Introducing Programming Languages through Data Acquisition Examples

Slavko Kocijancic, Tomaž Kušar and David Rihtaršič
University of Ljubljana, Slovenia

***Abstract*—Motivating students at low and upper secondary schools to learn programming languages is not an easy task for a teacher. Namely, programming is not and easy job to do and the results of first "programs" seem to be very poor compared even to free software available on the web. So why learn programming languages? To overcome the trouble of motivation in teaching programming languages, we have developed a course to introduce Delphi, Visual basic, C++, LabView, … based on practical examples supported by self developed, low-cost data acquisition modules. In-service teachers at secondary schools who attended the workshops accepted this concept positively, most of them reported about positive change of students' attitude regarding programming classes.**

***Index Terms*—Data acquisition, engineering education, programming languages, robotics.**

I. INTRODUCTION

About 25 years ago so called "microcomputers" became available for all levels of education according to their low-price and decent functionality for these times. ZX spectrum, Commodore, etc, were well accepted by kids from 10 years on. Programming language such as Basic was a part of the microcomputer having a role of operating system. However, learning of programming that was rather popular at the beginning of the IT era gradually lost its attractiveness. Characteristics of nowadays personal computers are incomparably better; efficient software support is available on commercial basis as well as on freeware and shareware platform through internet.

Courses on programming languages hardy follow rapid development of ICT especially regarding motivation of students [1, 2, 3]. Even if the foundation of programming languages such as Basic, C, C++, Pascal…, did not change much, traditional exercises used to train programming skills became rather obsolete. Web based applications can definitely motivate students to learn computer programming. However, web development environments (based on JavaScript for example) are rather specific and complex.

General programming environments such as Visual Basic, Visual C++, Delphi, etc. may provide more transferable foundation of programming skills. In order to provide attractive and motivating tasks for students at secondary schools, we developed a course based on the use of a data acquisition and control module. The data acquisition (DAQ) modules (often called data loggers) are becoming an indispensable part of school science and technology laboratories. Products offered by established companies such as National instruments are rather costly and often do not provide "open source" software support (i.e. drivers, programming libraries). The DAQ modules designed particularly for education, for instance Vernier and Pasco DAQ modules [4, 5] emphasize final user applications. They are also expensive and mostly limited to sampling of data from sensors; output functions are not supported adequately as well as programming of the DAQ systems is limited.

Within the project named ComLab-2 [6] running under EU programme Leonardo da Vinci, low-cost DAQ modules, PC drivers, programming libraries, example applications, user software and course material were developed. In the paper, hardware characteristics of the DAQ modules and the software support are outlined. Furthermore, some example exercises are presented.

II. EPRODAS DATA ACQUISITION MODULES

Under the eProDas platform, three DAQ modules have been designed, developed and produced so far: eProDas-SC1, eProDas-SC2 and eProDas-Rob1. All DAQ modules share the same basic architecture, the pre programmed 40-pin microcontroller PIC18F4550 (produced by Microchip Company) supporting USB communication. The abbreviation SC namely means Single Chip, while Rob denotes robotics. Further technical information is available on ComLab project website [6]. There are also "do-it-yourself" instructions available at the website for those who would like to build their own appliance based on eProDas platform!

A. eProDas-SC1 DAQ module

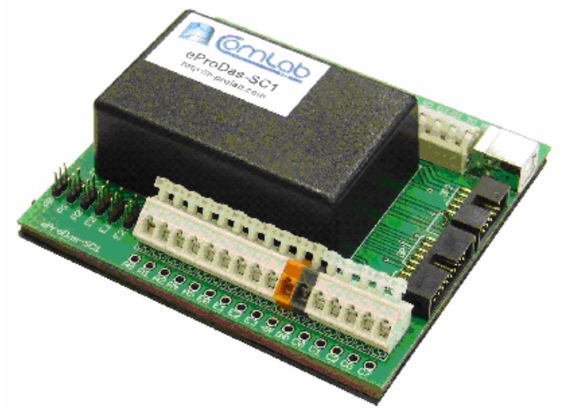


Figure 1. eProDas-SC1 module

The module (see "Fig. 1") s designed first of all for those who want to have an open and flexible hardware system and would like to develop particular applications.

