Grafcet to Arduino: Edit and Upload Grafcets on an Arduino Boards

Maurice Comlan, David Delfieu, Narcisse Assogba

Grafcet to Arduino : Edit and Upload Grafcets on an Arduino Boards

Maurice COMLAN
Université d'Abomey-Calavi, Bénin
comlan@hotmail.fr

David DELFIEU
Université de Nantes, France
david.delfieu@univ-nantes.fr

Narcisse ASSOGBA
Université d'Abomey-Calavi, Bénin
assogbanarcisse@yahoo.fr

***Abstract*—This paper deals with the design and implementation of a Grafcet editor and simulator, allowing to generate a directly executable program on an Arduino board from a Grafcet. This work is intended to be a solution to help carrying out automation laboratory exercises; and, more generally, to offer a Programmable Logic Controller (PLC) development environment allowing to obtain low-cost PCLs based on Arduino modules, from the Grafcet formalism. Arduino boards are one of the most accessible and easy-to-use microcontroller boards and Grafcet is a well-known and essential model in the world of automated systems. All these reasons motivated the choice of these technologies. Finally, to achieve the set objectives, we started from a free Grafcet editor JGrafchart, developed in Java and that we adapted by including, amongst other things, the code generator for Arduino.**

***Index Terms*—Grafcet, Arduino, Programmable Logic Controller, Code generator**

I. Introduction

The graphical representation describes the sequential operation of an automated system unambiguously and in a way that is comprehensible to all categories of personnel. Indeed, the human eye is able to grasp, with a glance, a sequential evolution represented graphically [1]. Among the possible methods are the flowchart, the state machines, the Grafcet, the Petri nets.

The Petri nets model was created in 1962 by Carl Adam Petri [2], to model communicating processes. The major interest of Petri nets, for the modeling of control systems, comes from the fact that they allow to represent in a simple, clear and graphic way the concepts of parallelism, synchronization, resource sharing, communication, causality [3]. Depending on their characteristics, Petri nets can be simple, autonomous or not, predicates, color, continuous, synchronized, timed, hybrids ....

From ordinary Petri nets derives the Grafcet. Nevertheless, the semantics of the two (2) models differs [4]. The Grafcet, standardized in 1988, makes it possible to describe the behavior of the sequential part of the control system of industrial processes [5], through a PLC, a programmable electronic machine intended to control in industrial environment and in real-time automated systems. In industry, the behavior of a process is physically obtained thanks to an industrial programmable logic controller: an electronic machine equipped with a programmable memory, intended to control automated systems in an industrial environment and in real time. It is a simplified computer, which is physically connected by an input interface to sensors and by an output interface to actuators. They are relatively expensive and difficult to acquire.

The behavior of an PLC can also be achieved with a computer or a card with a microcontroller such as the mbed modules [6], the Raspberry PI [7], LaunchPad [8] or Arduino boards. Which are much more accessible and offer acceptable performance for automated applications of medium size. For example, the Arduino UNO card (entry-level card) offers a 20Mhz processor, 32K of RAM and 14 IO [9].

Our contribution: The development of automata is done today from description in Grafcet or Ladder language. Apart from their relatively high costs, the automata are difficult to acquire. This situation is a hindrance to the realization of industrial automation laboratory exercises. On the other hand microcontroller cards present a better compromise between price, accessibility and performance to obtain the same results. The objective of this project is to development an automaton environment that makes it possible, from the Grafcet formalism, to produce low cost PLCs (Figure 1).

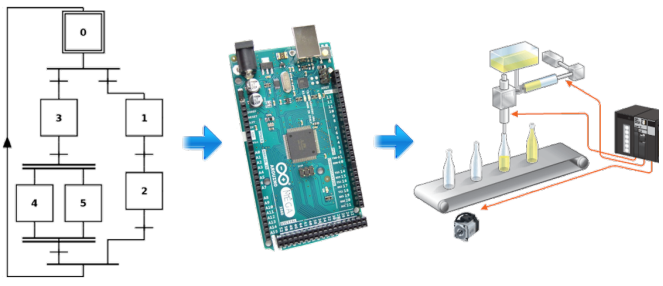


Fig. 1. Embedding Grafcet on Arduino

This paper is organized as follows. We first give related works in Section II. We will talk about the Programmable Logic Controller, the Grafcet and the Arduino Project. Then, in Section III, we present GrafcetToArduino, a tool to edit and upload grafcet on an arduino board. Finally, In Section IV we present a case study. We use this tool for the realization of a fire detection system.

