Modular PC Based System for Educational Laboratory Experiments

GEORGE POPOV
Faculty of Computer Science and Technology
Technical University of Sofia
1000, Sofia, 8 Kl. Ohridski bld.
BULGARIA
popovg@tu-sofia.bg

Abstract: - This article concerns a modular automated system for conducting student laboratory experiments, which contains two parts: hardware and software. The hardware part is made up of PC and laboratory training kit with a constant analog/digital periphery and removable module (different for each experiment). The software part comprises two independent and complimentary subsystems: menu-oriented software and specialized language. Menu based system offers various research functions as a capture of transfer characteristic, printing, saving, etc. On the other hand, the specialized language for conducting laboratory experiments ULLE offers a possibility to draw up research programs~~,~~ by using the stimulus-response method. It is implemented as a resident driver use~~s~~ing the so-called conception of "live variables". When declaring a type of I/O variable (for connection to the laboratory training kits), its address is transmitted to the driver. By means of a periodic hardware interrupt, the driver performs auto-refresh of the I/O channels (of the hardware laboratory kit) and the associated I/O variables. Due to the fact the driver receives only the address of the variable, it can be run with programs written in different programming languages.

Key-Words: - Laboratory experiment, Engineering education, Analogue electronics, Digital electronics, Distance engineering education

1 Introduction

Despite the diversity and complexity of the specialized company software for automated systems for conducting laboratory experiments (labware) [1,2,3,4,5], it can be said that there are two basic approaches:

- *monolithic systems* – these are hardware systems whose functionality is available through user menus, as data is entered/displayed in windows or charts are to be taken;
- *soft systems* – these are systems that use a specialized language for conducting laboratory experiments.

Monolith systems are menu-oriented and they perform a strictly defined function as capture of transfer characteristic, set/read voltage to/from an appropriate output/input, printing, saving, etc. Their main advantages are the rapid introduction of the system and the easier conduct of laboratory experiments. On the other hand, they follow a certain model from which they cannot deviate, which deprives them of the ability to adapt to different types of tasks and applications.

Soft systems that are based on a programming language are more "agile", enabling the researcher to set up a specific research program according to the needs of the experiment. Another advantage of the software systems is that to them it is possible to emulate through a program code part of the studied electronic circuit (or hardware research). An example of this is to make a virtual link (for example Negative Feedback) via program code, reading a signal from output (of circuit), then scaling and passing it to the input of another circuit.

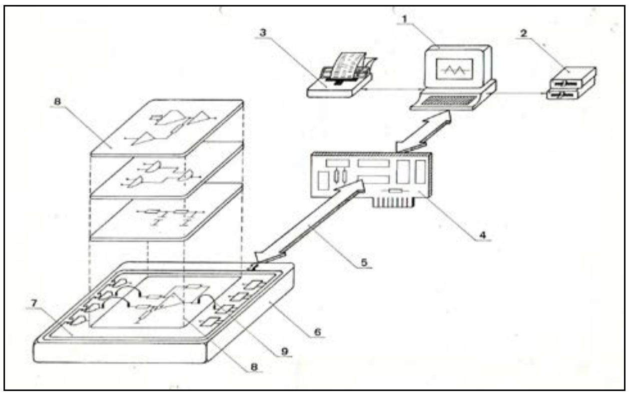

Fig.1. System for laboratory experiments MicroLab. Base (7) and exchangeable modules (8) can be observed on the picture.

The method proposed in the article was implemented in the microcomputer system MicroLab for conducting laboratory experiments [6,7]. It consists of a general-purpose PC and a „non-intelligent“ (non-CPU controlled) laboratory kit. The latter contains an base analog-digital periphery (4 analogue inputs and 4 analogue outputs) and a removable module (Fig. 1-8) as well, which is different for each laboratory experiment.

The researcher realizes the experimental layout of the laboratory model by performing the necessary connections by means of commutating cables (Fig. 2).

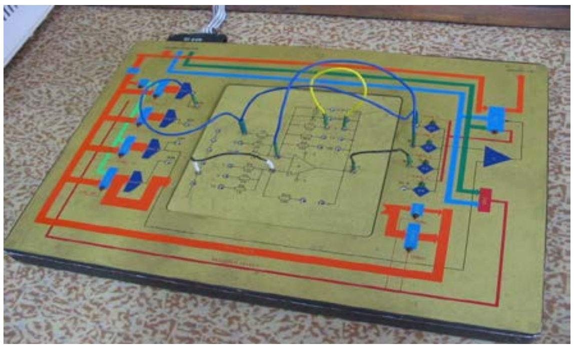

Fig. 2. MicroLab system is suitable for learning process: a large flat metallic corpus with changeable plate in the middle. Photo: Investigation of a non-inverting amplifier

It is easy to connect various external elements such as potentiometers, measuring instruments, electric motors, sensors, etc. (Fig.3.) to the laboratory kit.

