AC 2009-2063: UTILIZING ROBOTICS IN TEACHING MICROCONTROLLER PROGRAMMING TO MANUFACTURING ENGINEERING STUDENTS

Arif Sirinterlikci, Robert Morris University

ARIF SIRINTERLIKCI is currently an Associate Professor of Engineering at Robert Morris University. He has been the Coordinator of the RMU Learning Factory and Director of Engineering Laboratories. He holds a B.S. and an M.S., both in Mechanical Engineering from Istanbul Technical University in Turkey, and a PhD in Industrial and Systems Engineering from the Ohio State University. He has conducted research and taught in mechanical, industrial, manufacturing engineering, and industrial technology fields. He has been active in ASEE (American Society for Engineering Education) and SME (Society of Manufacturing Engineers) as an officer of the Manufacturing Division and an advisor to technical communities and student chapters, respectively.

Utilizing Robotics in Teaching Microcontroller Programming to Manufacturing Engineering Students

Abstract

This study presents the effort to add microcontroller content to an industrial controls course in a manufacturing engineering program. Industrial control courses in manufacturing engineering programs typically cover Programmable Logic Controllers (PLC's). In addition to the PLC content, the author has added hard-wired and integrated-circuit (IC) based elements to his course over the past few years. The schedule now encompasses simple but effective microcontroller content. VEX Robotics Design System and its Easy C programming language were selected due to their simplicity and engineering students' background in C++ programming. Students taking this controls course are assigned a simple fixed-goal task and work in teams to accomplish it. The task has been determined as navigating a rover robot through the department's Learning Factory. In the process, they learn about microcontrollers, their integration with sensors and actuators, and utilize their programming skills in a practical control application. Moreover, by taking the course students are exposed to a variety of technologies including hard-wired logic, IC's, PLC's, Programmable Automation Controllers (PAC's), and microcontrollers giving them the power of comparing and contrasting each element's capabilities and utilization areas. This activity is also the only exercise for the manufacturing engineering majors to utilize their high-level programming knowledge and skills in an engineering problem unlike the software engineering majors within the department.

Introduction

This study focuses on teaching microcontroller programming, and associated sensory input devices and actuators to manufacturing engineering students. In his semester long course ENGR 4400 - Device Control, the author has been following a sequence that included (i) hard-wired relay logic, (ii) integrated-circuit (IC) based digital logic, and (iii) Programmable Logic Controller (PLC) based controls.

With the hard-wired logic section, students learn about practical control applications such as standard push-button motor starters, H-bridges for directional motor controls, sequential control circuits, and auto-switching back-up lighting systems as well as generation of control logic based on switches and relays[1]. An example circuit built on a relay-trainer, designed and assembled in-house is shown in Figure 1.

On the other hand, the IC section introduces Fundamentals of Digital Electronics, Binary Logic and Boolean Algebra, especially the Objective Digital Design Methodology with Karnaugh Maps[2]. Students simulate their digital control circuits in National Instruments (NI) LabView software before building them on NI ELVIS (Electronics Laboratory Virtual Instrumentation Suite) workstations. While Figure 2.a depicts a digital logic generated in NI LabView, Figure 2.b is the actual circuit built on an NI ELVIS workstation. The IC section is also populated with realistic or close to realistic cases and examples.