II. Related works

A. Programmable Logic Controller

A programmable logic controller (PLC) controls the manufacturing processes for integrated production lines and equipment. PLCs were designed to replace the need for a large bank of relays or timers in facilities with numerous inputs and outputs. Due to their durability and ability to automate multiple processes, PLCs have become a staple in modern manufacturing. In this section, we will review the basic architecture of programmable logic controllers [10].

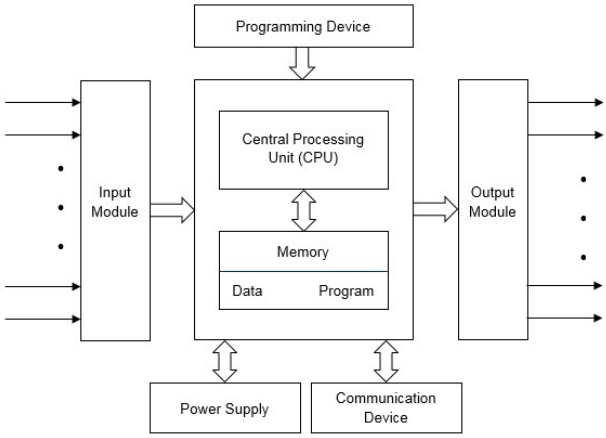


Fig. 2. PLC Architecture [10]

The main components of a PLC consist of following basic parts:

1. Power Supply : The power supply provides power to the PLC by converting the available incoming AC power to the DC power required by the CPU and the circuits in the Input and Output (I/O) interface modules to operate properly. This rectified voltage is transferred across back plane of the chassis. Therefore, careful selection of a suitably sized power supply is important to ensure it has the capacity to provide the necessary "bus power" to supply all Input and Output modules (I/O modules).

2. Processor Unit or Central Processing Unit (CPU) : The processor unit or central processing unit (CPU) is the unit containing the microprocessor and this interprets the input signals and carries out the control actions, according to the program stored in its memory, communicating the decisions as action signals to the outputs. The CPU comprises two components: the Controller and the Memory System

3. Input and Output (I/O) Modules : I/O modules are available as either input only, output only, or a combination of inputs and outputs. The I/O section establish the interfacing between physical devices in the real world outside the PLC and the digital arena inside the PLC. Typical input modules have either 8, 16 or 32 input terminals.

4. Programming Device : A PLC requires a programming terminal and programming software for operation. The programming device can be a handheld device, a desktop console, or a computer. The programming terminal is used for Programming the PLC, Monitoring the PLC operation, Download a ladder logic program (sending of a program from the programming terminal to the PLC) and Upload a ladder logic program (sending of program from the PLC to the programming terminal).

B. The Grafcet

The Grafcet is useful for designing graphs that give a graphical and synthetic representation of the behavior of systems. It is used in industry to describe the sequential behavior of processes. Inspired by the Petri nets and standardized by the IEC under the IEC-60848 standard [5], it obeys rules of syntax and well-defined semantics.

A Grafcet is basically composed of a set of:

- steps, they describes a system state. A step is either active or inactive. One or more actions are associated with it, to indicate the behavior of an output variable;
- transitions, which indicates the possibility of activity evolution between two or more steps. A Boolean expression, called receptivity, is associated with each transition;
- directed link, which connects one or several steps to a transition, or a transition to one or several steps.

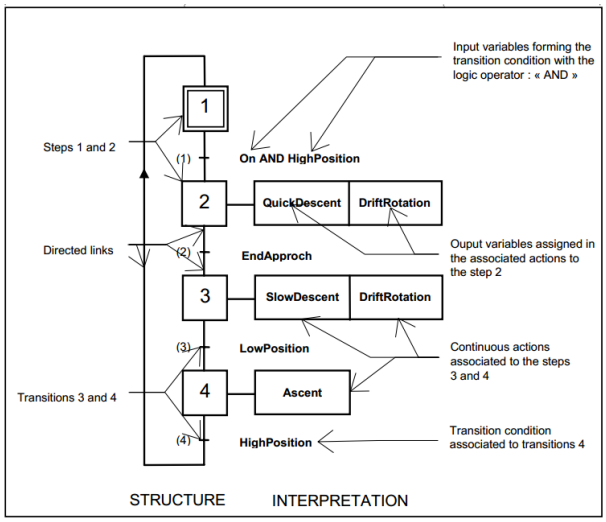


Fig. 3. Basic elements of a Grafcet

In addition to these basic elements, there are more complex components such as macro-steps, encapsulating steps, parallelism and synchronization elements (divergence/convergence).