Each I/O port of the microcontroller is connected with external WAGO terminal block strips with cage-clamp spring. The eProDas-SC1 has also two boxed headers connectors (10 and 16-way) with I/O pins and power supply to expand device with H-bridge control circuit [7] and six 3-pin strips to connect servo motors or sensors.

B. *eProDas-SC2 DAQ module*

The module (see "Fig. 2.") is designed for straightforward connection of sensors often practiced in biology, chemistry, physic and other exercises. Four 6-way DIN sockets are compatible with the Vernier's analogue sensors DIN type, or self-made analogue sensors. The same channels can be used also as digital input or output. Device also has two phone 6-way sockets for digital input/output which are compatible with Vernier's Sonic distance sensor MD-ULI (Universal Lab Interface). Moreover, both DAQ systems can be expanded with other circuits through serial communications, either SPI or UART.

These USB-powered, plug-and-play DAQ modules are ideal for portable and benchtop applications. Both are also supported by end-user freely software application named eProLab. The combination of eProDas DAQ hardware and easy-to-use eProLab software offers a high performance, flexible solution for computerized measurements, real-time data display, particularly convenient for science and technology education.

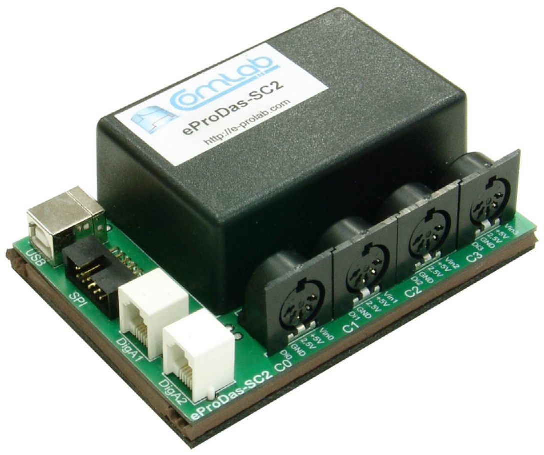


Figure 2. eProDas-SC2 module

C. *eProDas-Rob1 module*

Robotics can have good potential for motivation of young people to study technical disciplines. Robots got its role in films, computer games; they are becoming a part of everyday life through sophisticated devices being introduced even for housekeeping works. In K-12 schools (students age up to 18 years), the role of robotics could be to integrate science and technology curriculum [8, 9, 10].

The eProDas-Rob1 is built around ATmega16 microcontroller (Atmel's AVR family, see "Fig. 3."). The module tends to offer multi-purpose functionality for a low cost. The idea of the module is not to follow the design of well-established robotics kits such as Lego [11] and FischerTechnik [12] products but to offer an open system concerning connection of external gadgets, programming environments as well as regarding areas of applications. One can therefore connect third-party sensors, drives, circuits, motors; use various programming environments, etc.

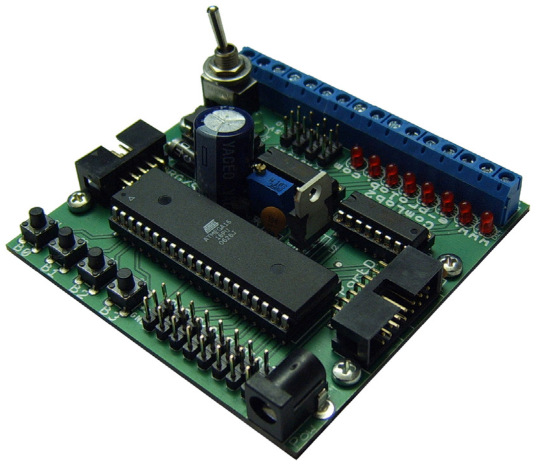


Figure 3. eProDas-Rob1 control module.

Characteristics of the eProDas-Rob1 module are versatile. There are 8 digital high-current out puts, 4 on-board keys, 16 TTL compatible digital outputs, up to 13-channel 10-bit analogue inputs and 20-bit TTL compatible digital inputs. Commonly available connectors provide effortless connection of external devices ("Fig. 4.") The module can be directly connected to the USB using eProDas platform. The module can be powered by a set of 6 AA batteries or via AC/DC adapter. Additional technical information with a complete datasheet is available on the web [13].

