following inputs:

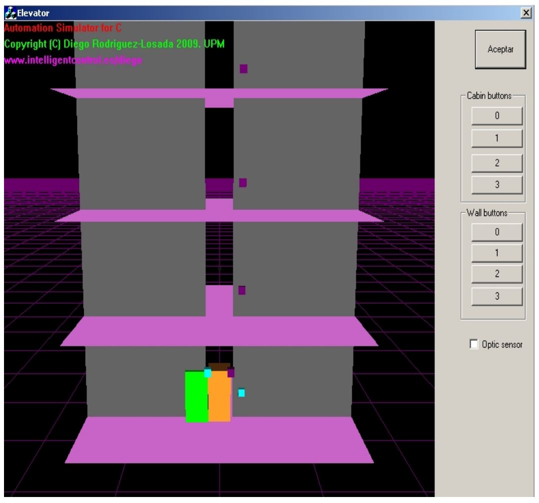

Elevator
Automation Simulator for C
Copyright (C) Diego Rodriguez-Losada 2009. UPM
www.intelligentcontrol.es/diego
Aceptar
Cabin buttons
0
1
2
3
Wall buttons
0
1
2
3
Optic sensor

Figure 1. Interface for the 3D lift

- Cabin buttons and outside buttons to indicate the floor chosen by a potential user.
- An optical sensor which detects that a person is entering the cabin (this makes the user appear on the screen).
- A sensor at every floor to indicate the position of the cabin. These sensors change color to blue in the 3D simulation when the cabin is detected.
- Sensors that detect whether the cabin door is open or closed

Control actions are reduced to moving the cabin upward and downwards as well as opening and closing the door.

B. Manufacturing line

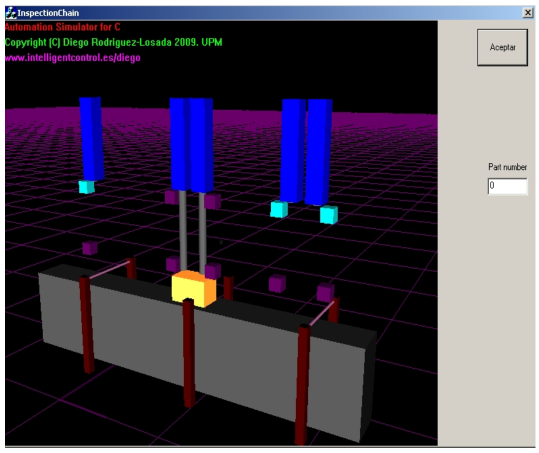

InspectionChain
Automation Simulator for C
Copyright (C) Diego Rodriguez-Losada 2009. UPM
www.intelligentcontrol.es/diego
Aceptar
Part number
0

Figure 2. Interface for the 3D manufacturing line

Figure 2 illustrates our 3D manufacturing line made up of 3 different stations. A product (or subproduct) moves along the chain and can be attacked in one or both of the last two stations. In the first one (which is obligatory for all products) a code with instructions relative to the last two stations is read. The user enters this code in the edit box at the right of the screen and clicks on the 'Accept' button to start the 3D simulation.

The system has the following inputs:

- Beginning and end induction sensors for all cylinders in the different stations.
- A bar code scanner which reads the code number of the product (inserted before simulation by the student in the edit box).
- Three presence detectors, one for each station.

Control actions include compression and expansion of all cylinders (two in the last two stations and one with the bar code scanner in the first one) as well as the line movement (forward and backwards).

C. Garage door

Figure 3 shows two different views of the 3D garage door in the repository. The upper view corresponds to the initial state of the system: the garage door is closed and the system is waiting for a car to arrive. In the second view a client comes and the door is opens to his request. The entries for the control are in this case:

- An optical sensor to detect the presence of a car. This is simulated by a red line which crosses the screen in the upper view.
- An overcurrent detector which turns on the red light placed in the top left corner of the door. The event is simulated by a checkbox in the right hand side of the screen.
- Sensors which detect that the garage door is open or closed.
- Two ways of opening the door for a potential user: either using a key or a remote control. These two events are simulated by two push buttons in the right hand of the screen.
- A reset event which leads the system to its initial waiting state (i.e. the door closed and the emergency light closed, assuming the optical sensor detects no car).

Control actions include the emergency light activated by the overcurrent sensor as well as the opening and closing of the garage door.

