Literature review on technological aspects of replacement of Programmable Logic Controllers with Arduino

Aniruddh Mali
EC Department
HGCE
Vahelal, India
aniruddhkmali@gmail.com

***Abstract*— Prgrammable Logic Controllers i.e. PLCs are being used for industrial automation as well large scale control system. It is used for sequential slow operations. It has been adopted as a traditional solution for automation industry and available only with proprietary vendor having license. With advancement of technology and emerging open source technological approach, arduino become a very famous and handy electronic development facility. It has possibilities of to be used as PLCs. In this paper, technological aspects of replacement of PLCs with Arduino have been discussed.**

***Keywords*— PLC, Arduino**

I. INTRODUCTION

PLC (Programmable Logic Controller) has been and still is the basic component of the industrial automation world. Industrial application made the PLC systems being very expensive, both to buy and repair, and also because of the highly specific skills requested to software designers to extract the maximum potentials from controllers. Arduino is a kind of universal open source programmable controller, although it is only the "core" and in any case it has been built for general applications; with a little of external hardware; essentially interfaces capable of transferring signals from sensors and to actuators, reducing the EMI which may damage the microcontroller and an appropriate software may, however, become something very similar to a PLC. Since for a long time PLCs earned a trust and reliability of industy due to its ruggedness and performance but its programming softwares makes it costlier. Now a days, some of manufacturers avail low cost PLC units but they charged high for its softwares. Arduino has the core of Atmega microcontroller which makes it a strong competitor of PLCs. As open source electronics development becomes a hot topic now a days, it is recommended to find out effective innovative solutions inplace of traditional methods.

II. APPROACH

There are two approaches to turn Arduino into a Programmable Logic Controller. First one is to write a program using KOP language (ladder). To do that, one should use two more applications in addition to Arduino IDE i.e. LDmicro that is the editor and compiler of ladder code [1]; second consists of a web page that will help us creating the code for the 'ladder.h' library; for simplicity's sake, in this guide one will consider only the DIGITAL I / O with no special features. The second method is to use plcLIB [2] (a library we suitably modified to take advantage of the IO shield coupled with Arduino UNO) so that you can edit our project code with a language similar to AWL (instructions: IF, AND, OR) having the control on timers and other functions;

III. EXAMPLE

Let's take a problem of home automation: automate electric sunshades. We want to control the sunshades so that in case of a strong wind they must be automatically retracted, while unrolled only after the wind has calmed down. Its behavior should be similar for the different lighting situations: roll them during the night and unroll on daylight, but obviously the wind conditions should prevail over the lighting.

As a possible solution we could use a real PLC, but given the simplicity of the algorithm and high cost of it, we will proceed for customized solution. We will use two sensors, twilight NO (normally open) and wind NO that will be connected to the IO shield. In addition, we will have to adapt and change the power scheme of the sunshades engines so that Arduino could manage them.

1) LDmicro

Before writing the ladder code (contact diagram), similar to that in figure, we need to download the LDmicro executable from the link http://cq.cx/ladder.pl. Once downloaded and saved to your desktop, just double click on the LDmicro icon. Now, before proceeding with the application of ladder diagram, we have to write a draft of the program that we want to create.

Since the program is very simple and we use few variables only, we can use the most basic and immediate programming approach for the PLC: the wired logic. This methodology consists of simple Boolean equations that will always return as output (after the equal sign) a value that can be 0 or 1. The only operators used will be the AND (series), the OR (parallel) and NOT (negation). Input and Output variables in our program shall be the following:

– INPUT:
WIND SENSOR: XSEN_VENTO
LIMIT SWITCH ROLL: XFINE_COR_A
LIMIT SWITCH UNROLL: XFINE_COR_S
TWILIGHT SWITCH: XIN_CREP

– OUTPUT:
ROLLER: YAVVOLG
LED ROLLER: YL_AVV
UNROLLER: YSVOLG
LED UNROLLER: YL_SVO
IDLE STATE: YL_RIP

Note that the Arduino output pins have not yet been declared yet, while the “x” and “y” placed at the beginning of each variable represent respectively the input and the output. The software adds this letter automatically during the coding. Once declared the variables, we can proceed on writing the logic equations, taking into account that Arduino with our shield reads all inputs as HIGH (1) when the contact is open and LOW (0) when the contact is closed: therefore we must negate the logic of all the inputs to ensure that they are properly executed.