IEC-60848 defines five (5) evolution rules. These rules describe the principle of sequential evolution between situations (all the stages active at a given moment and forming the system state) of the Grafcet.

1. Initial situation, choosen by the designer, is the situation at the initial time;

2. Clearing of a transition, a transition is said to be enabled when all immediately preceding steps linked to this

transition are active. The clearing of a transition occurs when the transition is ENABLED, AND WHEN its associated transition-condition is TRUE;

3. Evolution of active steps, the clearing of a transition provokes simultaneously the activation of all the immediate succeeding steps and the deactivation of all the immediate preceding steps;

4. Simultaneous evolutions, several transitions which can be cleared simultaneously are simultaneously cleared;

5. Simultaneous activation and deactivation of a step, if during the operation, an active step is simultaneously actived and deactived, it remains active.

C. Arduino boards

Arduino, is a generic brand name of the Italian company Smart Projects, given to a set of electronic cards. The diagrams of these cards are published under free license. Twenty-four (24) versions of Arduino-type cards have been produced and sold commercially until 2017 (Uno, Mega, Nano, Leonardo, Due ...) [11].

An Arduino module is generally built around an Atmel AVR microcontroller (ATmega328, ATmega32u4 or ATmega2560 for recent versions, ATmega168, ATmega1280 or ATmega8 for older versions), and complementary components that facilitate programming and interfacing with other circuits. Each module has at least one 5 V linear controller, a 16 MHz quartz oscillator and uses most microcontroller inputs/outputs for interfacing with other circuits.

The different versions of the Arduino cards work under the same general principle.

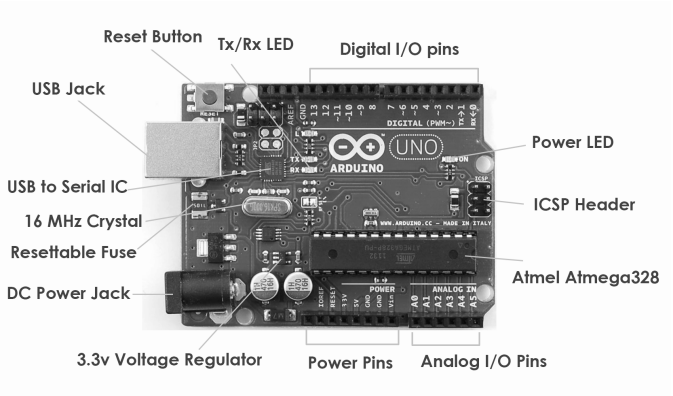


Fig. 4. Diagram of an Arduino card

The microcontroller is pre-programmed with a bootloader so that a dedicated programmer is not required. The modules are programmed with a serial TTL connection, but the connectors allowing this programming differ according to the models.

The programming software of the Arduino bords is a free, cross-platform Java application, serving as a code editor and compiler, which can transfer firmware and program through the serial link (RS-232, Bluetooth or USB depending on the module). The programming language used is C++, compiled with avr-g++.

III. GrafcetToArduino : A tool to edit and upload grafcet on an Arduino board

A. Editor

We drew inspiration for this proposal from the Jgrafchart, Automgen 8 and Unity Pro XL interfaces.

The main interface will consist of the following elements:

- a menu bar;
- a quick action bar, below the menu bar;
- a lateral pallet for the Grafcet components;
- an editing area;
- and an area for displaying errors and logs.

