An Integrated Course in Programming for Laboratory and Process Control

Dr. Warren A. Rosen, Drexel University (Eng. & Eng. Tech.)

Dr. Warren Rosen received his Ph.D. in physics from Temple University. He has served as Assistant Professor of Physics at Colby and Vassar Colleges where he carried out research in solar physics, medical physics, and instrumentation. Following this experience he was a research scientist at the Naval Air Warfare Center in Warminster, PA where he established a laboratory for research in high-performance computer networks and architectures for mission avionics and signal processing systems, and served as the Navy's representative on several national and international standards committees. In 1997 joined the staff of Drexel University, first as a research professor in the Electrical And Computer Engineering Department and later as a clinical assistant professor in the Department of Engineering Technology. Also in 1997, Dr. Rosen founded Rydal Research and Development, Inc., which has carried out research in networking devices and protocols for the Air Force Office of Scientific Research and the Office of Naval Research. Dr. Rosen is the author or co-author of over 80 publications and conference proceedings and the holder of six U.S. patents in computer networking and signal processing.

Dr. Irina Nicoleta Ciobanescu Husanu, Drexel University (Tech.)

Irina Ciobanescu Husanu, Ph. D. is Assistant Clinical Professor with Drexel University, Engineering Technology program. Her area of expertise is in thermo-fluid sciences with applications in micro-combustion, fuel cells, green fuels and plasma assisted combustion. She has prior industrial experience in aerospace engineering that encompasses both theoretical analysis and experimental investigations such as designing and testing of propulsion systems including design and development of pilot testing facility, mechanical instrumentation, and industrial applications of aircraft engines. Also, in the past 10 years she gained experience in teaching ME and ET courses in both quality control and quality assurance areas as well as in thermal-fluid, energy conversion and mechanical areas from various levels of instruction and addressed to a broad spectrum of students, from freshmen to seniors, from high school graduates to adult learners. She also has extended experience in curriculum development. Dr Husanu developed laboratory activities for Measurement and Instrumentation course as well as for quality control undergraduate and graduate courses in ET Masters program. Also, she introduced the first experiential activity for Applied Mechanics courses. She is coordinator and advisor for capstone projects for Engineering Technology.

Mr. M. Eric Carr, Drexel University

Mr. Eric Carr is a full-time Laboratory Manager and part-time adjunct instructor with Drexel University's Department of Engineering Technology. Eric assists faculty members with the development and implementation of various Engineering Technology courses. A graduate of Old Dominion University's Computer Engineering Technology program and Drexel's College of Engineering, Eric enjoys finding innovative ways to use microcontrollers and other technologies to enhance Drexel's Engineering Technology course offerings. Eric is currently pursuing a Ph.D in Computer Engineering at Drexel, and is an author of several technical papers in the field of Engineering Technology Education.

An Integrated Course in Programming for Laboratory and Process Control

Introduction

The Engineering Technology program of Drexel University emphasizes a holistic approach to programming for laboratory and process control. In this approach we address not only the basics of programming languages but also considerations relating to implementation such as cost, reliability, upgradeability, and maintainability as well as hardware issues such as power consumption and form factor. The approach spans several courses so that common problems are addressed using, e.g., microcontrollers, microprocessors, or programmable logic controllers, so that students acquire a feeling for the different uses and application space of each technology. In this paper we describe how we extend this approach to the basic first-year course in programming.

Students today are confronted with a wide range of programming languages and development tools. Typically these are offered as isolated courses or learning modules, with little attempt to provide them with the understanding needed to select the best tool for the job at hand. This paper describes an integrated course in programming techniques for controlling laboratory experiments and industrial processes. The course is intended to provide first-year engineering technology students with a broad overview of available programming technologies. To do this, two programming tools were chosen—the C programming language and LabVIEW. The first half of the course is dedicated to C programming. Topics include a brief introduction to computers and programming, I/O, data types, expressions and assignments, relational operators, loops and branching, functions, and arrays. This part of the course is taught in a computer laboratory so that each student has an individual computer. Every lecture includes simple exercises (~ 5 minutes each) that are preformed by the students in real time as the relevant topic is covered. Most lectures are followed by a laboratory exercise. The laboratory exercises are based on the Arduino microcontroller platform [1]. Code::Blocks [2] is used as the development environment. The advantage of using Code::Blocks is that it supports programming in both standard C and Arduino C

The LabVIEW part of the course is dedicated to enhancing students' understanding of monitoring and control processes. Using LabVIEW, students are required to create virtual experimental setups as well as create monitoring and control interfaces for physical experiments.