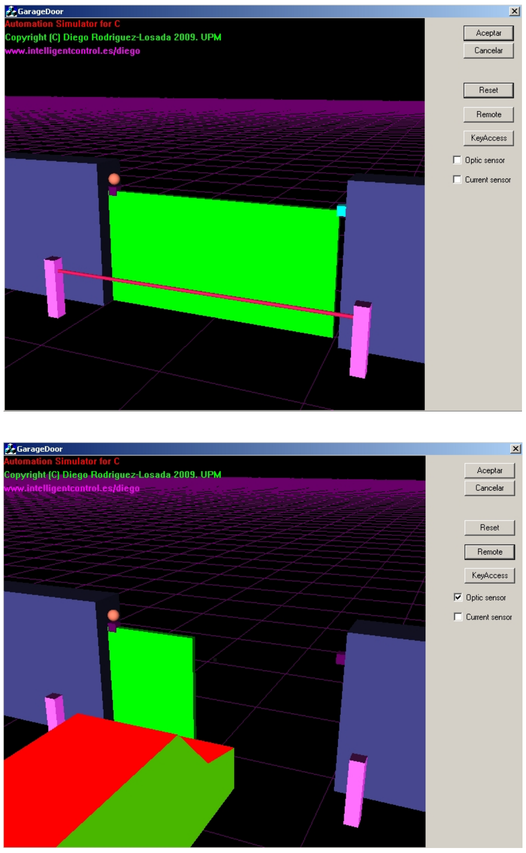

GarageDoor
Automation Simulator for C
Copyright (C) Diego Rodriguez-Losada 2009. UPM
www.intelligentcontrol.es/diego
Aceptar
Cancelar
Reset
Remote
KeyAccess
Optic sensor
Current sensor

Figure 3. Two different views of the 3D garage door system.

D. *Pneumatic cylinder working table*

Figure 4 shows the 3D pneumatic working table available in the system's repository. It contains three double effect cylinders which expand and compress according to the controllers instructions attacking a piece situated at the centre.

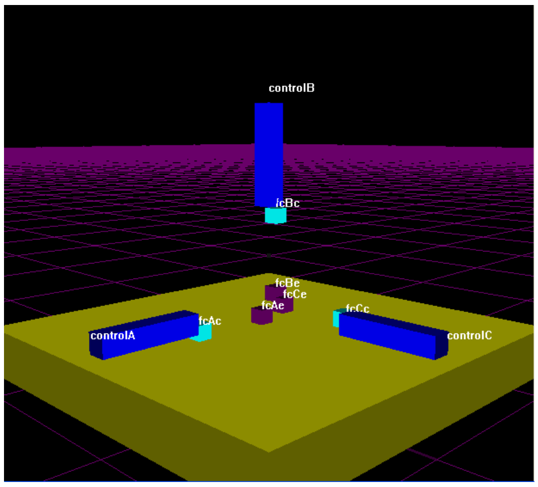


Figure 4. The 3D pneumatic working table

The inputs of the sequential control program are the following:

- Sensors to detect expansion and compression of all cylinders.
- A 'Start' button simulated by an equivalent push button in the right of the screen.

The system pre-actuators are three electro-pneumatic valves of two positions which allow expansion and compression of the cylinders.

III. Assignments

In this first course using the simulator we have asked the students to implement any reasonable control function for the four systems currently available (i.e. according to common sense semantics and avoiding the kinematic error models). It is to be noted that the systems, in spite of their apparent simplicity, are already flexible enough to allow for a wide variety of controlling schemes with different levels of difficulty.

A possible assignment for the manufacturing line is to assume that the bar code in the product (read in the first station) tells the controller that the product needs to stop at the last two. Another possibility would be to assume a stoppage in a different order (i.e. first in the last station, then back to the second and then once more to the first). Additionally, the cylinder expansion-compression sequences at each stop could depend on the concrete sequence read by the bar code scanner etc. Similarly, the lift could be programmed in a standard way or it could prioritize the latest order and similar variants can easily be found for the remaining systems in the repository.

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

IV. IMPLEMENTATION

The tool has been implemented in C++ using the Visual Studio development kit, and is currently only available for Windows platforms. The system repository employs the 3D accelerator graphical libraries OpenGL (which come in the software development environment) and the portable Window Manager (GLUT).

The development process has strictly followed the object oriented design (OOD) paradigm and engineering software techniques have been applied to decouple the graphical interface from the logical parts of the system (to the extent of being compiled in a separate library). The graphical representation of the different systems is broken down into common parts (i.e. objects which encapsulate the drawing information of light bulbs, sensors, doors etc.) all of which share a common virtual interface. This makes the tool much easier to manage and, more importantly, allows building a repository of common elements which may then be employed in any new control system added to the repository.