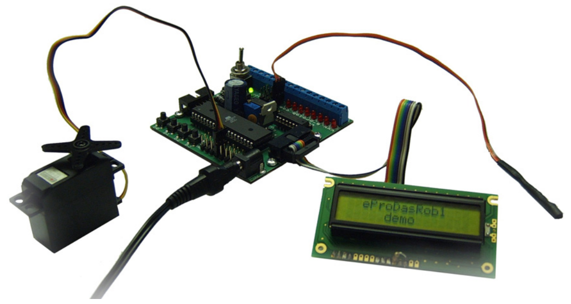


Figure 4. Examples of connecting different external gadgets.

Regarding the connection with the PC, there are two modes of operation, "on-line" and "off-line".

In "on-line" mode, the module gets commands from the PC and sends data back continuously; it is based on permanent connection with a PC through a parallel port or one of the USB modules (eProDas-SC1/SC2/SPI).

However, there are applications where permanent connection to a PC may be unfavorable. For example, sometimes a control module is required to have small size or is note near to the PCs, or it needs to be mounted on a small mobile device. Furthermore, PC is too expensive and unreliable to be in charge to control certain devices; imagine "intelligent garage door" being controlled by a PC… For those cases, "off-line" (or autonomous) mode is preferable. In "off-line" mode, the module operates as in-circuit programmable unit; after uploading a user program, the module can be unplugged from the PC and operates autonomously. Any compiler that supports

ATMega16 microcontroller and generates hex file can be used to develop off-line application.

III. BASIC CONCEPTS OF PROGRAMMING

One more featuring attribute of eProDas DAQ modules is free software, which is the accompanying part of whole eProDas platform. Software includes open source firmware for the PIC, Windows drivers (supporting Windows Vista as well!) and two dynamic linked libraries which contain a number of functions related to pre-programmed microcontroller.

The first library named eProDas.dll consists of low-level (LL) functions. All the functions are described in a comprehensive datasheet entitled eProDas Data acquisition system – Users' guide and programming manual [14]. The LL functions are rather complex, have long names with a number of parameters and are not so straightforward. But for experienced programmer, the LL functions provide all the flexibility that is possible. However, to simplify the use of LL eProDas functions, the so called High Level (HL) functions have been developed. They are essentially derived from the LL functions and implemented in dynamic link library named eProDasHL.dll. The number of HL functions is smaller; they have shorter names with less parameters, but still support the use of eProDas platform for most common I/O tasks. To make the functions in eProDasHL.dll easy accessible from a particular programming environment, two programming libraries have been developed so far. eProDasVBLib.vb for Microsoft Visual Basic and eProDasDBLib.pas for Borland Delphi (supports free “Turbo” version as well). To follow current trends emphasizing flow-chart “iconic” programming, the eProDas VI (virtual instruments) functions (or graphical icons) for LabVIEW programming environment have been developed too.

IV. EXAMPLE APPLICATIONS

Before starting to develop programs implementing the eProDas modules, one first needs to install the drivers, eProDas.dll and eProDasHL.dll, as well as to download the programming libraries from the ComLab website. In order to illustrate the development of a program in different environments, we present one rather straightforward example. Imagine we want to build a model of street lighting. When getting dark, the street lights should turn on, in the morning, they should turn off. In the evening as well as in the morning we would like to avoid multiple switching due to noise, small perturbations of light intensity caused by clouds etc. So the principle of hysteresis is introduced for this on/off automatic control system. The task is therefore to have adjustable value for turn-on/off the lamp as well as the hysteresis in illumination control; the lamp should be switched on at lower environment light in the evening as it is switched off in the morning. The circuit for the described model is simple: light intensity is detected through a light dependent resistor wired in voltage divider and connected to the eProDas analogue input pin. The lamp is switched on and off through digital output through a basic transistor-switch circuit. (“Fig. 5.”).

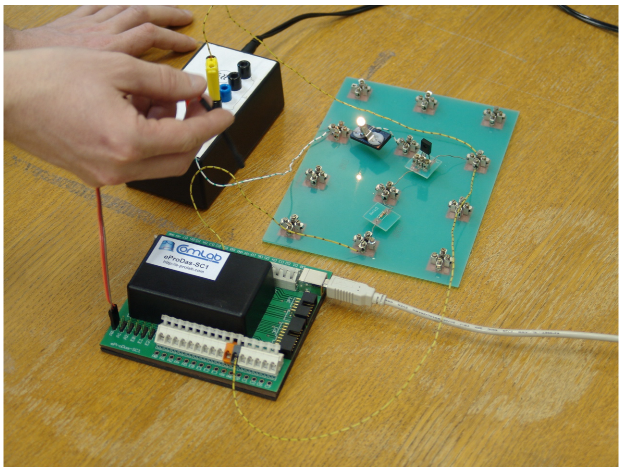


Figure 5. Experiment set-up for light control.