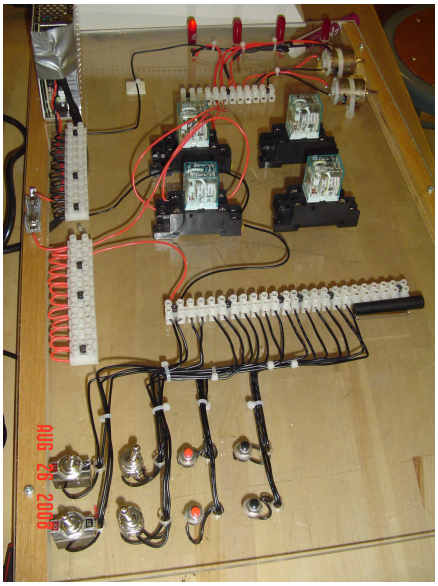


Figure 1. Hard-wired relay controls trainer

The third section, the PLC section, is the major section of the course and built on the two previous ones as students gain more understanding of the common principles for design and development of control circuits and logic programs. The PLC based content encompasses IEC 61131 Standard, basics of ladder logic programming, installation of PLC's, various field devices and their wiring into the PLC hardware, communications/networking issues and safe operation of PLC's[3]. Like the previous two sections, the PLC section is also reinforced with practical and realistic control applications such as the one shown in Figure 3. In another example, students are given the ladder logic for a rover robot of a sumo robot competition and asked to review the program. Limited content on PAC's are also included within the PLC section. Students are given background on Proportional-Integral-Derivative (PID) controls illustrated in Figure 4[4]. This is associated with a current research project on Real-Time PAC based controls.

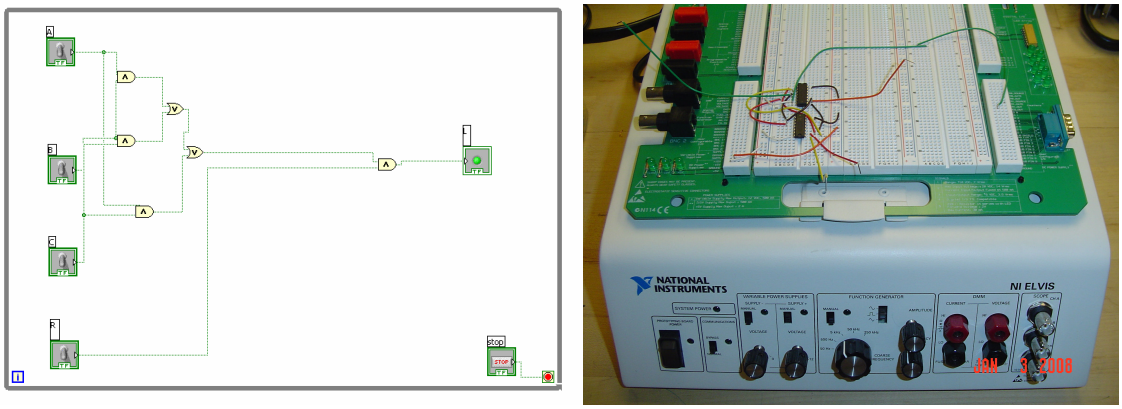


Figure 2.a NI LabView Model of a simple digital logic example .b NI ELVIS circuit for the model

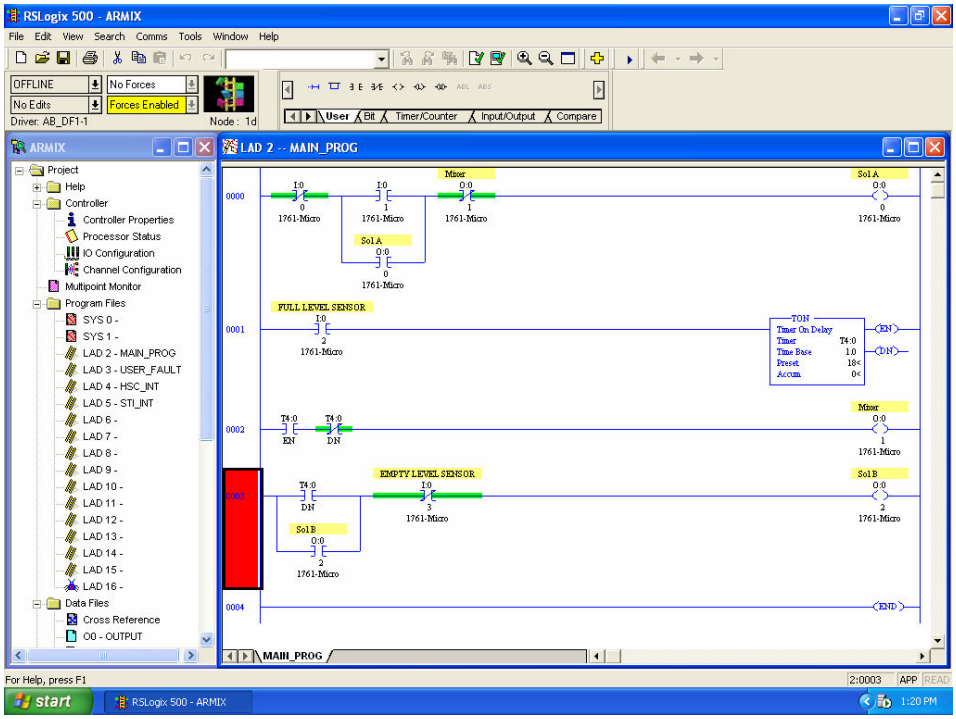


Figure 3. PLC Ladder logic for a mixer application written in RS Logix 500[3]