The menu bar : It will consist of six (6) menus (File, Edit, View, Execute, Automaton, Help). *The action bar* : It gathers just below the menus a series of the most important actions during editing. *The Side pallette* : Located to the left of the main interface, it offers the Grafcet components (step, transition, discrepancies, comments, etc.) that can be selected and dropped in the editing area. *The editing area* : It is in this area that the Grafcet is drawn, it receives the elements available in the side palette. At this level, double-clicking on a component opens a new window that allows you to edit the properties of the component. A right-click on the other hand opens a context menu containing, among other things, the actions copy, cut, paste, delete, properties and other actions that will depend on the type of component. If the right click is made in a zone without any element, then you get actions allowing to quickly add some elements of the palette (Step, transition, comment...) and other actions like paste, wizard, variables, preferences. *The error and log display area* : This area completely at the bottom of the main interface will display errors and messages resulting from the execution of an action. It will serve as the first basis for debugging and will allow the user to have a feedback of the actions they perform.

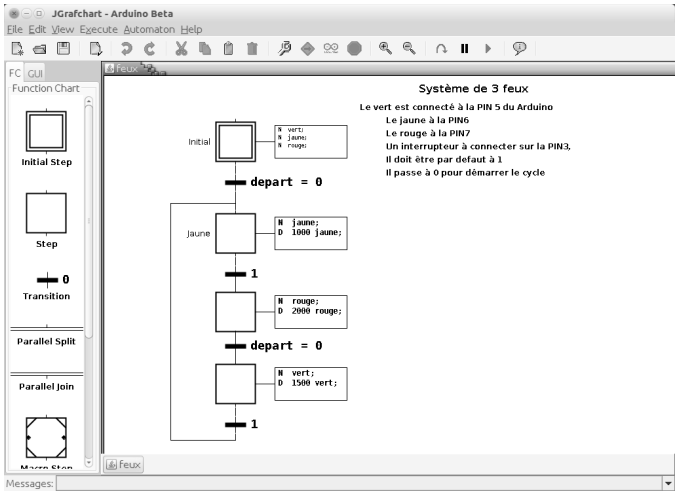


Fig. 5. Editor main interface

In order to reach our goal and keep on schedule, we started with the JGrafchart software. JGrafchart is a freeware

developed at the Automation Department of Lund University in Sweden [12]. The software is developed in Java with the Swing [13], enabling it to be deployed on all operating systems. A characteristic that was decisive in our choice.

To adapt the application to our needs, we had to make some changes by disabling certain features to lighten the application and then developing new ones according to the specifications.

So :

- the menu "Misc." (Others) has been disabled. This menu managed the application's interconnection to a computer network via a server on which files are shared using the DPWS protocol.
- the control module by PID, a control method often used for servos [14], has been disabled. This module was not part of our work.
- the side palette has been lightened by removing components related to disabled modules.
- the "Automaton" menu, which manages interactions with the connected PLC (Arduino board) has been created. Via this menu, it is possible to edit Grafcet variables; to define the type of card and the COM port to which it is connected; to export the source code for Arduino generated from the Grafcet; to directly upload the code to a pre-configured card.
- the variable management module has also been recreated, to make it easier to use the application for professionals familiar with environments like Unity Pro or PL7. From the new module it is possible to create and manage three (3) types of variables:
  - the standard variable;
  - the function block ;
  - transition session.
- changes have also been made to the edit box, including the edit menu of steps and transitions.
- finally, some adjustments have been made to the "File" menu, adding the "Open recent file" submenu, which gives access to the last edited files. And the upload icon has been added to the quick action bar.

The executable and the source code of the editor are available at :

https://github.com/maurice-comlan/Grafcet-To-Arduino.

B. Translation of Grafcet into Arduino source code

For the interpretation of the Grafcet and its translation into source code for Arduino, we wrote pseudo interpretation code 1.

CODE 1. Interpretation algorithm

```
Define the list of steps; /* Each step
including the list of its actions */
Define the list of transitions; /* Each
transition including its transition
condition, the list of its preceding steps
and the list of its following steps */
Create an empty set of fireable transitions;
Activate the initial steps;
Execute associated actions;
While TRUE Do
  Clear the set of fireable transitions;
  For each transition Do
    If transition validated AND transition
    conditions satisfied Then
      Add transition to the set of
      fireable transitions;
    EndIf
  EndFor
  If the set of fireable transitions is
  not empty Then
    For each fireable transition Do
    /* In simultaneous theory */
      Fire the transition;
      Disable the previous steps to the
      transition;
      Enable the following steps at
      transition;
      Execute the actions associated
      with the activated steps;
    EndFor
  EndIf
EndWhile
```

For the implementation of the Pseudo-code 1 we have created three (3) object models, representing three (3) main elements of a Grafcet.