Now that we have our Boolean equations, we can back to LDmicro. By pressing the “C” key or clicking on the “instruction> include contact” menu, we get the inclusion of an open contact. Now, without changing anything, by pressing “L” on keyboard (after placing the cursor after Xnew, on the right side), we should obtain a segment identical to that of figure. Moving the cursor below Xnew (by using the arrow keys on keyboard) and pressing “C”, we will get a new segment, always named Xnew but in parallel with the previous one; now, we must click on Ynew and putting the cursor vertically on the left (before) of the output, we have to press “C” twice. By doing this we have added two more contacts, in series with the two (parallel) previous ones. To finish up, we just need to click on Ynew (but this time positioning the cursor below the output symbol) and then press “L”: so we have added another output in parallel to the existing Ynew.

Now we need to modify all contact names accordingly to those used in the equations. To do this we simply double click on the first Xnew contact we have created, then we set the name and contact type in the popup shell. We will assign the name “SEN_VENTO” and select “INPUT PIN” and “|/| NORMALMENTE CHIUSO” (NC). Remember that the software automatically assign the initial letter “X” or “Y” whether the variable stands for INPUT or OUTPUT, respectively.

Pin configurations:

```
pinMode(2, INPUT);
pinMode(3, INPUT);
pinMode(4, INPUT);
pinMode(5, INPUT);
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
pinMode(10, OUTPUT);
pinMode(11, OUTPUT);
pinMode(12, OUTPUT);
```


Now, move four files i.e. ladder.ld, ladder.cpp, ladder.h and pinmap.ini to a single folder called "ladder". Then move this folder to (W7) C: \ programmiX86 \ Arduino \ libraries (or for older editions of Windows (XP) to C: \ Program Files \ Arduino \ libraries). Open or restart the Arduino IDE and write the following code (NOTE: it is not intended for using of
timers or other special functions). From the menu "SKETCH", we select "import library" and choose "ladder". Now, type the following code, which is valid for all projects similar to this:

```
#include <ladder.h>
void setup()
{
PlcSetup();
}
void loop()
{
PlcCycle();
delay(10);
}
```

Now, please click on the "verification check" of our code and wait until the sketch has been compiled. We should get a positive result, if not please follow again the instructions above systematically. After the compilation is finished you can click on the icon on the side, to transfer our program to Arduino; wait until it's loaded and … Congratulations: you've managed to turn your Arduino into a controller very similar to its rival PLC. Now you just have to interface your system, respecting the logic that we have established for inputs and outputs.

2) plcLIB

In this approach you must download the suitably modified library. The plcLIB library that has been modified to optimally interface the IO shield with Arduino UNO, is dedicated to those who want to start studying the world of automation without having to "hack" the code too. Simply download the library plcLIB and once unzipped, paste the folder in "C:\programmiX86\arduino\libraries" for Windows 7 or to "C:\ program files\arduino\libraries" for older Windows versions (XP). The digital I / O must follow the naming shown on table:

TABLE I. DIGITAL I/O

Tipo	*Arduino pin number*	*N KA05*	*plcLIB*
Input	2	DI1	X1
	3	DI2	X2
	4	DI3	X3
	5	DI4	X4
	6	DI5	X5
	7	DI6	X6
Output	8	OUT1	Y1
	9	OUT2	Y2
	10	OUT3	Y3
	11	OUT4	Y4
	12	OUT5	Y5
Output	13	OUT6	Y6

By reusing the Boolean equations listed above, we write our code in the Arduino IDE: in this way, i.e. considering them as real equations respecting the order of operators (calculating parenthesis first and then the rest). Once you have edited this code, simply click on Verify and wait for the IDE to compile the sketch, we should get a positive result, otherwise revise the instructions stepbystep. Once the compilation is complete click on "Upload" to program the Arduino. Wait until the loading is completed and you have turned your Arduino to PLC.

Summary

Both methods plcLIB and LDmicro are feasible but plcLIB seems easy than LDmicro. Still there are some technical issues and challenges to replace PLC with Arduino completely as its being a traditional controller for automation only. There are challenges with ruggedness, redundancy and programming ease for large scale control system. There is a need of more refined methods and technological approaches to make arduino a real PLC.

Acknowledgment

Author would like to thanks arduino developers community.

References

[1] http://cq.cx/dlG
[2] http://adam.horcica.cz/tools/laddergen
[3] https://www.**arduino**.cc/
[4] http://www.electronics-micros.com/software-hardware/plclib-arduino/
[5] http://controllino.cc/