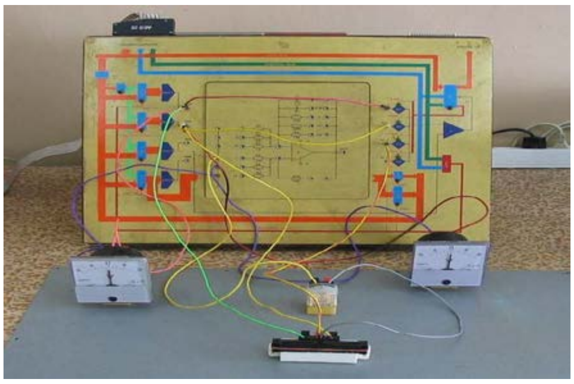

Fig.3. Experimental setup for "inventing" resistive transducers. The circuit elements (potentiometer and measuring instruments) are located outside the model. The potentiometer can be disassembled, allowing the student to have access to the resistive layer

The study is conducted using the stimulus-response method with the help of a special program written for the specific experiment. It sets the values of the voltages of the output channels and read the voltages of the input channels. A screen with a family of transfer characteristics is shown in Fig. 4.

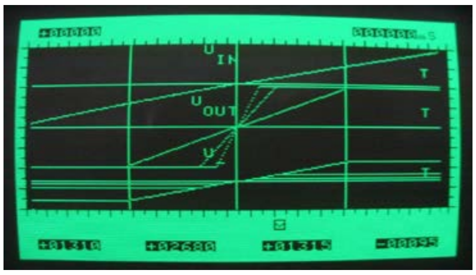

Fig.4. Transfer characteristics at different values of the gain set by resistors

The described labware provides a complete functional basis for conducting the experiments described in [8].

2 Description of Languages for Conduction of Laboratory Experiment

The universal language for laboratory experiments ULLE was established in 1988 at the Technical University of Sofia. It was developed over the years adequately to the software and hardware standards of the modern computer systems. The main idea of various versions of the language is to declare input/output variables of a type kit. Another feature is that these variables are refreshed automatically without engaging the researcher.

2.1 ULLE for PCs Apple II, IBM PC with DOS operating system

For Apple II PC, ULLE is an extension of the built-in language BASIC. The idea of this implementation comes from the built-in language operator &. When written in a specific point (address) of the main program, it makes the computer perform an additional user defined command from the special user program can be performed. A BASIC advantage is the researcher can send special commands in programming and dialog mode, i.e. to stop and continue the process of implementation (interpretation) of the program. In the IBM PC with DOS operating system, a ULLE universal resident driver has been implemented, working with most of the programming languages of this time: BASIC of C and Pascal.

The idea of this implementation is the following: after declaring a variable of type model, its address is intercepted by the extension of the language. Two types of actions are being carried out:

- If the variable is of type OUTPUT, its value is constantly fed by the dedicated hardware at the one of outputs of the stand (see Fig.3 and Fig.4, all 4 hardware outputs are on the left side);
- If the variable is of type INPUT, it is constantly refreshed by the one of 4 hardware inputs of stand. In this case, the selected hardware input on the lab stand continuously produces and updates the data in memory cells, which are really addresses of variables in the language of high level. The user (the program written in the language for conducting laboratory experiments) read them only when needed.

To achieve the above features, two conversions have been made. In the earlier version, the driver is called out by an interrupt 1Ch or 70 h, thus refreshing the corresponding input/output variables [9]. From here comes the term "live variable" and "Live Variables BASIC" or "LVBASIC" as a version of ULLE.
The latter version imposed uses a special hardware that monitors for a change in the values of the "living" variables and generates an interruption that invokes a driver execution. This implementation seems modern, but unfortunately, it would complicate greatly the laboratory staging. Below are given examples of code for these versions of ULLE (LVBASIC version).

2.1.1 Declaration of variable

- For the input channel on the laboratory stand:

&INP (<CHANNEL>, <VAR>, < VALUE>);

- For the output channel

&OUT (<CHANNEL >, <VAR>, < VALUE>);

where:

CHANNEL – the ID number of the input or output channel in Micro Lab periphery;

VAR –name of the "living" variable, which will monitor (and transmit) value in the chosen channel;

VALUE – the actual value in millivolts. If the voltage on the specified input changes with a bigger value than stated, an interruption with VALUE is generated to refresh the variable

2.1.2 Variable release

Release is done with the command:

& KILL <VAR>

After release, the "live variable" stops tracking (following) the channel, becomes a regular variable and retains its last value. This can be used to memorize value at a time and is extremely convenient when working with arrays.