A. *Borland Turbo Delphi example*

Borland Turbo Delphi is available free of charge and features complete functionality required for programming in Pascal language.

The main task of the program is to display current value sampled from the light sensor as 10-bit number from 10-bit ADC and to provide two controls. The first is edge value of ambient light intensity which indicates whether the lamp is supposed to be switched on or off. At the form on computer screen (“Fig. 6.”) it is set as “Adjusting value”. The second control is hysteresis, which determines the difference between the light intensity level for switching on the lamp while darkening and for switching off the lamp when the ambient light is increasing.

When creating a new program in Turbo Delphi, the procedure is the following. At the beginning we have to add the programming library (eProDasDBLib.pas) to the project. Next step is to establish the communication between the DAQ module and the PC using a function named eProDasInit which is normally invoked just after the program is opened. The function not only enables communication, but through the returned value reports about the type of the eProDas module connected to USB of the PC (say is it eProDas-SC1, eProDas-SC2, eProDas-Rob1, etc)! Because the eProDas devices differ in some characteristics and functionality, all the functions may not be appropriate for all DAQ devices.

Besides eProDasInit function at the beginning and eProDasClose function called at the end, the code for light controlled example is invoked within periodically repeated timer procedure as follows:

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
MinValue:= midvalue -  trunc((hysteresis/2));
Label5.Caption:= inttostr(MinValue);
MaxValue:= midvalue + trunc((Hysteresis/2));
label1.Caption:= inttostr(MaxValue);

if PreviousValue < MinValue then
    begin
      DOutBit(PortB,0,1);
    end;
  if PreviousValue > MaxValue then
    begin
```