- the *action* objet model;
- the *step* objet model;
- the *transition* objet model.

The definition of these object classes, combined with the use of some libraries created for Arduino, facilitated the translation of the Pseudo-code 1 into C++.

IV. APPLICATION CASE: FIRE DETECTION SYSTEM

To test the performance of the environment we have set up, we have developed an application case. It is a fire detection system based on a dual capture system. Excessive light at the fire scene is initially captured by a flame sensor that triggers a first light signal. The abnormal temperature increase is then captured by a temperature sensor which triggers a second light signal. These two conditions trigger a third light signal followed by an audible signal. The action of an operator on a button stops the different light and sound signals.

This case is very interesting in that it tests several aspects of our system:

- taken into account action on analog and digital outputs;
- steps without action or with multiple actions;
- simple and complex transition conditions;
- forced transition;
- parallel evolution and synchronization;
- finally the compilation and the transfer on card is tested.

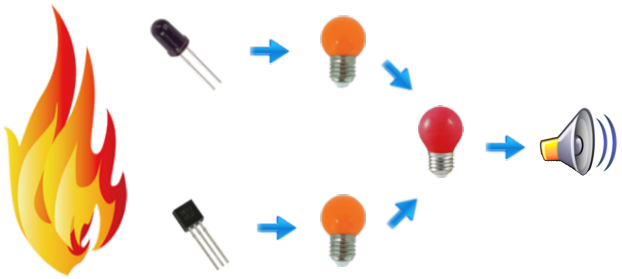


Fig. 6. Principle of fire detection

A. System Grafcet

Figure 7 presents the Grafcet proposed for the application case. The variables used for the realization of this Grafcet are listed in Figure 8. The XML code and the C++ code for Arduino for application case are generated by our GrafcetToArduino tool.

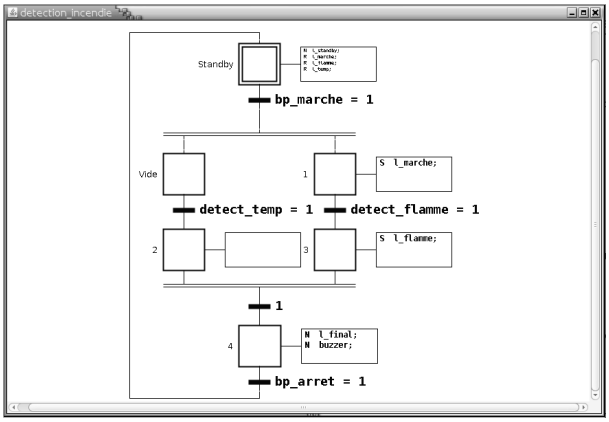
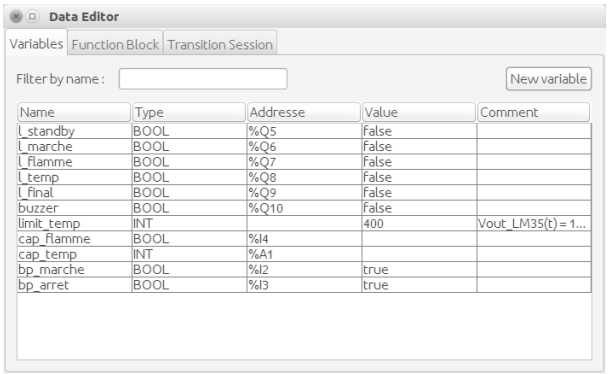


Fig. 7. Grafcet of fire detection system



Name	Type	Addresse	Value	Comment
l_standby	BOOL	%Q5	false	
l_marche	BOOL	%Q6	false	
l_flamme	BOOL	%Q7	false	
l_temp	BOOL	%Q8	false	
l_final	BOOL	%Q9	false	
buzzer	BOOL	%Q10	false	
limit_temp	INT		400	Vout_LM35(t) = 1...
cap_flamme	BOOL	%I4		
cap_temp	INT	%A1		
bp_marche	BOOL	%I2	true	
bp_arret	BOOL	%I3	true	

Fig. 8. Variables used in the Grafcet of the application case

B. Montage diagrams

The diagrams have been made with the software Fritzing [15]. The components used for the realization are listed in the Table I.