Up to this point, the schedule of the course includes hard-wired controls based on ladder diagrams and PLC systems programmed by ladder logic. The crucial role of ladder diagrams and ladder logic in controls has been covered in detail with the utilization of two these areas. On the contrary, the design environment for IC logic is NI LabView's block diagram based G programming. Not long ago, new NI LabView exercises were added to the course for PAC's or flexible IC's to strengthen this component as well[4].

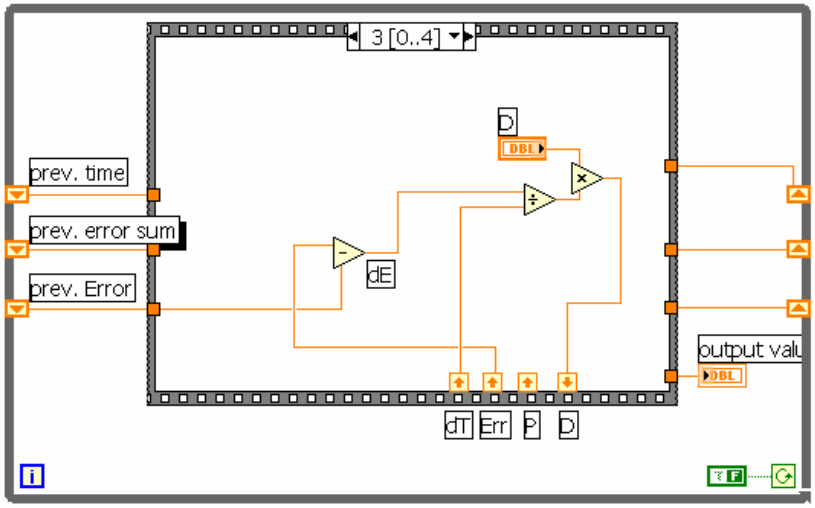


Figure 4. A NI LabView PID control system including shift registers, a WHILE loop, and event driven loop structures[4]

The original content of the course is now being further modified with the addition of microprocessor programming components. This effort is employing the VEX Robotics Design System and the Easy C programming language. VEX Robotics Design System and the Easy C were selected since engineering major takes the INFS 3184 - C++ programming course prior to taking the ENGR 4400 and the simplicity of these tools is making the learning experience easy for students.

Integration of VEX Robotics into ENGR 4400

To begin with, students are given programming kits including the programming manual and the Easy C software. Then, a presentation on the VEX components is made to students. The VEX components covered include structural, motion, electrical/electronic, radio control (RC) elements, and the microcontroller. Afterwards the class goes through the programming hardware set-up and downloading process. Hardware set-up includes the installation of the communication cables and downloader circuitry, and set-up of the COM/USB ports within the Easy C software interface. After learning about the Master-code (firmware for the microcontroller) and the default code used in clearing the microcontroller from corrupt programs, students build and download sample programs available to them. Two sample programs are included with this paper and they are shown in Figure 5 and 6[5]. The program in Figure 5 is generated by dragging programming blocks to the programming window. The Figure 6 is the Easy C syntax of the program being generated and is created automatically as the blocks are dragged into programming window. This process is similar to the LEGO programming environments for Mindstorms and NXT.