A key feature of the approach is the seamless integration of the C and LABVIEW laboratory activities in such a manner that students get to practice their programming skills in both languages using similar physical experiments, as described below. Students learn by comparison various basic and intermediate programming skills, learning the advantages and disadvantages of both text-based and graphical coding. In addition to the academic goals of these activities, students gain tremendous hands-on experience using real-life examples.

The course is a required 3-credit course for the Engineering Technology degree that integrates both traditional in-class lecture and laboratory activities. Most students enter the course with little or no formal training in either C or LabVIEW. Expected outcomes include the ability of the

students to explain and use basic I/O operations, data types, assignments, functions, and simple control statements to develop C and LabVIEW programs for industrial and laboratory applications, and to use industry-standard development environments. Multiple forms of assessment were used to demonstrate success, including student surveys, course exams, and homework.

Course Description

Table 1 shows the 10-week course schedule. The first five weeks are dedicated to C programming. The first week's lecture provides a brief introduction to how a computer executes a program and what a computer program is. This is followed by introduction to basic C programming, including data types, expressions, assignments, I/O, and mathematics libraries. In the second week relational operators and *while* loops are covered. The next two weeks cover branching and functions and arrays. A comprehensive midterm exam is given in the fifth week, assessing the students' knowledge of C programming, and a final project is assigned to be developed by week 11 as a assessment for the LABVIEW sequence..

Week	Topic
1	Introduction to programming, I/O, data types, expressions and assignments
2	Relational operators, *while* loops
3	Branching
4	Functions and Arrays
5	Midterm Exam
6	LabVIEW programming principles; LabVIEW environment
7	Data types, software constructs, and structures in LabVIEW Graphical User Interface (GUI) elements
8	LabVIEW variables and functions; Objects. Simple design patterns. Quiz 1 (in class)
9	SubVI (Virtual Instrument) design, VI design and documentation
10	Strings and File I/O, Error handling, Debugging tools and techniques, State Machines
11	Final Exam (Quiz 2 – take-home project)

Table 1. Course schedule.

The course is offered in a 28-seat computer laboratory in which each student has access to an individual computer and experimental hardware setup. Every lecture includes simple exercises

(~ 5 minutes each) that are preformed by the students in real time as the relevant topic is covered. For example, the first exercise involves opening the development tool and writing C code to print the message, “Hello World” to the screen. Students then have to figure out how to modify the code to print variations in text, spacing, etc. The lectures in weeks 2 through 4 are followed by more detailed laboratory exercises. These labs use the Arduino Uno microcontroller board. Arduino modules used in this way have been shown to enhance students learning in C/C++ courses [3].

In the first lab exercise the students use a custom built three-color LED board, shown in Figure 1. The board is designed to plug directly into the Arduino board in such a way that it connects to ground and three I/O pins. Before mounting the board the students use jumper wires to connect sequentially each of the internal R, G, and B LEDs to the Arduino’s 5V power supply so that they understand how the board is wired. Next, they move the jumper wire to one of the I/O pins and develop and run the code to make a 1-second LED flasher using the Code::Blocks development tool. Code::Blocks was chosen as the Integrated Design Environment because it supports programming in both standard C and Arduino C. This means that the students do not need to use a separate tool (e.g., Arduino IDE) to program the microcontroller.

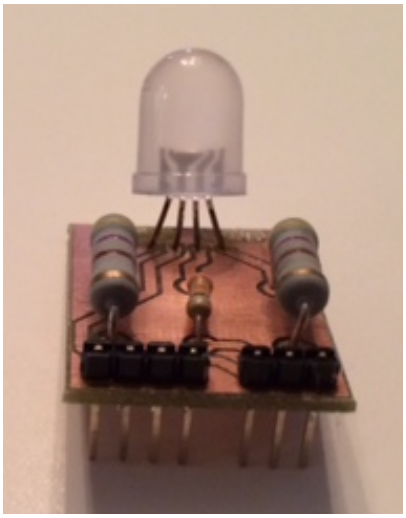

Figure 1. Three-color LED mini-shield.