2.1.3 Notation of data structures and operations

A living variable can participate in an arithmetic expression similar to an ordinary variable. For example, if U1 is the input voltage and U2 is the output and are both defined as "live variables", the transfer characteristic can be plotted with:

100 DRAW (Ul, U2): GOTO 100,

and the dependence of the output power with load R on the output voltage is plotted with:

100 DRAW (U2, (U2*U2)/R): GOTO 100

Filling a buffer (array) with data from an input channel is realized as follows:

```
10 DIM A [100]
20 FOR TO 100
30 &INP (1, A[I], 200)
40 NEXT I
```

Here the process of "come to life" on each new element of the array, while the previous one holds its value constant. In this case the follow line has to be inserted:

25 NOT STROBE THEN GOTO 25

A software strobe for the input data can be performed through the living variable STROBE, reading at the input channel of the system.
Similarly, the output of a predefined data buffer is implemented in the following way:

```
10 DIM A [100]          ; Array to output
20 &OUT (1, X)          ; Define living variable X for
                        output channel 1
30 FOR I=1 TO 100: X=A[I]: NEXT     ; Send the
                  array through the output channel 1
```

When it is necessary to store the current value of a "living" variable, for example X, it can be done in several ways:

-release the variable

&KILL X

-assign to a different variable (e.g. TEMP):

TEMP= X

-define another "live" variable on the channel:

&INP (1, Y, 150)

It is possible the same “living” variable X to be defined simultaneously as input and output. Then it will send the voltage observed on the input to the output channel.
LVBasic makes it easy to describe an experimental set-up and to implement linked data structures. The example below illustrates the implementation of a large-scale amplifier (through a real operational amplifier of the removable module) and a feedback with ratio specified by the constant FB. The input of the adjustable inverting amplifier is connected to an output channel 1 and its exit to input channel 2.

10 &OUT (1, O_1) ; Voltage output 1 will be equal to the value of the variable IN_1

20 INP (2, IN_2) ; The variable IN_2 tracks the value of channel 2.

30 INP (1, C) ; The variable C tracks the value of channel 1.

40 &O_1 = IN_2*FB ;Output is equal to Input*FB

50 PRINT IN_2 ; Print the current result.

60 IF C < 10000 THEN 40 ; While C is less than 10000 mV, the cycle 40-60 is running.

2.2. ULLE for PC with Windows OS
To make full use of the increased capabilities of the computer systems, the following ULLE implementations were made:

2.1.2 A realization based on multithreading environment, developed in TU-Sofia
The environment represents the core which supports time-sharing and cooperative mode as well as synchronization and communication of user level threads [10]. New features have been added to the kernel resident module API, supporting the entrances and exits of the model. The essential advantage of this realization is the better structure of the code, which allows the base of this ULLE version to build a newly designed hard system for conducting laboratory experiments.

2.2.2 Implementation by Open MP.
The underlying disadvantage of its own multithreading environment is that it represents itself as a process of Windows and cannot efficiently use the resources of multicore platform. For this purpose, the last modification of ULLE was made.

3 Remote Implementation of Laboratory Exercises

In conjunction with the ever-expanding distance learning, some experiments were made to conduct laboratory exercises from a distance.
The student receives a message containing a set of theory and tasks about analog circuit design, as well as a schedule for conducting remote exercise. Lab assistant at the university has set up five kits with different experimental stagings. The student accesses each of the computers and performs the tasks. By means of a videoconference program (e.g. Skype, Viber) they carries out the following actions:

1. Identification of the Student;
2. Giving technical consultation;
3. Test the learner;
4. Report of irregularities.

The first attempts to remotely conduct laboratory experiments were conducted with remote desktop application TeamViewer. The ability to record the session and establishing a connection via chat give additional advantages of this tions.
Students have the opportunity to work with a menu-based system and remote start the program on LVBasic or ULLE. As described above, it is possible to create virtual connections on the laboratory kit through the living variables.
A standard remote lab exercise consists of 3 stages, It starts with a simulation of the electronic circuit under investigation with a cloud simulation and a subsequent remote practice exercise. At the end of each exercise, the student must complete a lab report including tables, graphics and conclusions.

4 Future Works

Enthusiasts are working hard for the future development of the system. Laboratory experimental kit has proven in time with advantages of its shape and construction. Constructors are going to update hardware completely, using the Raspberry PI 3 board, which has the possibilities to connect different modules: mouse, keyboard and monitor. The sheet diagram of Raspberry kit with ADC and DAC converters is given in Fig. 5 and can be found in [13].
The Raspberry basis system has a lot of means for conducting a distance learning education. Built-in interfaces as LAN, Wi-Fi, USB and GPRS are sufficient for connecting the MicroLab with cloud infrastructure.