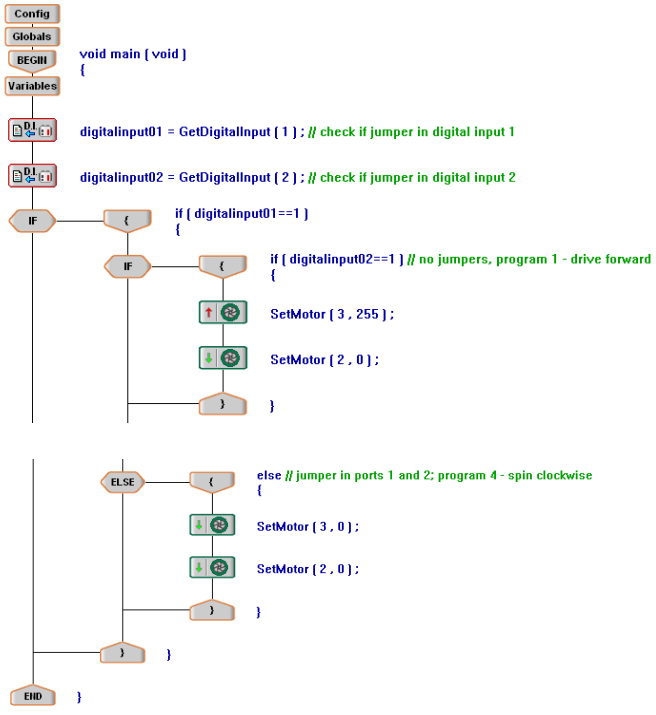


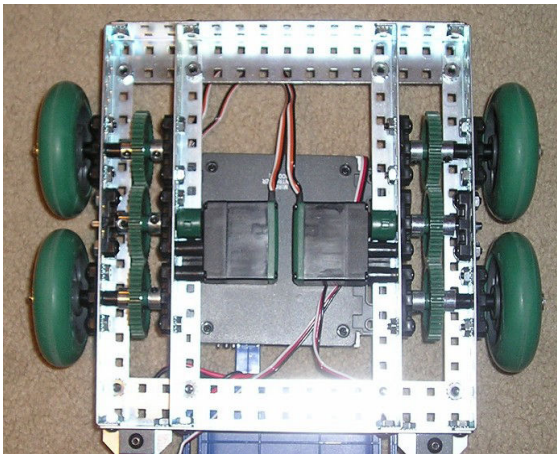
Figure 5. An example Easy C program (Block Diagram Version) for rover robot navigation[5]

```
1  #include "Main.h"
2
3  void main ( void )
4  {
5      int digitalinput01;
6      int digitalinput02;
7
8      digitalinput01 = GetDigitalInput ( 1 ) ; // check if jumper in digital input 1
9      digitalinput02 = GetDigitalInput ( 2 ) ; // check if jumper in digital input 2
10     if ( digitalinput01==1 )
11     {
12         if ( digitalinput02==1 ) // no jumpers, program 1 - drive forward
13         {
14             SetMotor ( 3 , 255 ) ;
15             SetMotor ( 2 , 0 ) ;
16         }
17         else // jumper in port 2 only; program 2- drive back
18         {
19             SetMotor ( 2 , 255 ) ;
20             SetMotor ( 3 , 0 ) ;
21         }
22     }
23     else
24     {
25         if ( digitalinput02==1 ) // jumper in port 1 only; program 3 - spin counterclockwise
26         {
27             SetMotor ( 3 , 255 ) ;
28             SetMotor ( 2 , 255 ) ;
29         }
```

Figure 6. Easy C program syntax[5]

The Assignment

Students are given a fixed-goal task to accomplish. They are presented with Square Bots based on a two-wheel robot design that is included with the original VEX Inventor's Kit manual[6]. This robot is shown in Figure 7.a. Students are asked to connect two different types of switches/sensors to the microcontroller (Figure 7.b) and generate an Easy C program to navigate the robot through the Learning Factory, the Engineering Laboratory Suite of the department.