When students open Code::Blocks and select, “Create a new project,” they’re presented with a choice of project options, including an Arduino Project or Console application, as shown in Figure 2. The student uses Console application for conventional C programming during the lectures or for homework. Code::Blocks is available as freeware, although the version with Arduino IDE is only available in the Windows build.

Once the students have the single flasher code working, they are instructed to plug the LED board into the Arduino board. They are then challenged to develop the code to implement an “Arduino Traffic Light,” that sequentially turns on Green for 30 seconds, then yellow for 15 seconds (yellow is produced by turning on both Red and Green LED elements), then red for 30 seconds.

In the second lab exercise students use an ultrasonic rangefinder module [4] with the Arduino microcontroller board to determine the distance to nearby objects and to generate a warning when collisions are imminent. The rangefinder module comprises an acoustic transmitter that, on command from the microcontroller, transmits a burst of 40 KHz sound, and a receiver that detects returning echoes of the signal when it reflects from nearby surfaces. The rangefinder module then notifies the microcontroller that the echo has been detected. Students are given prewritten code that measures the time interval between transmission and detection and returns the distance to the reflecting object. Once the students have this code running and generating distances, they use the three-color LED board from the previous lab to develop a proximity warning system. The rangefinder is shown in Figure 3.

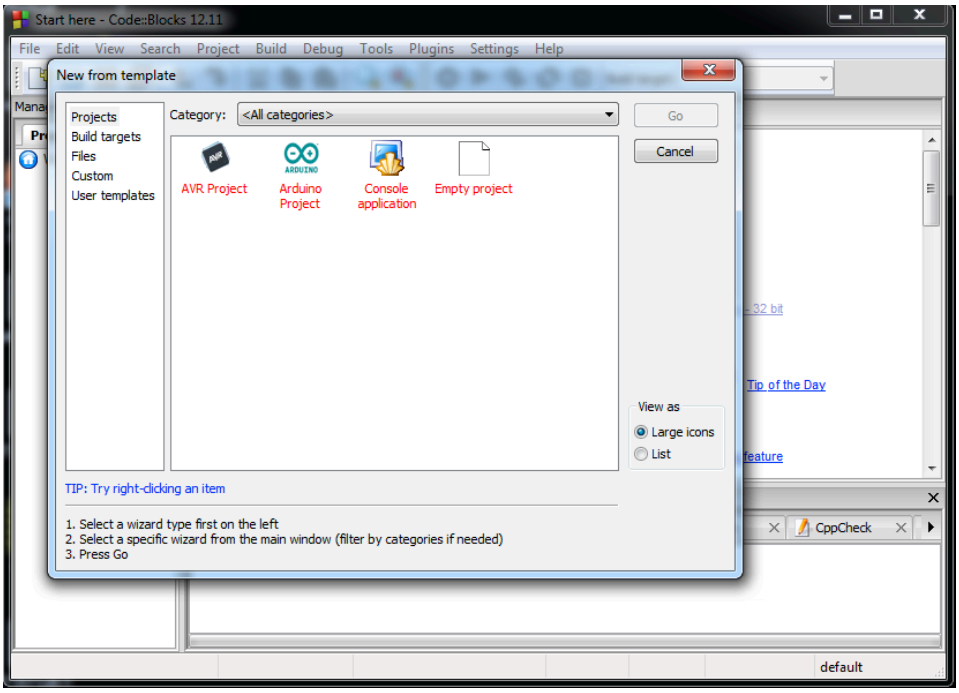

Figure 2. Code::Blocks project selection window.

Figure 3. Rangefinder module. The four pins at the bottom are, from left to right, Power (5 V), Trigger, Echo, and Ground.

In the third laboratory exercise (corresponding to the lecture on functions and arrays) students learn to control a servo motor. They begin by using the Arduino Servo library to calibrate the servo motor that they are given. They then develop the code to implement a 5-position indicator

such as one might find in an old building elevator dial. For this last exercise they don't use the Servo library, but instead develop their own code using an array of delay times.