```
        DOutBit(PortB,0,0);
      end;
  CurrentValue:= AIn10Bnum(adcno0);
  PreviousValue:= CurrentValue;
  Label3.Caption:=inttostr(currentvalue);
  end;
```

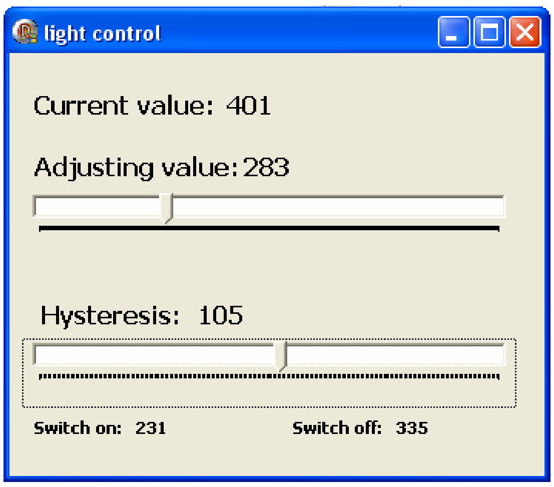


Figure 6. Experiment set-up for light control.

B. *LabView example*

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a popular, powerful and flexible graphical development environment [15]. Programming in this environment is essentially different compared to text-based programming languages such as Visual Basic and Delphi. In LabVIEW, the graphical icons are used to construct the programs (Fig. 7.).

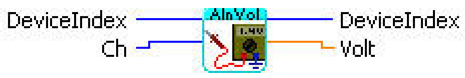


Figure 7. VI icon for 10-bit analogue input function.

For those who are familiar in LabVIEW environment, implementation of the eProDas system should be recognizable. All eProDasHL functions are supported by icons. Finally, the functions' parameters are the same as in previously described development environment!

Specific for LabVIEW are ''Wire data paths'' used to link icons together. The execution of the program is controlled by the data flow and not by a linear execution of lines of code. The concept is known as data flow programming. For this reason, each icon has one additional parameter which is to be used just to connect icons together and is very useful, when manipulation of more then one function at the same time is required.

With this principle in mind, the programming is rather clear. "Fig. 7" represents the VI block diagram we use to read value from the analogue input. The parameter 'Ch' beside the icon AInVol the channel of the analogue input, say 0. Of course, we also have to start with eProDasInit and finish with eProDasClose functions! The program screen presentation is presented in "Fig. 8.", its chart flow is in "Fig. 9."

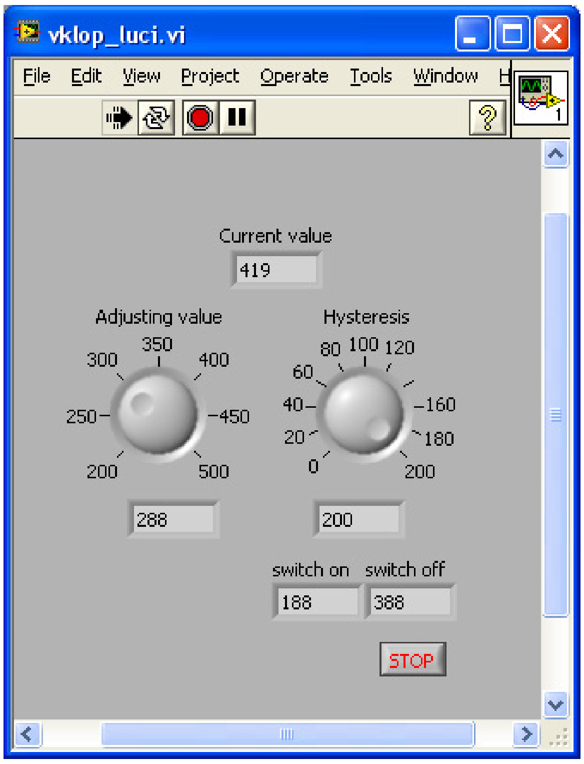


Figure 8. Screen form made in LabView.

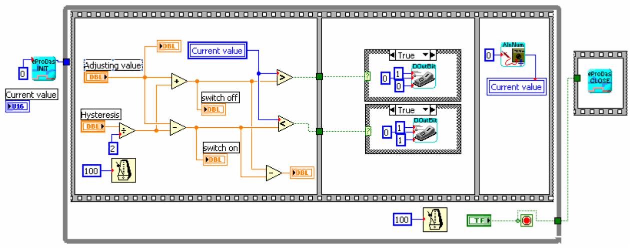
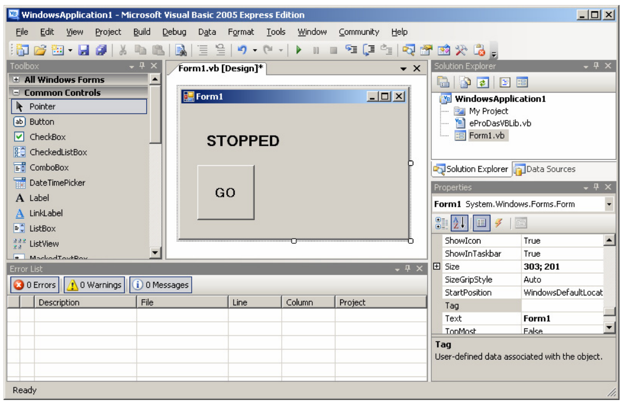


Figure 9. Chart-flow in LabView.

C. *Visual Basic example*

Visual Basic Express Edition is also a free programming development tool [16]. Similarly as in Delphi, programming in Visual Basic is supported by a program module named eProDasVBlib.vb. Solving the task in Visual Basic IDE (Integrated Development Environment) is shown in "Fig. 10.", followed by a code similar to the Delphi example.