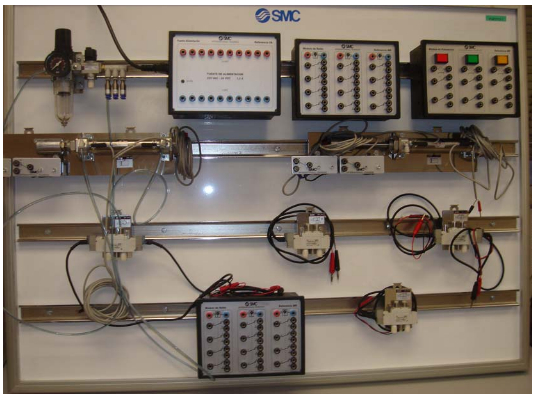
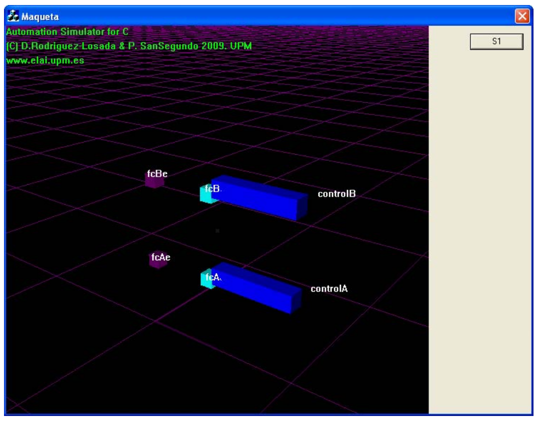


Figure 5. A two cylinder workbench used in practical lessons.

Adding a new system to the repository becomes a very easy task if it uses the components already defined in the repository. An example: the department uses in some of the practical lessons a two double effect cylinder pneumatic workbench (see Figure 5). It employs electro-pneumatic valves as pre-actuators to compress and expand both cylinders and uses different types of sensors to detect end of expansion and compression. The task is to program a PLC to produce a pre-defined sequence of expansion and compression of the cylinders. A more detailed description is available at [4].

It took only a couple of hours for one of the tool programmers to implement a very simplified, but yet again useful, simulated system of the workbench and add it to the repository. The system had only the two cylinders with sensors to detect expansion and compression and two state valves which had already been defined and employed in the four previously described systems.



Automation Simulator for C
(C) D.Rodriguez-Losada & P. SanSegundo 2009. UPM
www.elai.upm.es

Figure 6. A two cylinder workbench

However, it could already be employed to define different kinds of assignments for the students. The graphical interface of this workbench is shown in Figure 6.

This year the students were assigned four different software projects which shared a common workspace. Once the finite state machine program (which must be written in C) is implemented, it is possible to compile the particular project with the new control function and simulate the system. Figure 7 gives a partial example of a typical control function that the student must implement. The declarative data defining the identifiers for sensor entries and actuators at the top of the file is already given to the student. The finite state machine program is encapsulated inside the definition of the control function (void *control* () in the figure) which is initially empty.

```
#include "control.h"

//Inputs
bool S1;
bool fcAe,fcAc;
bool fcBe,fcBc;


//Outputs
int controlA;
int controlB;


int estado=0;

void control()
{
    if(estado==0)
    {
        if(S1)
        {
            estado=1;
            controlA=1;
        }
    }
    if(estado==1)
    {
        if(fcAe)
        {
            estado=2;
            controlA=0;
            controlB=1;
        }
    }

    //...
```

Figure 7. Part of a typical control function for a system in the repository.

V. Conclusions

This is the first year that the tool is being tested and, as yet, it has not been possible to assess the overall effect on our students. The tasks given to them related to the software tool have been included as part of the requirements needed to pass the practical exam. We do have at the moment some personal very positive feedback from some of the pupils which would indicate that the approach taken is a step forward in the correct direction. At the end of the semester it will be possible to have a better formed opinion to this respect.

Further work in the near future will focus in developing new components for the repository (i.e. valves, pumps) and building additional system models.

References

[1] Course on Programmable logic controllers imparted at EUITI. URL: www.elai.upm.es/spain/~/Automatizacion/Automatizacionbas.htm

[2] ELAI Department at EUITI technical school in Madrid (UPM). URL: www.elai.upm.es

[3] WinCC (SCADA) Siemens software application. URL: www.automation.siemens.com.

[4] Simatic Manager 5.4 Siemens software application. URL: www.automation.siemens.com.