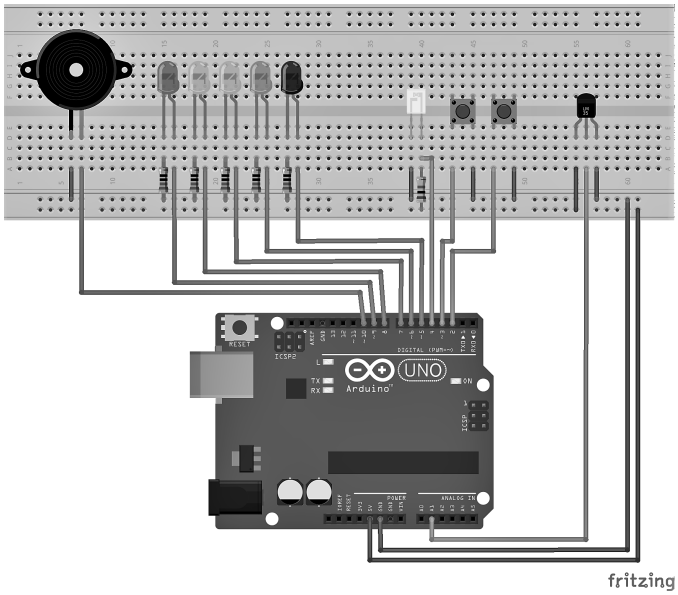


Fig. 9. Montage diagrams with Fritzing : View on breadboard

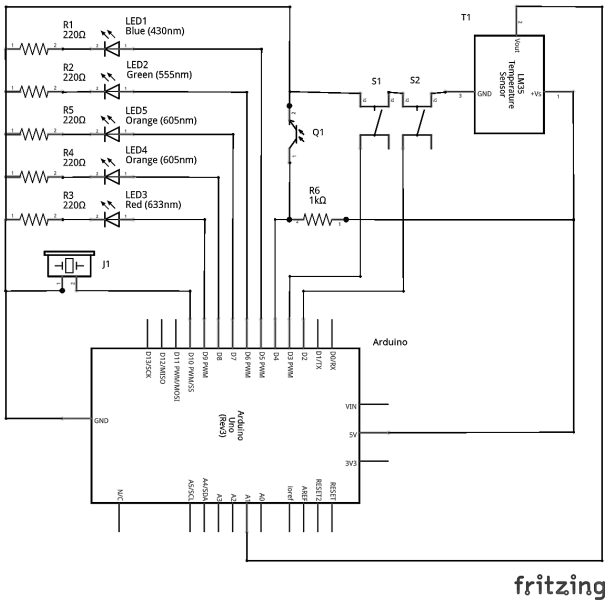


Fig. 10. Montage diagrams with Fritzing : Schematic view

TABLE I
LIST OF COMPONENTS

Components	Quantity
Arduino Uno	1
Buzzer	1
LED	5
Résistances	6
Bouton poussoir	2
LM35	1
Phototransistor	1

V. ANALYSES ET DISCUSSIONS

For the Grafcet edition, we relied heavily on the performance already offered by JGrafchart [12]. This made it

easier to focus on other features. However, we have lost performance in terms of checking the consistency of actions associated with steps and conditions associated with transitions because the basic language to perform these operations has been modified; we went from the Grafchart language [12], developed by the JGrafchart team, to the C ++ language, used to program Arduino boards. It was therefore necessary to integrate a C ++ syntactic and semantic analyzer into the editor to maintain this performance. This change of language also had an impact on the simulation mode which strongly depends on the syntactic and semantic analyzer to calculate the value of the variables and to interpret the actions.

For this first version, some Grafcets functionalities have not been taken into account. These are the means of structuring by forcing the situation of a partial grafcet and of structuring by encapsulation [5].

The tests carried out on the Arduino Uno boards were conclusive. The actions associated with the steps are performed in the normal order. The stages are indeed activated and deactivated as the transitions are completed. Due to lack of hardware, we were unable to test the performance of the application on other types of Arduino boards.

Also the use of the software in a Windows environment requires the installation of the program *nmake*, in addition to the C ++ compiler, to take into account the *makefile* which is not supported by default on Windows [16]. But on the other hand if the design should not lead to execution on an Arduino board and is limited only to the generation of code for Arduino, the application works perfectly without the need for additional software except the JVM.