During the LABVIEW portion of the course, besides acquiring basic graphical programming skills, students are encouraged to think creatively about how to implement on a virtual level a physical experimental setup and how to develop a monitoring and control interface using loops and structures, culminating with developing simple state machines. During this half of the course, students are led—step-by-step at first, then in increasingly high-level terms—through the creation of several LabVIEW VIs of increasing complexity. While other activities are also conducted during the course as in-class or homework assignments, in this paper we are presenting only those that are implemented as common C and LABVIEW programming during the course.

First, the students are introduced to the concepts of controls and indicators (corresponding to traditional inputs and outputs, or data "sources" and "sinks." After a few preliminary exercises, the students write a VI using a *While* loop and an elapsed time control to create a blinking light, as shown in **Figure 4**.

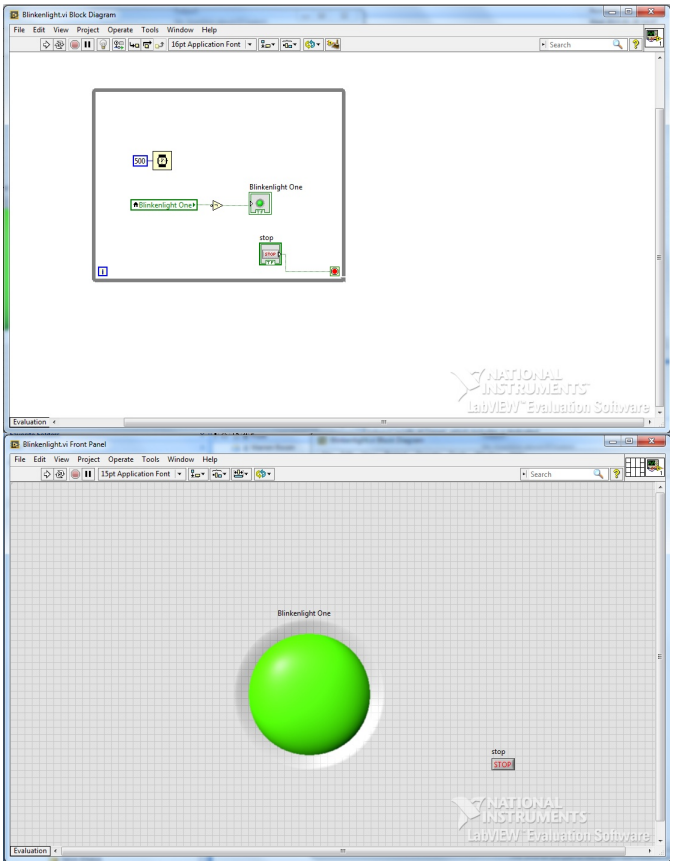

Figure 4. Front panel and block diagram for a blinking LED.

After the students complete the blinking-light VI, the idea of *FOR* loops is introduced. The elapsed-time control is retained, but rather than controlling the time spent in a simple *WHILE* loop, the delay now controls the amount of time spent in each cycle of the *For* loop. The iteration

loop counter is then used, along with a series of comparators, to light up a green, amber, or red control based on the current iteration count of the For loop. This results in a "traffic light" display, showing green, then amber, then red lights, shown in Figure 5. An outer *While* loop ensures that the cycle continues until the program is stopped. The idea of different actions based on different values of a variable can lead into a discussion of state sachines (although such a timer-based device is a very crude state machine.)