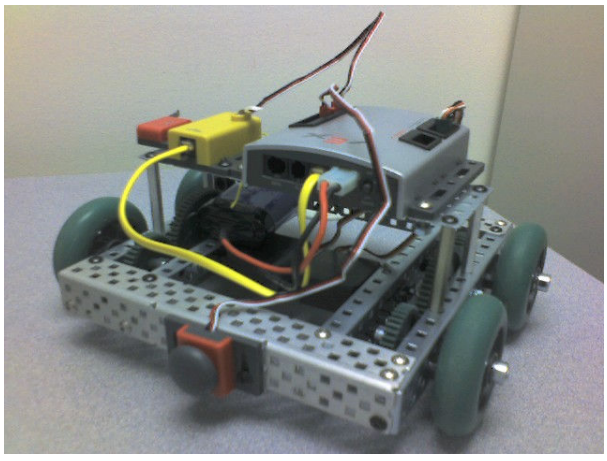


Figure 7.a Square Bot .b Square Bot with sensors attached

They plug in the necessary sensors to the VEX controller. A wide variety of input devices can be integrated into their solutions including optical shaft encoders, light sensors, ultrasonic range finders, bumper switches, and limit switches. Details of the assignment are given below[1]:

- **Objective:** To get familiar with a microcontroller based control system, its associated sensory input devices and output devices.
- **Procedure:** Each group will be given a Square Bot including VEX microcontroller, various sensors and actuators including DC continuous motors. Groups need to follow the procedure below in order to conduct the simple task of navigating through the Learning Factory (You will be given the starting and finishing point).
  - Utilizing at least two type of switches/sensors in navigation (i.e. bumper switch, ultrasonic range finder, light sensor, limit switch etc.)
  - Writing an Easy C program to conduct the navigation process.
- **Deliverables:**
  - Demonstration of successful robot operation.
  - Delivery of the robot.
  - A final report that includes the robot design including actuators and sensory inputs, the strategy utilized and associated robot program.
  - Additional credit will be given to any added value to the basic design and programs.

In Fall of 2008, multiple groups of ENGR 4400 students worked on this assignment. Each group studied the Learning Factory main laboratory area and devised a plan or strategy. Groups successfully generated Easy C programs for their robots to navigate through the area. However, each group chose the simplest of the input devices, the bumper sensor and the limit switch. The complexity was left for the programming stage. However, with guidance of the author the groups simplified their programs. The projects ended with the project deliverables being completed and turned in to the author. The experience spanned about 2 and ½ week time frame including the initial learning stage.

Discussion

In terms of the actual assignment experience, students came up with intelligent solutions to the problem. One group used time driven sequential programming to navigate through the laboratories without using the support of sensors. A limit switch was used to initiate the navigation process while the bumper switch was used to terminate it. As the robot reached its destination the bumper switch was made by the door to stop the robot operation. Another group utilized the combination of both the event and time driven sequential programming. In the first half of their logic, the group utilized time driven programming and then switched to event driven programming. The event driven programming included use of the bumper sensor and its associated logic, moving forward until the bumper sensor is made, backing up for a few seconds and making a left turn. This logic ran in an unconditional WHILE loop until the robot reached its destination. Then, the robot operation was terminated manually by using the limit switch. In addition to the conditional and unconditional WHILE loops, integer variable definition and initiation, IF and IF-ELSE statements, the Easy C laboratory functions for I/O devices, and

BREAK command for terminating the WHILE loops were utilized. Overall, this experience allowed students to employ their C++ language background in a practical and simple setting. Mechanical inaccuracies and battery power were the two major issues students needed to resolve. The groups handled these problems with ease by readjusting the programs they wrote and by charging the functional batteries long enough for consistent results respectively.