```
  DOutDir(Port.C, 4, 5, DirectX.Plus)
  DOutDir(Port.C, 7, 6, DirectX.Plus)
  ACT_label.Text = "Going forward"
    While   (DInBit(Port.A,   5)   =   0)   And
(DInBit(Port.A, 4) = 0)
      Application.DoEvents()
    End While
  DOutDir(Port.C, 4, 5, DirectX.Off)
  DOutDir(Port.C, 7, 6, DirectX.Off)
  ACT_label.Text = "Waiting for 2s"
  Wait(2)
  DOutDir(Port.C, 4, 5, DirectX.Minus)
  DOutDir(Port.C, 7, 6, DirectX.Minus)
  ACT_label.Text = "Going backward for 1s"
  Wait(1)
  DOutDir(Port.C, 4, 5, DirectX.Off)
  DOutDir(Port.C, 7, 6, DirectX.Off)
  ACT_label.Text = "STOPPED"
```



Form1 System.Windows.Forms.Form

ShowIcon	True
ShowInTaskbar	True
Size	303; 201
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocat
Tag	
Text	Form1
TopMost	False

Tag
User-defined data associated with the object.

Figure 10. Screen capture of Visual Basic programming environment.

D. *BASCOM example*

Above examples operate in on-line mode, but eProDas-Rob1 also supports off-line (or autonomous) mode.

In the following example, the eProDas-Rob1 module is mounted on a mobile robot kit named eProKit-MR1. The kit has two wheels each of them being driven by a DC motor connected to high-current digital inputs (two bits for each motor). Two "tentacles" in the front of the robot are having a function of switches connected to digital inputs. The robot could be also extended by other sensors, servo-motors, LCD display, etc.

The task of the exercise is as follows. The mobile robot ("Fig. 11") starts driving forward and stops when it hits the obstacle with at least one of the two tentacles. Then it waits for two seconds, drives backward for one second and stops again. While the program is running, the action that was taken by the robot is displayed at the at the LCD.

The Bascom code is the following:

```
  PortC.7 = 1         'left wheel going forward
  PortC.4 = 1         'right wheel going forward
  Cls
  Lcd "Going forward"
  While Switches = 0
    Switches = Pina.5 Or Pina.4
  Wend
  PortC = 0            'turn all motors off
  Cls
  Lcd "Waiting for 2s"
  Wait 2
  PortC.6 = 1         'left wheel going backward
  PortC.5 = 1         'right wheel going
backward
  Cls
  Lcd "Going back for 1s"
  Wait 1
  Portc = 0
  Cls
  Lcd "STOPPED"
```

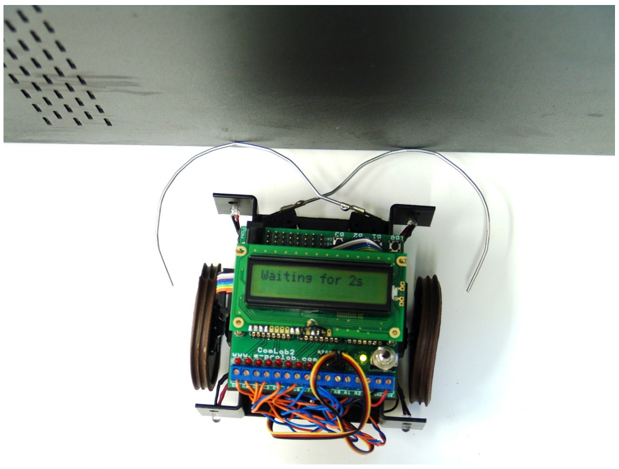


Figure 11. Mobile robot working in an autonomous mode, hitting the obstacle.

V. CONCLUSIONS

At the Department of physic and technology, Faculty of Education in Ljubljana, a lot of efforts have been given in the previous years to improve skills and knowledge of trainee teachers at the field of informatics, electrical engineering, computerized science laboratory, etc. Furthermore, in order to introduce our work to in-service teachers, we organized a couple of workshops for them too ("Fig. 12"). Participants were programming in Visual Basic, Bascom and LabVIEW environments. At the same time, eProDas DAQ platform and modules were tested. The workshops were based on pre-prepared program examples and participants were asked to explore and modify the code. Their answers to the questionnaire delivered to the teachers after the workshops were undeniably positive and some of them later reported about encouraging respond of their students.

To summarize the conclusions:

- • Low cost of the set of the DAQ modules is crucial for educational purposes.
- • Slovenian in-service teachers of science and technology that are willing to introduce DAQ example applications to their students can be trained to implement the hardware, software and courseware in two days (16 hours).
- • All other, not so enthusiastic pre-service teachers, can be trained to use the platform in routine way in a short time.
- • In general, freely available textual programming environments can compete with iconic-based programming tools.
- • eProDas modules are reliable, wide-ranging and easy to use; they offer project oriented work covering wide range of applications provoking creativity and interdisciplinary approach, what seems to be the most important advantage compared to established options.

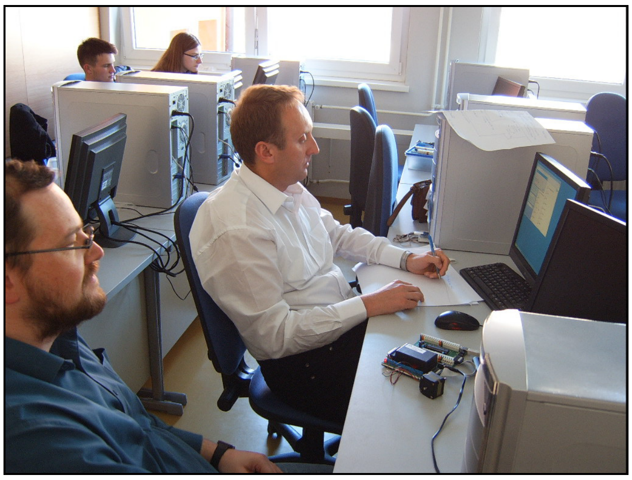


Figure 12. Workshop for in-service teachers.

ACKNOWLEDGMENT

Support received under EU Leonardo Da Vinci Community Vocational Training Action Program, project SI-05-B-F-PP-176008 (http://e-prolab.com/comlab) and Slovenian Ministry of Education and Sport is gratefully acknowledged.

REFERENCES

[1] M.A. Perez-Quinones, "Teaching history of programming languages to undergraduate students", *Frontiers in Education Conference*, vol. 1, p: 301-306 vol.1, 1998

[2] R. B. Dewar, E. Schonberg, "Computer Science Education: Where Are the Software Engineers of Tomorrow?", The Journal of Defense Software Engineering, Januray 2008 (http://www.stsc.hill.af.mil/CrossTalk/2008/01/0801DewarSchonberg.html)

[3] W.K. Chen, Y. C. Cheng , "Teaching Object-Oriented Programming Laboratory With Computer Game Programming", *IEEE Transactions in Education*, vol. 50 (3), p. 197-203, 2007 (doi:10.1109/TE.2007.900026)

[4] *Vernier Software and Technology*, http://www.vernier.com, availability verified in January 2008..

[5] *Pasco*, http://www.pasco.com, availability verified in January 2008.

[6] eProLab project, http://e-prolab.com/comlab, availability verified in January 2008.

[7] http://e-prolab.com/en/eqsens/eqsens.html, availability verified in January 2008.

[8] E. Kolberg, N. Orlev, "Robotics learning as a tool for integrating science-technology curriculum in K-12 schools", *Frontiers in Education Conference*. 31st Annual vol. 1, Issue , 2001, p. T2E - 12-13vol.1, 2001

[9] M. Robinson, "Robotics-driven activities: Can they improve middle school science learning?" *Bulletin of Science, Technology & Society*, vol. 25(1), p. 73-84, 2005. (doi:10.1177/0270467604271244)

[10] R. D. Beer, H. J. Chiel and R. F. Drushel, "Using robotics to teach science and engineering", *Communications of the ACM*, vol. 42(6), p. 85-92, 1999. (doi:10.1145/303849.303866)

[11] http://www.lego.com, visited in February 2008

[12] http://www.fischertechnik.de, visited in February 2008

[13] http://e-prolab.com/en/eqdaq/rob1/rob1.html, visited in February 2008

[14] http://e-prolab.com/en/eqdaq/eprodas_plat/eprodasplat.html, availability verified in January 2008.

[15] http://www.ni.com/labview, availability verified in January 2008.

[16] http://msdn2.microsoft.com/en-us/express/default.aspx, visited in February 2008

AUTHORS

S. Kocijancic is with the University of Ljubljana, Faculty of Education, Kardeljeva ploscad 16, SI-1000 Ljubljana, Slovenia (e-mail: slavko.kocijancic@pef.uni-lj.si).

T. Kušar, is with the University of Ljubljana, Faculty of Education, Kardeljeva ploscad 16, SI-1000 Ljubljana, Slovenia (e-mail: tomaz.kusar@guest.arnes.si).

D. Rihtaršič is with the University of Ljubljana, Faculty of Education, Kardeljeva ploscad 16, SI-1000 Ljubljana, Slovenia (e-mail: david.rihtarsic@pef.uni-lj.si).

This article was modified from a presentation at the 31st International Convention MIPRO 2008 in Opatija, Croatia, May 2008. Manuscript received 29 July 2008. Published as submitted by the author(s).