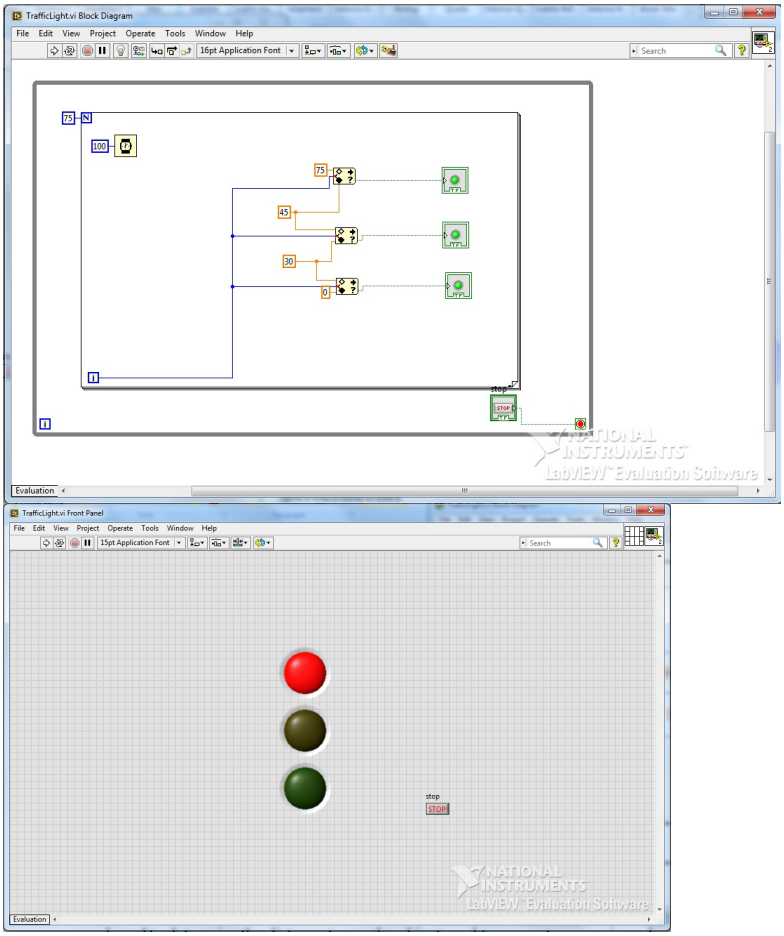

Figure 5. Traffic light VI.

Finally, the idea of local variables is expanded on, and the students build an elevator VI, complete with Up and Down buttons and a dial. A local integer variable is used to store the current floor number. The dial indicator representing this variable is updated from either itself (if no buttons are pressed), itself decremented (if the Down button is pressed), or itself incremented (if the Up button is pressed.) Debounced, single-shot action is selected on the button inputs, and the output is coerced to the range 1 through 6, thereby preventing passengers from using the elevator to explore outer space or any mineshafts.

It should be noted that each student could actually design his/her own unique LabView VI, since each experimental setup may be devised as an open-ended design. In this way students gain

knowledge about designing an experiment at physical and virtual level that should answer certain pre-designed requirements.