Our vision is to create an open source and multiplatform automation workshop. To achieve this objective, improvements will be continually added to this version of the software to take into account other Grafcet / SFC functionalities and integrate other types of models such as Ladder and extend the possibilities of low-cost PLCs.

VI. Conclusion

The work carried out on behalf of our thesis project consisted in the design and implementation of a Grafcet editor and simulator, allowing us to generate a program directly executable on an Arduino card from the Grafcet. The interest of the work is to provide an environment to facilitate the achievement of the objectives of the project. practical industrial automation work on low-cost automatons, designed around Arduino microcontroller cards.

To achieve this objective, we first studied the specifications of one of the most widely used models in the field of automation and process control: the Grafcet. We then analyzed the operation and functionalities of three (3) existing Grafcet editors. From this analysis, the free editor JGrafchart was selected as the basis for the development of our solution. JGrafchart was then modified and adapted to the requirements of the specifications.

Once the editor was up and running, the second step was to set up the Grafcet translation module in source code for Arduino and send code to an Arduino board. At this level, a translation algorithm based on the work of [17] has been implemented, followed by an analysis of the principle of compiling Arduino source codes and their transfer to the board via Arduino's IDE.

To appreciate how the editor works, an application test case was developed to test the most important features and validate the work that was done.

The scalability of this type of application is a major advantage because it will improve the editor's functionality by adding, for example, the languages defined in the IEC-61131-3 standard to the list of languages used to edit actions related to a step or define instructions for block functions. In perspective to this work, it would be interesting to add to the editor taken into account other formalism such as Ladder and also extend the possibilities of automatons to other microcontroller board.

References

[1] E. H. El Mimouni, A. Hanafi, R. Lajouad, A. Rmicha, A. Jemily, A. Errahouti, and K. Moujibi, *Sciences de l'ingénieur, 1ere STE*, 1st ed., 2006.

[2] C. A. Petri, "Communication with automata," Ph.D. dissertation, PhD thesis, Institut fuer Instrumentelle Mathematik, 1962.

[3] M. Nielsen, G. Plotkin, and G. Winskel, "Petri nets, event structures and domains, part i," *Theoretical Computer Science*, vol. 13, no. 1, pp. 85 – 108, 1981, special Issue Semantics of Concurrent Computation. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0304397581901122

[4] M. Sogbohossou and A. Vianou, "Formal modeling of grafcets with time petri nets," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1978 – 1985, 2015.

[5] C. E. I. CEI, "Cei 60848: Langage de spécification grafcet pour diagrammes fonctionnels en séquence," 2002.

[6] M. M. Macías, J. E. Agudo, C. J. G. Orellana, H. M. G. Velasco, and A. G. Manso, "The" mbed" platform for teaching electronics applied to product design," in *2014 XI Tecnologias Aplicadas a la Ensenanza de la Electronica (Technologies Applied to Electronics Teaching)(TAEE)*. IEEE, 2014, pp. 1–6.

[7] E. Upton and G. Halfacree, *Raspberry Pi user guide*. John Wiley & Sons, 2014.

[8] A. V. Parkhomenko, "Development and application of remote laboratory for embedded systems design," 2015.

[9] Arduino, "Arduino Uno Rev3," 2017. [Online]. Available: https://store.arduino.cc/arduino-uno-rev3

[10] "Plc – architecture/hardware," jul 2021. [Online]. Available: https://www.plcautonetics.com/plc-architecture/

[11] "Arduino - compare," jul 2017. [Online]. Available: https://www.arduino.cc/en/Products/Compare

[12] Lund University, "Automatic Control - Grafchart," 2017. [Online]. Available: http://www.control.lth.se/Research/tools/grafchart.html

[13] R. Eckstein, M. Loy, D. Wood, and M. Loukides, *Java swing*. O'reilly Cambridge, MA, 1998.

[14] D. Ross, E. Deguine, and M. Camus, "Asservissement par pid," *rose. eu. org*, vol. 3, 2010.

[15] A. Knörig, R. Wettach, and J. Cohen, "Fritzing: a tool for advancing electronic prototyping for designers," in *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, 2009, pp. 351–358.

[16] Microsoft, "Makefiles (Windows)," 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa380049(v=vs.85).aspx

[17] R. David and H. Alla, *Du Grafcet aux réseaux de Petri*, 2nd ed., ser. Série automatique. Hermès, 1992.