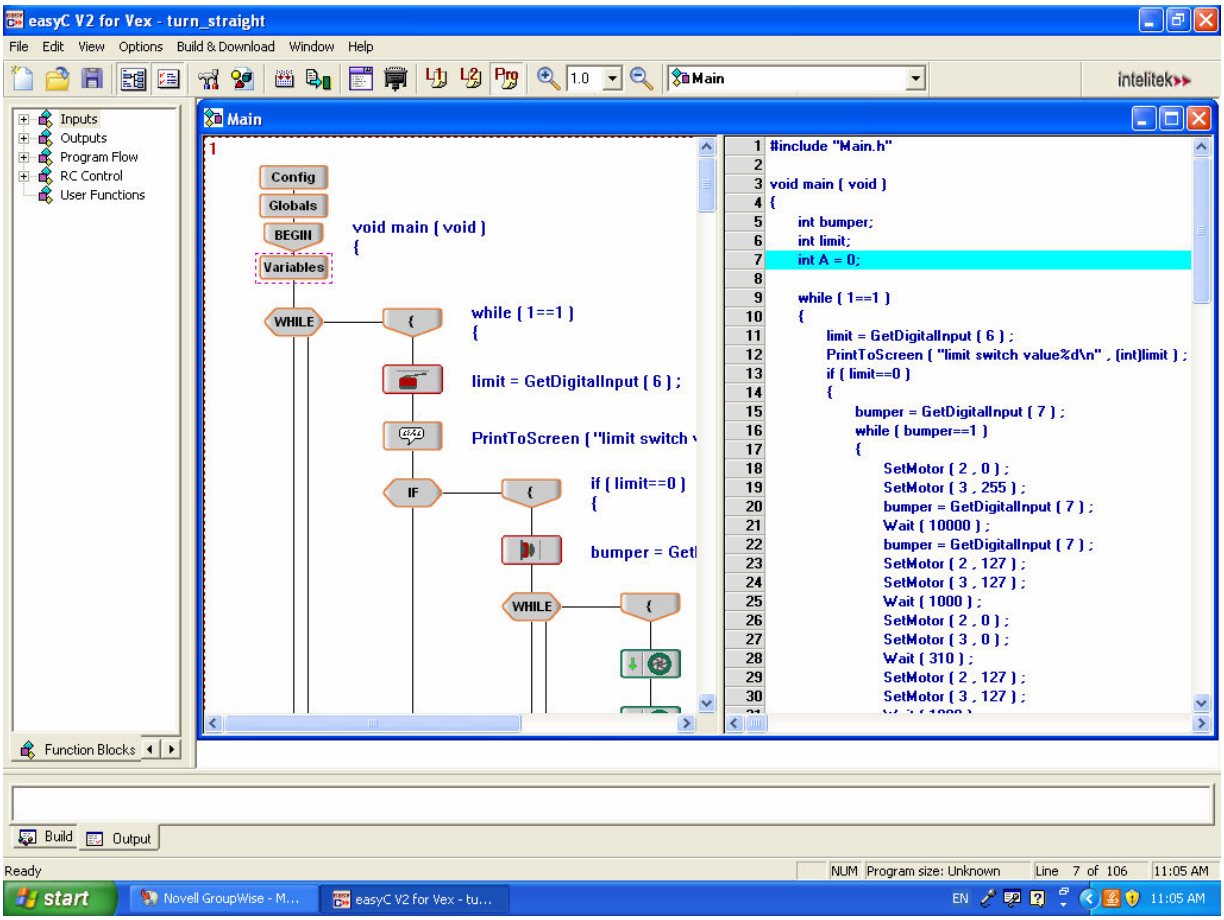


Figure 8. An assignment program in both formats

Conclusions and Future Work

The activity was a good way of adding microprocessor programming content into an industrial control course in a manufacturing engineering program. It gives students a good technical background on microcontroller based systems and associated I/O devices, and most importantly some programming skills other than the ladder logic or NI's block based G programming language. The students also gain critical knowledge allowing them to differentiate amongst various tools available including hard-wired logic, IC's, PLC's, PAC's, PC's, and microcontrollers.

Future microcontroller assignments will follow student suggestions from the course evaluations and their feedback from the reports such as a request for additional presentations on the VEX system. There are three programming environments which can work with a VEX microcontroller

system. These are the Easy C, ROBOT C, and MP Lab Development System. In the near future, a switch will be made to the ROBOT C language due its similarity to the Easy C and having more functionalities than the Easy C.

Bibliography

[1] Unpublished ENGR 4400 - DEVICE CONTROL course notes, Robert Morris University, 2007.

[2] Streib, W., J., Digital Circuits, The Goodheart-Willcox Company, Tinley Park, IL, 1997.

[3] Petruzella, F., D., Programmable Logic Controllers (3rd Edition), McGraw Hill, New York, NY, 2005.

[4] http://search.ni.com/nisearch/main/p?q=pid+example

[5] Petersen, Y., Petersen, J., VEX for the Technically Challenged by the Technically Challenged (Version 3.1), 2008.

[6] Inventor's Guide - Programming Chapter, Vex Robotics Design System, 2007.