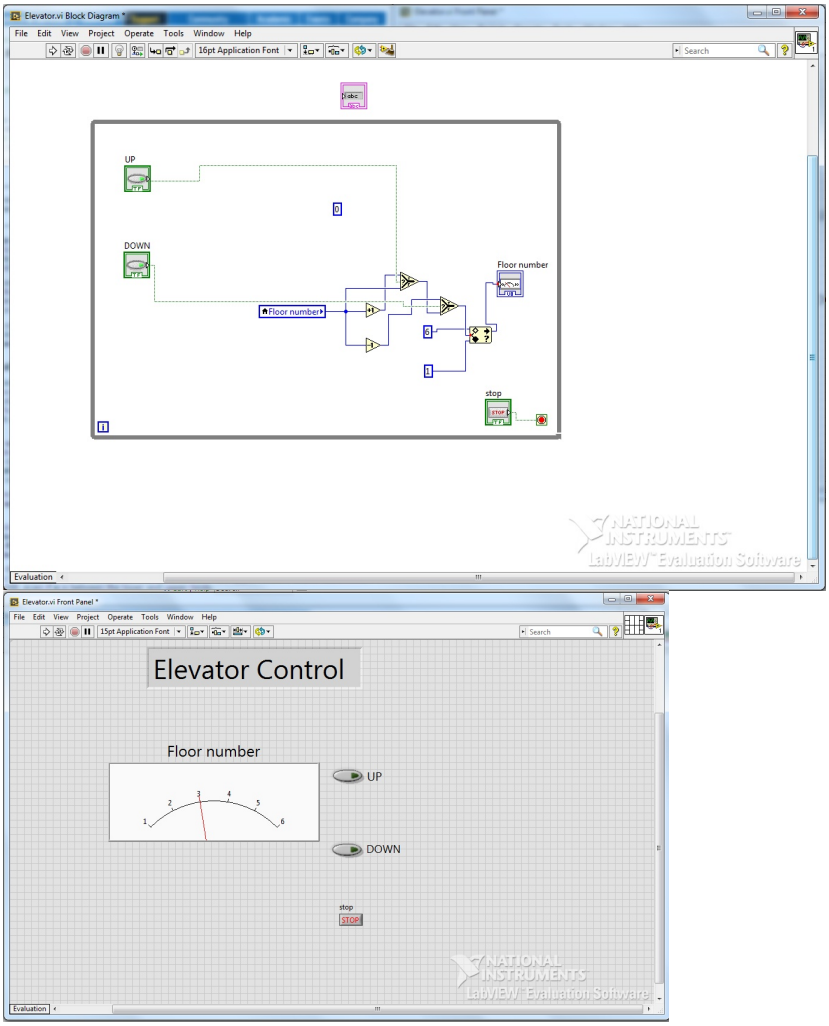

Figure 6. Elevator VI.

Results

The course, with revised laboratory exercises, was offered in the spring of 2014. Twenty-four students completed the course. All but a few had no experience with either C or LabVIEW.

The efficacy of the course was assessed using homework, lab reports, and midterm and final exams. Scores for the first and last half of the term were evaluated separately. For the first half, the average scores were 80.7, 82.6, and 86.4 (100 maximum) for homework, lab reports, and the midterm. In addition, at the end of the first half of the course students were surveyed using a Likert-type scale to assess how much the course contributed to their understanding of these topics. The rubrics used were 0 = "Nothing" and 5 = "A great deal". The questions are shown in Table 2, together with average scores and standard deviations for each.

Eighteen of the 24 students responded. Average scores ranged from 3.83 (How much did this course contribute to your understanding of how computers work?) to 4.44 (How much did this course contribute to your understanding of how microcontrollers work?). The relatively large standard deviations appear to be largely due to scores from one student who strongly disliked programming. If this student's scores are removed the average for each question (other than question 2) goes up about 0.2 and σ goes down by about 0.24. Ten of the 18 responding students provided comments. These were strongly positive with the exception of the student mentioned above.

Question	Average	σ
1. How much did this course contribute to your understanding of how computers work?	3.83	1.1
2. How much did this course contribute to your understanding of how microcontrollers work?	4.44	0.62
3. How much did this course contribute to your understanding of basic input/output operations in the C programming language?	4.06	1.0
4. How much did this course contribute to your understanding of C data types and declarations	4.06	0.94
5. How much did this course contribute to your understanding of arrays in C?	3.63	1.2
6. How much did this course contribute to your understanding of basic control statements (while, if-else, for) in C?	4.06	1.39
7. How much did this course contribute to your ability to use C to perform simple control tasks?	4.39	0.92
8. How much did you enjoy this part of the course?	4.17	1.04

5 = a great deal, 0 = nothing

Table 2. Survey results for C programming.

The final assessment is based on an individual project consisting of designing a LabVIEW VI of intermediate complexity that should be used for monitoring and control of a series of physical parameters. Students scored well during the past two academic years when this course was offered, the average score for the course being 87%.

Conclusions

In this paper we described an integrated course in programming techniques for controlling laboratory experiments and industrial processes. Two programming tools were chosen—the C programming language and LabVIEW. A key feature of the course is the seamless integration of the C and LABVIEW laboratory activities in such a manner that students get to practice their programming skills in both languages using similar physical experiments, thereby learning the advantages and disadvantages of both text-based and graphical coding. The laboratory experiments are designed to provide students with commonly encountered real-world experiences.

A range of assessment tools were used to determine the efficacy of the course, including student surveys, course exams, homework, and an end-of-term project. The results indicated that this approach provided an effective learning experience.

Bibliography

1. Retrieved from http://www.arduino.cc. Accessed 15 March 2015.
2. Retrieved from http://www.codeblocks.org. Accessed 2 February 2015.
3. M. Rubio, R. Romero-Zaliz, C. Mañoso Hierro, and Á. Pérez de Madrid y Pablo, "Enhancing an introductory programming course with physical computing modules," *Electronic Proceedings of the 2014 Frontiers in Education Conference*, October 22-25, 2014, Madrid, Spain.
4. Retrieved from http://www.mpja.com/Ultrasonic-Ranging-Raspberry-Pi-Arduino-Compatible-Module/productinfo/19605%20UT. Accessed 15 March 2015.