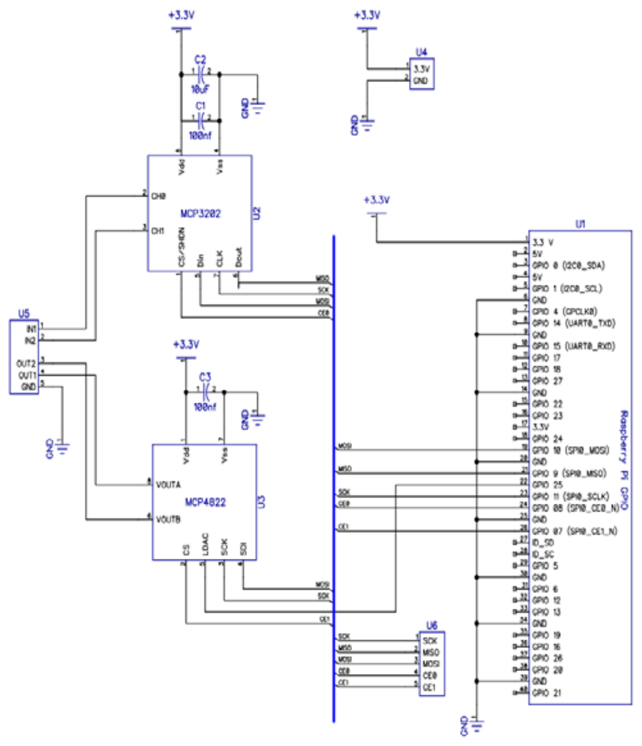

Fig. 5. Typical connection between Raspberry PI and ADC&DAC board

5 Conclusion

The system for laboratory exercises MicroLab is used to conduct laboratory experiments on analogue and digital circuitry disciplines for more than 30 years and has gained considerable experience. For the sake of brevity of the exhibition, not all the features of ULLE are given. Software provisioning was changing with different computer generations, following two main concepts: MicroLab Integrated (Monolithic) Environment and Universal Language for Laboratory Experiments ULLE. The conclusions drawn are:

- To enhance the experiment and put it in the forefront, the tools used should not be too complicated and distracting the learner;
- There's been a comparative survey among students in analogue circuitry, as most of them prefer the language extension with live variables instead of a library with corresponding functions;
- Using of BASIC is convenient, because it has an automatic declaration of variables, it provides opportunities for direct work with the interpreter (i.e. the ability to directly set/read out values from the model), including interrupting and continuing a program as well as inserting a code during the execution.

References:

[1] H. Austerlitz, AP, *Data Acquisition Techniques Using PC*, San Diego, California, (1991)

[2] Nuno Sousa, Gustavo R. Alves, and Manuel G. Gericota, *An Integrated Reusable Remote Laboratory to Complement Electronics Teaching*, IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES, VOL. 3, NO. 3, JULY-SEPTEMBER 2010

[3] V.J. Harward et al., "*The iLab Shared Architecture: A Web Services Infrastructure to Build Communities of Internet Accessible Laboratories,"* Proc. IEEE, vol. 96, no. 6, pp. 931-950, June 2008

[4] D. Magin and S. Kanapathipillai, *"Engineering Students' Understanding of the Role of Experimentation"*, European Journal of Engineering Education, 2000, Vol. 25, no. 4, pp. 351-358

[5] Mechkov C.,, *Computer Based Technical Laboratory for Conduction of Laboratory*, Experiments, Announcements, VMEI, 44th, Book 9 (1989) (in Bulgarian lang.)

[6] http://acumenlabware.com/electronics-training-kits/communication-lab-equipment.html

[7] Zhekov Z., C. Mechkov. *Modular System for Automation of Practical Laboratory Experiment*, 3rd Nat. Ass.: Syst. Auto. Eng. Labour and Sci. Res. SAITNI-89, B.G. Albena, 10 (1989), (in Bulgarian lang.)

[8] https://wiki.analog.com/university/courses/electronics/labs

[9] Popov G., "*Parallel Data Exchange Through Living Variables*", SAITNI-93, Albena, 1993,September (bulg. version)

[10] Nikolov L., *Operating systems: A guide for exercises and project*, Sofia, Technical University Press, 2003 (in Bulgarian lang.)

[11] Hakima Mostefaoui,.Abdelhalim Benachenhou and Abderrahmane Adda Benattia, "*Design of a low cost remote electronic laboratory suitable for low bandwidth connection*" , Computer Applications in Engineering Education Volume 25, Issue 3, pages 480–488, May 2017

[12] Popov G., Krastev F., *Universal language extension for conducting laboratory experiments on analogue and digital circuit design*, WSEAS, Crete, July 14-17, 2017

[13] https://www.abelectronics.co.uk/docs/stock/raspberrypi/adcdacpizero/adcdacpizero-schematic.pdf